Report:

State-dependent Quadrotor LQR Control

Sheng-Wen Cheng (shengwen.c@nycu.edu.tw)
Hsin-Ai Hung (louise.c@nycu.edu.tw)

Initial release: June 2, 2020
Last update: July 6, 2025

Contents

1 Symbols

$[\phi, \theta, \psi]^T$	Euler angles
$R_i \in SO(3)$	Rotation matrix from the body-fixed frame to the inertial frame
$[x, y, z]^T \in \mathbb{R}^3$	Position in the inertial frame
$v_B = [u, v, w]^T \in \mathbb{R}^3$	Velocity in the body-fixed frame
$\Omega = [p, q, r]^T \in \mathbb{R}^3$	Angular velocity in the body-fixed frame
$M = [\tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^3$	Torque in the body-fixed frame
$f_t \in \mathbb{R}$	Total thrust of the quadrotor
$m \in \mathbb{R}$	Mass of the quadrotor
$J \in \mathbb{R}^{3\times3}$	Inertia matrix with respect to the body-fixed frame
$T \in \mathbb{R}^{3\times3}$	Matrix mapping the body-fixed angular velocity to Euler angle rates
$\mathbf{x} \in \mathrm{R}^{12}$	State vector of the LQR controller
$\mathbf{u} \in \mathrm{R}^4$	Control vector of the LQR controller
$A \in \mathbb{R}^{12\times12}$	State transition matrix of the LQR controller
$B \in \mathbb{R}^{4\times10}$	Control matrix of the LQR controller
$C \in \mathbb{R}^{10\times12}$	Measurement matrix of the LQR controller
$Q \in \mathbb{R}^{12\times12}$	State penalty matrix of the LQR controller
$R \in \mathbb{R}^{4\times4}$	Control penalty matrix of the LQR controller
$K \in \mathbb{R}^{12\times12}$	Feedback gain matrix of the LQR controller
$X \in \mathbb{R}^{4\times10}$	Unique solution of the Continuous-time Algebraic Riccati Equation (CARE)

2 Motivation

LQR is a popular control method for linear systems. For nonlinear dynamic systems like a quadrotor, it is common to restrict the operating region and linearize the system around an equilibrium point [4]. This approach reduces the computational cost compared to continuously linearizing the nonlinear system over time, but it also decreases control performance and precision.

In this report, we studied state-dependent LQR control for a quadrotor. Instead of limiting the operating region and applying small-angle approximations, we update the state matrix A over time and compute the optimal control signal to achieve better performance and precision.

In order to calculate the optimal gain by solving the Continuous-time Algebraic Riccati Equation (CARE) in real time, we explored the Structure-Preserving

Doubling Algorithm (SDA). Based on simulation results, SDA is 5.1 times faster than MATLAB's built-in care function while maintaining high accuracy.

3 Mathematical model

3.1 Euler Angles and Rotation Matrix

Euler angles were first introduced by Leonhard Euler to describe the orientation of a rigid body. From the Euler angles, we can generate three transformation matrices.

Rotation about the x-axis by an angle ϕ:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$$

Rotation about the y-axis by an angle θ:

$$R_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

Rotation about the z-axis by an angle ψ:

$$R_z(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By combining the three matrices in the order of x-y-z rotation:

$$R_{zyx} = R_z(\psi)R_y(\theta)R_x(\phi)$$

$$= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

The rotation matrix belongs to the special orthogonal group $SO(3)$, , which is not commutative under multiplication. It also satisfies the properties $R^{-1} = R^T$ and $det(R) = 1$. Note that there are several ways to construct a rotation matrix. However, when using Euler angles, a singularity occurs at $\theta = \pm 90°$, known as gimbal lock.

3.2 Euler Angle Rates and Angular Velocity

Unlike the angular velocity vector, each Euler angle rate $\dot{\phi}, \dot{\theta}, \dot{\psi}$ lies in a different coordinate frame. To calculate the angular velocity in the body-fixed frame, we

must convert each rate individually into the same frame as follows:

$$\Omega = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_z \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_z R_y \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}$$

This can be expressed as:

$$\Omega = J \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

The matrix J is called the Jacobian matrix. Its inverse J^{-1} allows us to compute Euler angle rates from the body-fixed angular velocity Ω:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J^{-1}\Omega$$

$$T \equiv J^{-1} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}$$

3.3 Quadrotor Dynamics

The compact form of the quadrotor dynamics is given as follows [5]:

$$\dot{x} = v$$

$$m\dot{v} = mge_3 - f_t R e_3$$

$$\dot{R} = R\hat{\Omega}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M$$

Although this form is slightly different from the one we derive in the next section for controller design, we use these equations to update the dynamics in the simulator.

3.4 Deriving the State-Space Model for the Quadrotor

In this section, we derive the 12-state quadrotor dynamic equations used in the controller design. From the previous section, the transformation from body-fixed angular velocity to Euler angle rates is:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T \cdot \Omega$$

$$T = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}$$

Expanding this, we obtain the rotation rate dynamics:

$$\begin{cases} \dot{\phi} = p + r(c\phi t\theta) + q(s\phi + t\theta) \\ \dot{\theta} = q(c\phi) - r(s\phi) \\ \dot{\psi} = r\frac{c\phi}{c\theta} + q\frac{s\phi}{c\theta} \end{cases}$$

The transformation of translational velocity from the body-fixed frame to the inertial frame is:

$$v_I = R \cdot v_B$$

Expanding this yields the translational velocity dynamics:

$$\begin{cases} \dot{x} = w(s\phi s\psi + c\phi c\psi s\theta) - v(c\phi s\psi - c\psi s\phi s\theta) + u(c\psi c\theta) \\ \dot{y} = v(c\phi c\psi + s\phi s\psi s\theta) - w(c\psi s\phi - c\phi s\psi s\theta) + u(c\theta s\psi) \\ \dot{z} = w(c\phi c\theta) - u(s\theta) + v(c\theta s\phi) \end{cases}$$

The Euler equation, describing the relationship between torque, angular velocity, and angular acceleration, is:

$$J\dot{\Omega} + \Omega \times J\Omega = M$$

Expanding this gives:

$$\begin{cases} \tau_x = \dot{p}I_x - qrI_y + qrI_z \\ \tau_y = \dot{q}I_y + prI_x - prI_z \\ \tau_z = \dot{r}I_z - pqI_x + pqIy \end{cases}$$

We will later reorder these to extract angular acceleration equations.
From Newton's second law, the force equation is:

$$m(\Omega \times v_B + \dot{v}_B) = f_B$$

Expanding this gives:

$$\begin{cases} -mg(s\theta) = m(\dot{u} + qw - rv) \\ mg(c\theta s\phi) = m(\dot{v} - pw + ru) \\ mg(c\theta c\phi) - f_t = m(\dot{w} + pv - qu) \end{cases}$$

Where:

$$f_b = Rm\dot{v} = R(mge_3 - f_t Re_3) = Rmge_3 - f_t e_3$$

Combining all, the full quadrotor dynamics are:

$$\mathbf{f} = \begin{cases} \dot{\phi} = p + r(c\phi t\theta) + q(s\phi t\theta) \\ \dot{\theta} = q(c\phi) - r(s\phi) \\ \dot{\psi} = r\frac{c\phi}{c\theta} + q\frac{s\phi}{c\theta} \\ \dot{p} = \frac{I_y - I_z}{I_x} rq + \frac{\tau_x}{Ix} \\ \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_y}{I_y} \\ \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_z}{I_z} \\ \dot{u} = rv - qw - g(s\theta) \\ \dot{v} = pw - ru + g(s\phi c\theta) \\ \dot{w} = qu - pv + g(c\theta c\phi) - \frac{f_t}{m} \\ \dot{x} = w(s\phi s\psi + c\phi c\psi s\theta) - v(c\phi s\psi - c\psi s\phi s\theta) + u(c\psi c\theta) \\ \dot{y} = v(c\phi c\psi + s\phi s\psi s\theta) - w(c\psi s\phi - c\phi s\psi s\theta) + u(c\theta s\psi) \\ \dot{z} = w(c\phi c\theta) - u(s\theta) + v(c\theta s\phi) \end{cases}$$

4 LQR control

4.1 Linear Control of the Quadrotor

The state variables for the designed linear controller are defined as:

$$\mathbf{x} = \begin{bmatrix} \phi, \theta, \psi, p, q, r, u, v, w, x, y, z \end{bmatrix}^T$$

The control input vector is defined as:

$$\mathbf{u} = \begin{bmatrix} f_t, \tau_x, \tau_y, \tau_z \end{bmatrix}^T$$

The linear state-space representation of the system is given by:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

4.2 Linearization

Using the dynamic equations derived in the previous section, we obtain the A and B matrices by linearizing around the equilibrium point:

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\mathbf{x_e},\mathbf{u}=\mathbf{u_e}} = \begin{bmatrix} \frac{\partial \dot{\phi}}{\partial \mathbf{x}} \\ \frac{\partial \dot{\theta}}{\partial \mathbf{x}} \\ \frac{\partial \dot{\psi}}{\partial \mathbf{x}} \\ \frac{\partial \dot{p}}{\partial \mathbf{x}} \\ \frac{\partial \dot{q}}{\partial \mathbf{x}} \\ \frac{\partial \dot{r}}{\partial \mathbf{x}} \\ \frac{\partial \dot{u}}{\partial \mathbf{x}} \\ \frac{\partial \dot{v}}{\partial \mathbf{x}} \\ \frac{\partial \dot{w}}{\partial \mathbf{x}} \\ \frac{\partial \dot{x}}{\partial \mathbf{x}} \\ \frac{\partial \dot{y}}{\partial \mathbf{x}} \\ \frac{\partial \dot{z}}{\partial \mathbf{x}} \end{bmatrix}$$

Each partial derivative row vector is derived as follows:

$$\frac{\partial \dot{\phi}}{\partial \mathbf{x}} = \left[-rs\phi t\theta + qc\phi t\theta, rc\phi sec^2\theta + qs\phi sec^2\theta, 0, 1, s\phi t\theta, c\phi t\theta, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{\theta}}{\partial x} = \left[-qs\phi - rc\phi, 0, 0, 0, c\phi, -s\phi, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{\psi}}{\partial \mathbf{x}} = \left[-r\frac{s\phi}{c\theta} + q\frac{c\phi}{c\theta}, rc\phi sec\theta t\theta + qs\phi sec\theta t\theta, 0, 0, \frac{s\phi}{c\theta}, \frac{c\phi}{c\theta}, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{p}}{\partial \mathbf{x}} = \left[0, 0, 0, 0, \frac{I_y - I_z}{I_x}r, \frac{I_y - I_x}{I_y}q, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{q}}{\partial \mathbf{x}} = \left[0, 0, 0, \frac{I_z - I_x}{I_y}r, 0, \frac{I_z - Ix}{I_y}p, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{r}}{\partial \mathbf{x}} = \left[0, 0, 0, \frac{Ix - Iy}{I_z}q, \frac{Ix - Iy}{I_z}p, 0, 0, 0, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{u}}{\partial \mathbf{x}} = \left[0, -gc\theta, 0, 0, -w, v, 0, r, -q, 0, 0, 0\right]$$

$$\frac{\partial \dot{v}}{\partial \mathbf{x}} = \left[gc\phi c\theta, -gs\phi s\theta, 0, w, 0, -u, -r, 0, p, 0, 0, 0\right]$$

$$\frac{\partial \dot{w}}{\partial \mathbf{x}} = \left[-gc\theta s\phi, -gs\theta c\phi, 0, -v, u, 0, q, -p, 0, 0, 0, 0\right]$$

$$\frac{\partial \dot{x}}{\partial \mathbf{x}} = \begin{bmatrix} w(c\phi s\psi - s\phi s\psi s\theta) + v(s\phi s\psi + c\psi c\phi s\theta), \\ w(c\phi c\psi c\theta) + v(c\psi s\phi c\theta) - u(c\psi s\theta), \\ w(s\phi c\psi - c\phi s\psi s\theta) - v(c\phi c\psi + s\psi s\phi s\theta) - u(s\psi c\theta), \\ 0, \\ 0, \\ 0, \\ 0, \\ c\psi c\theta, \\ -c\phi s\psi + c\psi s\phi s\theta, \\ s\phi s\psi + c\phi c\psi s\theta, \\ 0, \\ 0, \\ 0 \end{bmatrix}$$

$$\frac{\partial \dot{y}}{\partial \mathbf{x}} = \begin{bmatrix} v(-s\phi c\psi + c\phi s\psi s\theta) - w(c\psi c\phi + s\phi s\psi s\theta), \\ v(s\phi s\psi c\theta) + w(c\phi s\psi c\theta) - u(s\theta s\psi), \\ v(-c\phi s\psi + s\phi c\psi s\theta) + w(s\psi s\phi + c\phi c\psi s\theta) + u(c\theta c\psi), \\ 0, \\ 0, \\ 0, \\ 0, \\ c\theta s\psi, \\ c\phi c\psi + s\phi s\psi s\theta, \\ -c\psi s\phi + c\phi s\psi s\theta, \\ 0, \\ 0, \\ 0 \end{bmatrix}$$

$$\frac{\partial \dot{z}}{\partial \mathbf{x}} = \begin{bmatrix} -w(s\phi c\theta) + v(c\theta c\phi), \\ -w(c\phi s\theta) - uc\theta - v(s\theta s\phi), \\ 0, \\ 0, \\ 0, \\ 0, \\ -s\theta, \\ c\theta s\phi, \\ c\phi c\theta, \\ 0, \\ 0, \\ 0 \end{bmatrix}$$

Now for the B matrix:

$$B = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_{\mathbf{x}=\mathbf{x_e},\mathbf{u}=\mathbf{u_e}} = \begin{bmatrix} \frac{\partial \dot{\phi}}{\partial u} \\ \frac{\partial \dot{\theta}}{\partial u} \\ \frac{\partial \dot{\psi}}{\partial u} \\ \frac{\partial \dot{p}}{\partial u} \\ \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{r}}{\partial u} \\ \frac{\partial \dot{u}}{\partial u} \\ \frac{\partial \dot{v}}{\partial u} \\ \frac{\partial \dot{w}}{\partial u} \\ \frac{\partial \dot{x}}{\partial u} \\ \frac{\partial \dot{y}}{\partial u} \\ \frac{\partial \dot{z}}{\partial u} \end{bmatrix}$$

Which simplifies to:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

4.3 Linear Quadratic Regulator (LQR) Control

The objective of optimal control is to minimize a cost function (which reflects performance and physical constraints) and stabilize the system by determining the appropriate control signal. The cost function for the Linear Quadratic Regulator is defined as:

$$J(\mathbf{x}, \mathbf{u}) = \int_0^\infty (\tilde{\mathbf{x}}^T Q \tilde{\mathbf{x}} + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}}) dt$$

Given a desired state vector $\mathbf{x_0}$ and feedforward control vector $\mathbf{u_0}$, the feedback control signal and optimal gain are defined as:

$$\mathbf{u} = \mathbf{u_0} - K(\mathbf{x} - \mathbf{x_0})$$

$$K = R^{-1}B^T X$$

The optimal gain matrix K is computed from the unique solution X of the Continuous-Time Algebraic Riccati Equation (CARE), if a solution exists:

$$A^T X + XA - XGX + H$$

Where

$$G = BR^{-1}B^T$$
$$H = C^T QC$$

The matrix C is designed as

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that this design omits direct control over ϕ and θ, since the quadrotor is an underactuated system. Translational control requires ϕ and θ to be indirectly adjusted rather than directly controlled.

5 Numerical method for solving CARE

5.1 Traditional methods for solving CARE

The Hamiltonian matrix, defined as $\mathscr{H} = \begin{bmatrix} A & -G \\ -H & -A^T \end{bmatrix}$ is closely associated with the CARE used in optimal control of the LQR controller. Under the assumptions that matrix A is stabilizable and detectable, the CARE is guaranteed to have a unique symmetric positive semi-definite solution X.

Several algorithms have been proposed to solve for X. One well-known method, implemented in MATLAB and proposed by Laub [2], reformulates the CARE as an eigenvalue problem $\mathscr{H}x = \lambda x$ and applies the QR algorithm. However, the QR algorithm does not preserve the Hamiltonian structure or the associated spectral properties.

Other classical methods include: (1) Fixed-point iteration method, which applies DARE (Discrete-time Algebraic Riccati Equation) and calculates the converged solution X with the sequence of $\{X_k\}$, and (2) Newton's method are widely used.

5.2 Structure-preserving Doubling Algorithm

Instead of producing a linear sequence $\{X_k\}$, the Structure-Preserving Doubling Algorithm (SDA) [1] generates a sequence $\{X_{2^k}\}$ that converges quadratically to the stabilizing solution. Due to its structure-preserving nature, especially for the Hamiltonian matrix $\mathscr{H}$, the SDA yields fast and highly accurate solutions.

Below is the algorithm of the SDA:

Algorithm 1: Structure-preserving Doubling Algorithm

Input: $\mathscr{H} = \begin{bmatrix} A & -G \\ -H & -A^T \end{bmatrix} \in \mathscr{H}$ with $\sigma(\mathscr{H}) \cap Im = \emptyset; \epsilon$

Output: The stabilizing solution $X = X^T \geq 0$ to the CARE

1 Compute

$\hat{A}_0 \leftarrow I + 2\hat{\gamma}(A_{\hat{\gamma}} + GA_{\hat{\gamma}}^T H)^{-1}$;

$\hat{G}_0 \leftarrow 2\hat{\gamma}A_{\hat{\gamma}}^{-1}G(A_{\hat{\gamma}}^T + HA_{\gamma}^{-1}G)^{-1}$;

$\hat{H}_0 \leftarrow 2\hat{\gamma}(A^T\hat{\gamma} + HA_{\hat{\gamma}}^{-1}G)^{-1}HA_{\gamma}^{-1}$;

$j \leftarrow 0$;

2 Do until convergence:

Compute

$\hat{A}_{j+1} \leftarrow \hat{A}_j(I + \hat{G}_j\hat{H}_j)^{-1}\hat{A}_j$;

$\hat{G}_{j+1} \leftarrow \hat{G}_j + \hat{A}_j\hat{G}_j(I + \hat{H}_j;\hat{G}_j)^{-1}\hat{A}_j^T$;

$\hat{H}_{j+1} \leftarrow \hat{H}_j + \hat{A}_j^T(I + \hat{H}_j;\hat{G}_j)^{-1}\hat{H}_j\hat{A}_j$;

If $||\hat{H}_j = \hat{H}_{j-1} \leq \epsilon\hat{H}_j||$, Stop;

End

3 Set $X \leftarrow \hat{H}_j$

6 Numerical Method for Updating the Rotation Matrix

In this section, we introduce a method for updating the rotation matrix using angular velocity, based on the approach described in [3].

6.1 Updating the Rotation Matrix with Angular Velocity

The well-known kinematic relationship between a position vector's tangent velocity and angular velocity is given by:

$$v = \frac{dr}{dt} = \omega(t) \times r(t)$$

Here, $\omega(t)$ is the angular velocity vector. We can update the position vector by integrating the kinematic equation over time:

$$r(t) = r(0) + \int_0^t d\omega(\tau) \times r(\tau)d\tau$$

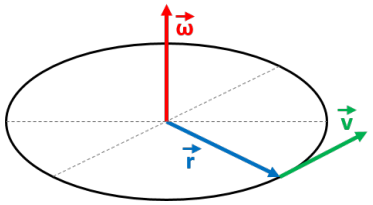

Figure 1: Tangent velocity, angular velocity and rotation vectors

$$= r(0) + \int_0^t d\theta(\tau) \times r(\tau)$$

Since the column vectors of a rotation matrix represent unit vectors pointing in the three orthogonal axes of a body frame, we can apply this same update rule to each column vector. Thus, the rotation matrix can be incrementally updated by applying this differential rotation to its columns.

$$R(t+dt) = R(t) \begin{bmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{bmatrix}$$

$$d\theta_x = \Omega_x dt$$

$$d\theta_y = \Omega_y dt$$

$$d\theta_z = \Omega_z dt$$

6.2 Rotation Matrix Renormalization

Numerical errors introduced during integration can gradually degrade the orthogonality of the rotation matrix. Therefore, orthogonalization and renormalization are required after each integration step to restore the matrix's validity as a rotation matrix.

Suppose we have a rotation matrix R:

$$R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$$

Let X and Y represent the first two rows of R:

$$X = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}, \; Y = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}$$

Ideally, the dot product $X \cdot Y$ should be zero. Any nonzero result indicates a loss of orthogonality due to numerical drift:

$$error = X \cdot Y = X^T Y = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \end{bmatrix} \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}$$

To correct this, we distribute half the error to each vector and rotate them slightly in opposite directions:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}_{orthogonal} = X_{orthogonal} = X - \frac{error}{2} Y$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}_{orthogonal} = Y_{orthogonal} = Y - \frac{error}{2} X$$

To restore the orthogonality of the third row Z, we compute it as the cross product of the corrected X and Y,vectors:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}_{orthogonal} = Z_{orthogonal} = X_{orthogonal} \times Y_{orthogonal}$$

Finally, we normalize all three vectors to obtain unit length:

$$X_{normalized} = \frac{X_{orthogonal}}{||X_{orthogonal}||}$$

$$Y_{normalized} = \frac{Y_{orthogonal}}{||Y_{orthogonal}||}$$

$$Z_{normalized} = \frac{Z_{orthogonal}}{||Z_{orthogonal}||}$$

7 Simulation Result

7.1 Trajectory Tracking

We simulated the quadrotor tracking the following circular trajectory:

$$[p_x, p_y, p_z] = [0.5cos(0.25\pi t), 0.5sin(0.25\pi t), -0.1t]$$

$$[v_x, v_y, v_z] = [-0.5sin(0.25\pi t), 0.5cos(0.25\pi t), -0.1]$$

The desired setpoint for the LQR controller is given as:

$$x_0 = [0, 0, 0, 0, 0, 0, v_x, v_y, v_z, p_x, p_y, p_z]$$

The following figures show the results of the simulation:

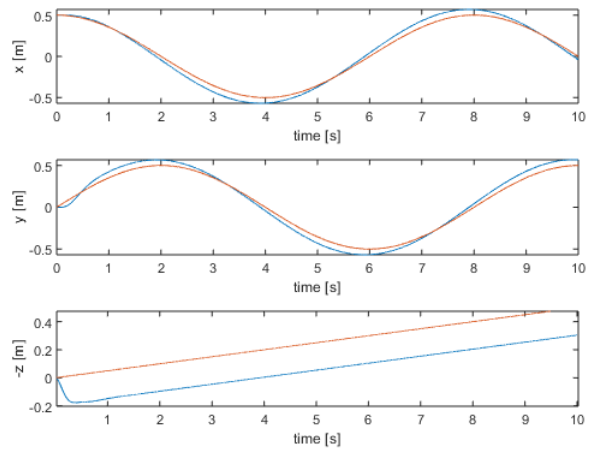

Figure 2: Position tracking (blue: true state, orange: desired state)

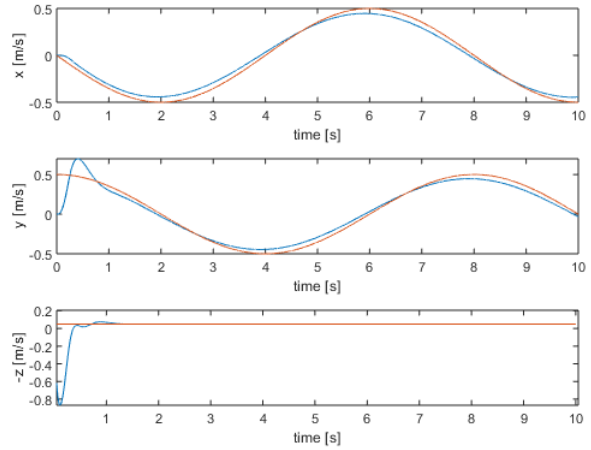

Figure 3: Velocity tracking (blue: true state, orange: desired state)

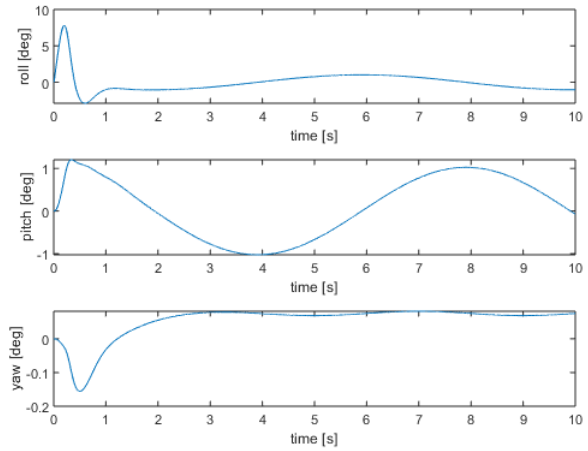

Figure 4: Attitude

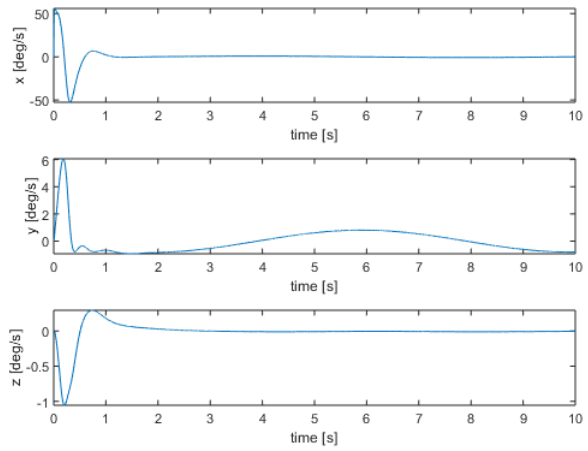

Figure 5: Angular velocity

7.2 Performance of Structure-Preserving Doubling Algorithm

We ran the simulation using both MATLAB's built-in care function and the Structure-Preserving Doubling Algorithm (SDA) to solve the CARE. The simulation results show that SDA is 5.1 times faster and achieves slightly higher precision compared to MATLAB's care function, as demonstrated in Figure 6

and Figure 7.

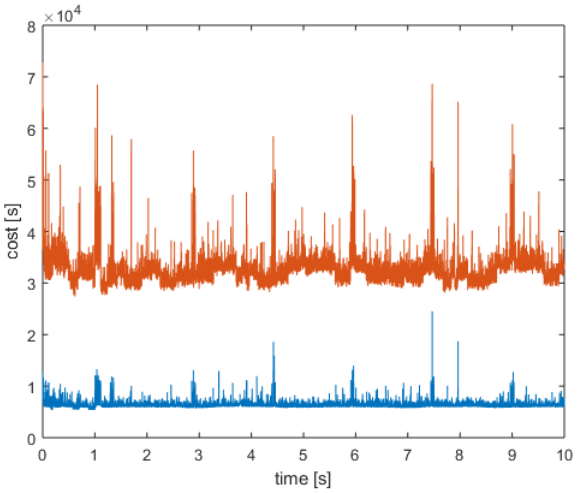

Figure 6: Time cost comparison (blue: SDA, orange: MATLAB care)

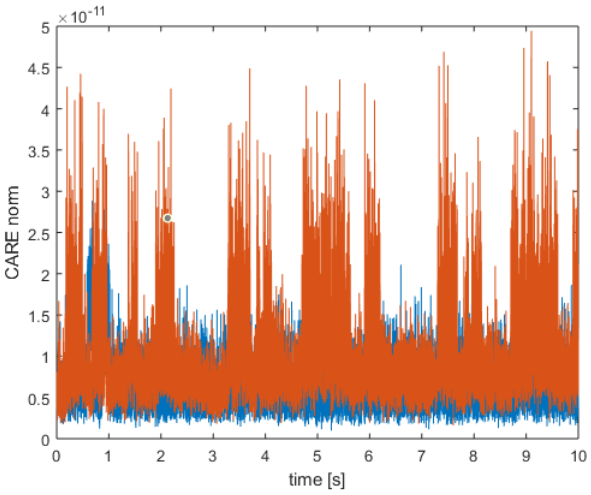

Figure 7: Precision comparison (blue: SDA, orange: MATLAB care)

References

[1] W.-W.Lin E.K.-W.Chu, H.-Y.Fan. A structure-preserving doubling algorithm for continuous-time algebraic riccati equations. *Linear Algebra and*

its Applications.

[2] Alan J. Laub. A schur method for solving algebraic riccati equations. *IEEE Transactions on Automatic Control*.

[3] William Premerlani and Paul Bizard. Direction cosine matrix imu: Theory.

[4] Francesco Sabatino. Quadrotor control: modeling, nonlinear control design, and simulation.

[5] N. Harris McClamroch Taeyoung Lee, Melvin Leok. Geometric tracking control of a quadrotor uav on se(3). *IEEE Conference on Decision and Control*.