Report:

State-dependent Quadrotor LQR Control

June 2, 2020

Cheng, Sheng-Wen
Hung, Hsin-Ai

Contents

1 Motivation

LQR is a popular control method for linear system. For non-linear dynamics system like quadrotor, it is common to restrict the operating region and linearize the system at a equilibrium point [4]. This reduces the computational cost of linearizing the non-linear system over time but also decreases the control performance and precision.

In this report, we studied the state-dependent LQR quadrotor control. Instead of limiting the operating region of quadrotor and apply small angle approximation, we update the state matrix A over time and calculate the optimal control signal to gain better performace and precision.

In order to calculate the optimal gain by solving CARE (Continuous-time Algebraic Riccati Equation) in real-time, we explored the algorithm "Structure-preserving Doubling Algorithm" (SDA). By the simulation result, SDA is 5.1 times faster than the MATLAB built-in function care and having great accuracy.

2 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 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 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 frame
$T \in \mathbb{R}^{3\times3}$	matrix that mapping the body-fixed frame angular velocity to euler angles' rate
$\mathbf{x} \in \mathrm{R}^{12}$	state vector of LQR controller
$\mathbf{u} \in \mathrm{R}^{4}$	control vector of 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
$C \in \mathbb{R}^{12\times12}$	measurement matrix of the LQR controller, which is equal to the identity matrix in the report
$Q \in \mathbb{R}^{12\times12}$	state penalty matrix of LQR controller
$R \in \mathbb{R}^{4\times4}$	control penalty matrix of 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 CARE (Continuous-time Algebraic Riccati Equation)

3 Mathematical model

3.1 Euler angles and Rotation Matrix

Euler angles was first introduced by Leonhard Euler to describe the orientation of rigidbody. We can generate three transformation matrices from Euler angles:

Rotate x axis with ϕ angle:

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

Rotate y axis with θ angle:

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

Rotate z axis with ψ angle:

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

Combining three matrices with the multiplication order of x-y-z:

$$R_{zyx} = R_z(\psi)R_y(\theta)R_x(\phi)$$
$$= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

Rotation matrix is a member of special orthogonal group $SO(3)$, which is not multiplicative commutative, and also has the property of $R^{-1} = R^T$ and $det(R) = 1$

Notice that there are several ways to generate Rotation Matrix. However with Euler angles, the singularity happens at $\theta = \pm 90°$, which is called "gimbal lock".

3.2 Euler angles' rate and angular velocity

Unlike the angular velocity vector, each joint speed $\dot{\phi}, \dot{\theta}, \dot{\psi}$ of Euler angles are sitting on different coordinate frames. To calculate the body-fixed frame angular velocity, we need to convert them individually to a same coordinate frame as follow:

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

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

The matrix J is called the Jacobian matrix. The inverse Jacobian matrix J^{-1} can help us calculate Euler angles' rate from body-fixed frame 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 quadrotor dynamics are given as follow [5]:

$$\dot{x} = v$$

$$m\dot{v} = mge_3 - f_t Re_3$$

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

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

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

3.4 Deriving state-space model for Quadrotor

In this section, we will derive 12 quadrotor dynamics equations for controller design in later section:

As the derivation in last section, the transformation of body-fixed frame angular velocity to Euler angle's rate is given as:

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

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

We get the dynamics equations of rotation rate after doing expansion:

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

The transformation of transnational velocity from body-fixed frame to inertial frame is given as:

$$v_I = R \cdot v_B$$

We get the dynamics equations of velocity after doing expansion:

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

The Euler's equation for describing the relation between torque, angular velocity and angular acceleration is given as:

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

Expansion yields the following equations. we will later reorder them to get the dynamics equations of angular acceleration:

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

By Newton's law, we have the equation of force as follow:

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

Similarly, we get three equations after expansion. we will later reorder them to get the dynamics equations of transnational acceleration:

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

where,

$$f_b = Rm\dot{v} = R(mge_3 - f_tRe_3) = Rmge_3 - f_te_3$$

Finally, the full dynamics of quadrotor can be written as:

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

4 LQR control

4.1 Linear control of Quadrotor

The state variables of the designed linear controller is described as:

$$\mathbf{x} = \left[\phi, \theta, \psi, p, q, r, u, v, w, x, y, z\right]^T$$

The control input of the controller is designed as:

$$\mathbf{u} = \left[f_t, \tau_x, \tau_y, \tau_z\right]^T$$

Finally, the state space representation of the linear system is given as:

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

4.2 Linearization

Using the dynamics equations derived in last section, we can get A and B matrix by doing linearization around the equilibrium point:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.3 Linear Quadratic Regulator control

The objective of the optimal control is to minimizes a cost function (which represents the physical constraints) then stabilize the system by determine the control signal.

The cost function of the Linear Quadratic Regulator is defined as:

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

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

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

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

The optimal gain can be calculated by the unique solution of the Continuous-time Algebraic Riccati Equation (CARE) if it exists:

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

$$G = BR^{-1}B^T$$

$$H = C^T QC$$

5 Numerical method for solving CARE

5.1 Traditional methods for solving CARE

Hamiltonian matrix $\mathscr{H} = \begin{bmatrix} A & -G \\ -H & -A^T \end{bmatrix}$ was derived and associated with CARE for the optimal control of LQR controller. Under the assumptions of stabilizable and detectable of matrix A, CARE is proven to has a unique symmetric positive semi-definite solution X.

Several algorithm for solving X have been proposed by researchers. The one used by MATLAB, [2] published by Laub, computes X by applying QR algorithm and changes the problem to the eigenvalue problem $\mathscr{H}x = \lambda x$. Unfortunately, the QR algorithm preserves neither the Hamiltonian structure nor the associated eigenvalues.

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

$$X_{k+1} = \hat{A}^T X_k (I + \hat{G} X_k)^{-1} \hat{A} + \hat{H}$$

5.2 Structure-preserving Doubling Algorithm

Instead of producing the sequence of $\{X_k\}$, a novel algorithm: Structure-preserving Doubling Algorithm (SDA) [1] are able to generate the stable solution X with sequence of $\{X_{2^k}\}$. By the same time, because of its strong structure-preserving property (of Hamiltonian matrix $\mathscr{H}$), it is able to generate a high accuracy solution. Therefore, SDA is fast and accurate.

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 Rotation Matrix

In this section, we will introduce the method for updating Rotation Matrix with angular velocity reffering from [3].

6.1 Updating Rotation Matrix with angular veolcity

The famous formula of tangent velocity change with respect to the angular velocity is given as:

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

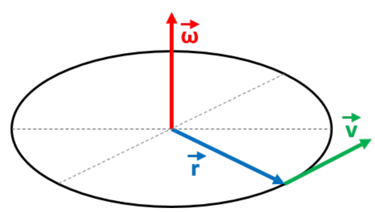

Figure 1: Tangent velocity, angular velocity and rotation vector

Where $\omega(t)$ is the rotation rate vector. We can update the position vector by integrating the above kinematics equation:

$$\begin{aligned} r(t) &= r(0) + \int_0^t d\omega(\tau) \times r(\tau) d\tau \\ &= r(0) + \int_0^t d\theta(\tau) \times r(\tau) \end{aligned}$$

Since column vectors of the Rotation Matrix represent the vectors point to three orthogonal directions in space. We apply the above formula to update the Rotation Matrix:

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

$$d\theta_x = \Omega_x dt$$

$$d\theta_y = \Omega_y dt$$

$$d\theta_z = \Omega_z dt$$

6.2 Rotation Matrix renormalization

Numerical error caused by the integration will gradually reduce the orthogonality property of Rotation Matrix therefore requires orthogonalization and renormalization after every integration step.

Suppose we have a Rotation Matrix R:

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

We compute the dot product of X and Y, which is supposed to be zero. The result is a measure of how much X and Y raws are rotating towards each others:

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

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

We apportion half of the error each to the X and Y rows, and approximately rotate the X and Y rows in the opposite direction by cross coupling:

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

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

To adjust Z row to be orthogonal to X and Y, we can simply let new Z be the cross product of X and Y rows:

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

The final step is to do vector normalization, which can be done by re-scaling X, Y and Z with their own norm:

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

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

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

7 Simulation result

7.1 Trajectory tracking

We simulated quadrotor to track the following circular trajectroy:

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

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

The desired setpoint for LQR controller is given as:

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

Notice that the desired zero attitude $(\phi, \theta, \psi) = (0°, 0°, 0°)$ actually violates quadrotors dynamics since the position can only be changed after tilting the angles. This implies quadrotor need large tilting angles to track a fast changing trajectory. In this case, the equilibrium point may be far from $(0°, 0°, 0°)$ makes linearization illegal and breaks the stability. The solution is to plan the angular trajectory which follows the constraints of quadrotors dynamics.

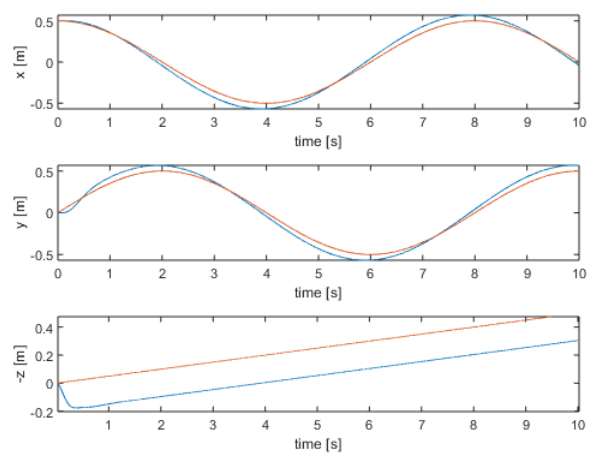

Figure 2: Position tracking (blue: true state, orange: desired state)

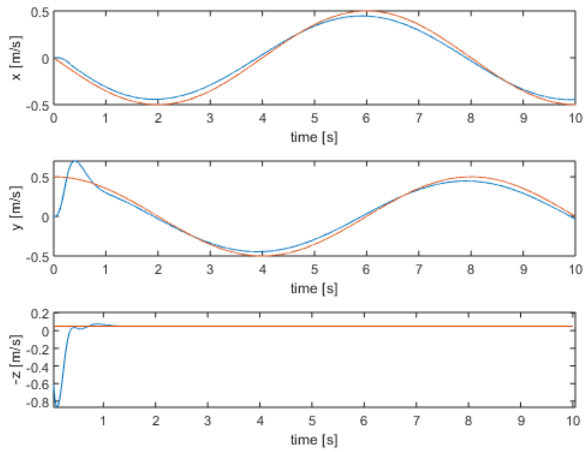

Figure 3: Velocity tracking (blue: true state, orange: desired state)

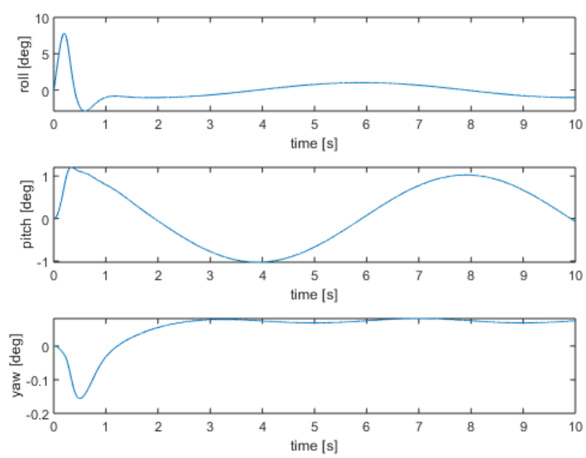

Figure 4: Attitude

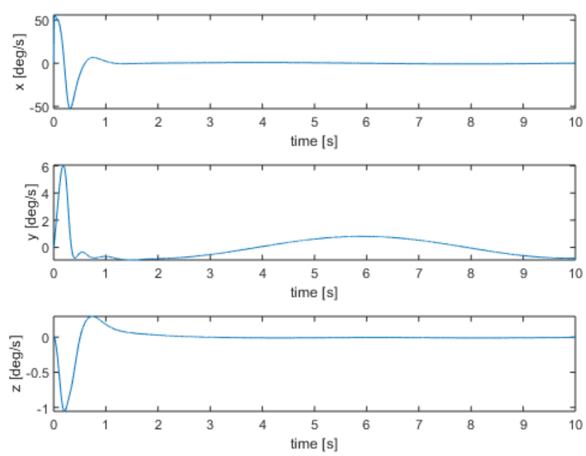

Figure 5: Angular velocity

7.2 Performance of Structure-preserving Doubling Algorithm

We ran the simulation and solved CARE with MATLAB built-in function care and SDA. Simulation result shows that SDA is 5.1 times faster and got slightly higher precision than MATLAB care function as Figure 6 and Figure 7.

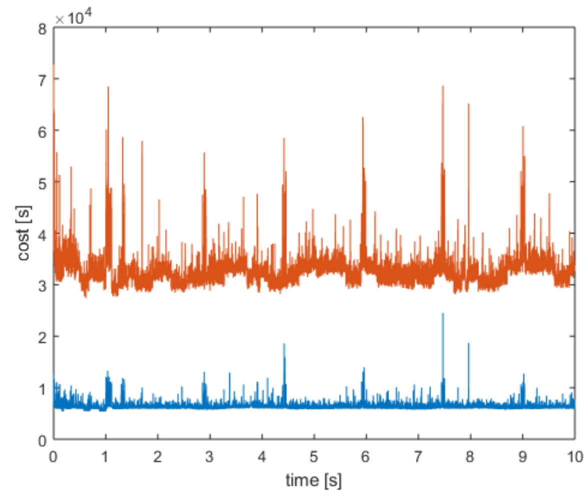

Figure 6: Time cost comparison (blue: SDA, orange: MATLAB care)

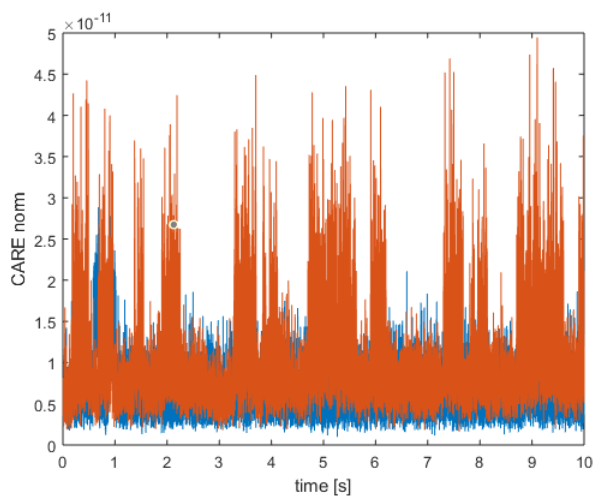

Figure 7: Precision comparison (blue: SDA, orange: MATLAB care)

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*.