Derivation of Equations of Motion and Model-based Control for 3D Rigid Body Approximation of LUVOIR

William Bentz*, Lia Sacks†

June 2018

1 Introduction

The Large UV/Optical/IR Surveyor (LUVOIR) is a conceptual space observatory under development by NASA Goddard Space Flight Center for the Astronomy and Astrophysics Decadal Survey. As with similar space telescopes, e.g., James Webb, this spacecraft will have ultra-stringent pointing stability requirements, i.e., on the order of miliarcseconds. Thus, the design of the LUVOIR attitude control system (ACS) may benefit from a model-based approach. In this document, we present an approximate model of the LUVOIR composed of two 3D rigid bodies connected by a 1-DOF rotary joint. From this model, we derive the nonlinear equations of motion, following the approach of Eric T. Stoneking [1], and linearize them about an arbitrary reference orientation. Using the subsequent state space representation, it is straightforward to compute a set of control gains (e.g., via the "lqr" or "place" commands in MATLAB). We conclude this memo with a comparison of our LQR approach with the existing PID slew controller designed by a former NASA intern.

2 System Model

We model the spacecraft as two rigid bodies connected by a 1-DOF rotary joint. The two bodies are referred to as the spacecraft and the payload respectively. The spacecraft's body-fixed frame $F_{sc} = [\hat{x}_{sc}\ \hat{y}_{sc}\ \hat{z}_{sc}]$ may be defined relative to an inertial frame $F_N = [\hat{x}_N\ \hat{y}_N\ \hat{z}_N]$ via $F_{sc}^T = \mathcal{O}_{sc/N} F_N^T$ where $\mathcal{O}_{sc/N}$ is the *orientation matrix* of F_{sc} relative to F_N. In other texts, $\mathcal{O}_{sc/N}$ is often referred to as the *direction cosine matrix* of the spacecraft. However, it should be noted that this is distinctly different than the rotation matrix from F_N to F_{sc}, i.e., the transpose. For a truly rigorous description of rotation formalisms, see [2]. We parameterize $\mathcal{O}_{sc/N}$ as 3-2-1 sequence of intrinsic Euler angle rotations by yaw angle ψ, pitch angle θ, and roll angle ϕ respectively:

$$\mathcal{O}_{sc/N} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

*PhD Candidate, University of Michigan, Department of Aerospace Engineering
†Aerospace Engineer, NASA Goddard Space Flight Center (591)

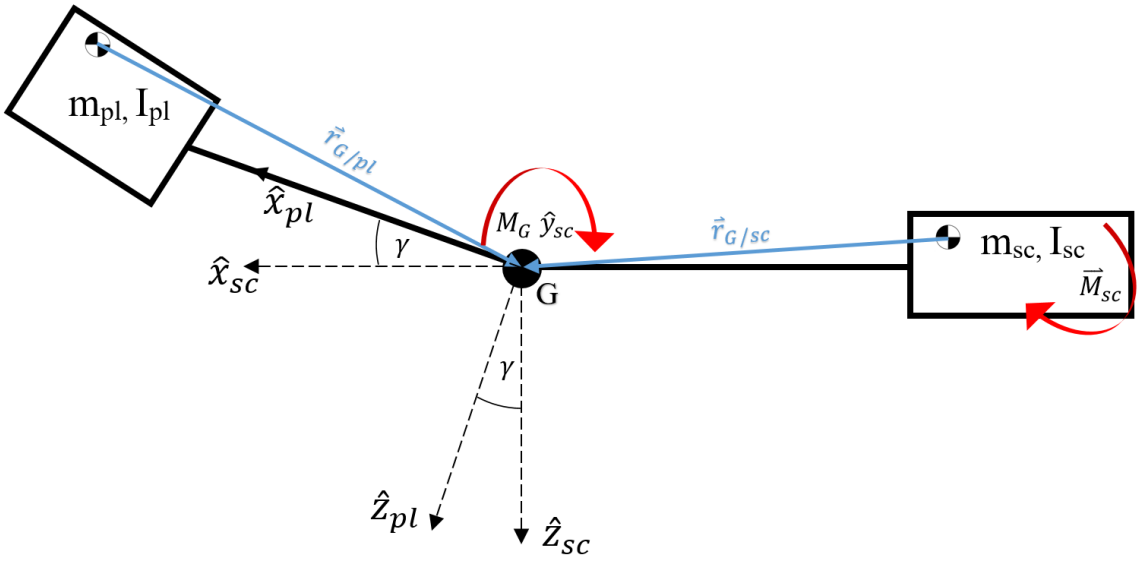

Figure 1: The spacecraft and payload bodies and frames are illustrated above. The two bodies interface at a 1-DOF rotary joint located a G that permits rotation about the $\hat{y}_{sc}$ axis.

Noting that the rotary joint permits rotation about $\hat{y}_{sc}$, we may define the orientation of the payload relative to the spacecraft as $\mathcal{O}_{pl/sc} = \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix}$ and it follows that $\mathcal{O}_{pl/N} = \mathcal{O}_{pl/sc}\mathcal{O}_{sc/N}$. We refer to γ as the payload angle and $\gamma = \bar{\gamma} + \acute{\gamma}$ where $\bar{\gamma}$ and $\acute{\gamma}$ are the gimbal and torsional joint angles respectively. Essentially, the motor articulating joint G contributes stiffness and damping terms to the moment induced on the payload. These terms are functions of $\acute{\gamma} << 1$ and $\dot{\acute{\gamma}} << 1$ respectively. We may define M_G as the net torque induced on the payload by the spacecraft about the $\hat{y}_{sc}$ axis. M_G is thus a virtual input from which one may construct the true input: $M_{G'} = M_G + K\acute{\gamma} + C\dot{\acute{\gamma}}$ where K and C are the stiffness and damping coefficients respectively.

The system has three additional inputs aside from M_G, namely the three orthogonal components of a moment $\vec{M}_{sc}$ induced on the spacecraft by its ACS. The spacecraft and payload have moments of inertia I_{sc} and I_{pl} respectively as well as masses m_{sc} and m_{pl}. The vectors $\vec{r}_{G/sc}$ and $\vec{r}_{G/pl}$ define the positions of the rotary joint relative to each body's center of mass. These vectors have constant coefficients when resolved in the appropriate body frames: $\vec{r}_{G/sc}|_{sc} = b_1\hat{x}_{sc} + b_2\hat{y}_{sc} + b_3\hat{z}_{sc}$ and $\vec{r}_{G/pl}|_{pl} = a_1\hat{x}_{pl} + a_2\hat{y}_{pl} + a_3\hat{z}_{pl}$. The system model as described above is illustrated in Fig. 1.

3 Equations of Motion

We derive our equations of motion following Kane's method as described in [1]. The angular velocity of the payload relative to the inertial frame $\vec{\omega}_{pl/N}$ may be written as the following sum of angular velocities:

$$\vec{\omega}_{pl/N} = \vec{\omega}_{pl/sc} + \vec{\omega}_{sc/N}, \tag{2}$$

where $\vec{\omega}_{pl/sc}$ and $\vec{\omega}_{sc/N}$ are the angular velocities of the payload relative to the spacecraft and the spacecraft relative to the inertial frame respectively. $\vec{\omega}_{pl/sc}$, when resolved in F_{pl}, can be parameterized as follows:

$$\vec{\omega}_{pl/sc}|_{pl} = \Gamma\sigma, \tag{3}$$

where σ is the generalized speed of the rotary joint. For 2-DOF or 3-DOF joints, σ may be defined as the *Euler angle rate vector* (i.e., the time deriviatives of the sequential rotations defining F_{pl} relative to F_{sc}) and Γ is the kinematic differential equation matrix associated with the rotation sequence (see tabulations on left column of Appendix II in [3]). Note that the example Γ defined in (23) of [1] contains errors which may provide a source of confusion to a first time reader. For our 1-DOF joint, $\Gamma = [0\ 1\ 0]^T$, $\dot{\Gamma} = [0\ 0\ 0]^T$, and $\sigma = \dot{\gamma}$.

Upon describing our payload's angular velocity, we now define the linear velocity of its center of mass:

$$\vec{v}_{pl/N} = \vec{v}_{sc/N} + \vec{\omega}_{sc/N} \times \vec{r}_{G/sc} - \vec{\omega}_{pl/N} \times \vec{r}_{G/pl}. \tag{4}$$

Combining (2-4), we can rewrite the angular and linear velocities of our system:

$$\begin{bmatrix} \vec{\omega}_{sc/N}|_{sc} \\ \vec{\omega}_{pl/N}|_{pl} \end{bmatrix} = \underbrace{\begin{bmatrix} U_{3\times3} & 0_{3\times1} & 0_{3\times3} \\ \mathcal{O}_{pl/sc} & \Gamma & 0_{3\times3} \end{bmatrix}}_{\Omega} \begin{bmatrix} \vec{\omega}_{sc/N}|_{sc} \\ \sigma \\ \vec{v}_{sc/N}|_{N} \end{bmatrix}, \tag{5}$$

$$\begin{bmatrix} \vec{v}_{sc/N}|_{N} \\ \vec{v}_{pl/N}|_{N} \end{bmatrix} = \underbrace{\begin{bmatrix} 0_{3\times3} & 0_{3\times1} & U_{3\times3} \\ \left(\vec{r}_{G/pl}|_{N} - \vec{r}_{G/sc}|_{N}\right)^{\times} \mathcal{O}_{N/sc} & \vec{r}_{G/pl}|_{N}^{\times} \mathcal{O}_{N/pl}\Gamma & U_{3\times3} \end{bmatrix}}_{V} \begin{bmatrix} \vec{\omega}_{sc/N}|_{sc} \\ \sigma \\ \vec{v}_{sc/N}|_{N} \end{bmatrix}, \tag{6}$$

in terms of our state vector $u = \begin{bmatrix} \vec{\omega}_{sc/N}|_{sc} & \sigma & \vec{v}_{sc/N}|_{N} \end{bmatrix}^T$ where $U_{3\times3}$ denotes a 3×3 identity matrix and $0_{3\times3}$ similarly denotes a matrix of zeros. Note that a $\times$ superscript in (6) refers to the skew-symmetric cross product matrix and that Ω and V are referred to by Stoneking as the *partial velocity matrices*. Taking an inertial frame derivative of (2) and resolving in F_{pl} yields:

$$\vec{\alpha}_{pl/N}|_{pl} = \vec{\alpha}_{sc/N}|_{pl} + \left(\Gamma\dot{\sigma} + \dot{\Gamma}\sigma + \vec{\omega}_{pl/N}|_{pl} \times \Gamma\sigma\right), \tag{7}$$

while the inertial frame derivative of (4) is:

$$\vec{a}_{pl/N} = \vec{a}_{sc/N} + \vec{w}_{sc/N} \times \left(\vec{\omega}_{sc/N} \times \vec{r}_{G/sc}\right) + \vec{\alpha}_{sc/N} \times \vec{r}_{G/sc} - \vec{\omega}_{pl/N} \times \left(\vec{\omega}_{pl/N} \times \vec{r}_{G/pl}\right) - \vec{\alpha}_{pl/N} \times \vec{r}_{G/pl}, \tag{8}$$

which when resolved in F_N yields:

$$\begin{aligned} \vec{a}_{pl/N}|_{N} = {} & \left(\vec{a}_{sc/N} + \vec{w}_{sc/N} \times \left(\vec{\omega}_{sc/N} \times \vec{r}_{G/sc}\right) + \vec{\alpha}_{sc/N} \times \vec{r}_{G/sc}\right)\Big|_{N} - \left(\vec{\omega}_{sc/N}|_{N} + \mathcal{O}_{N/pl}\Gamma\sigma\right) \times \\ & \left(\left(\vec{\omega}_{sc/N}|_{N} + \mathcal{O}_{N/pl}\Gamma\sigma\right) \times \vec{r}_{G/pl}|_{N}\right) - \left(\vec{\alpha}_{sc/N}|_{N} + \mathcal{O}_{N/pl}\left(\Gamma\dot{\sigma} + \dot{\Gamma}\sigma + \vec{\omega}_{pl/N}|_{pl} \times \Gamma\sigma\right)\right) \times \vec{r}_{G/pl}|_{N}. \end{aligned} \tag{9}$$

It should be noted that (7) and (8) contain terms that are not functions of $\dot{u}$, these are referred to as the *remainder accelerations*:

$$\vec{\alpha}^{r}_{pl/N}|_{pl} = \dot{\Gamma}\sigma + \vec{\omega}_{pl/N}|_{pl} \times \Gamma\sigma, \tag{10}$$

$$\vec{a}^{r}_{pl/N} = \vec{w}_{sc/N} \times \left(\vec{\omega}_{sc/N} \times \vec{r}_{G/sc}\right) - \vec{\omega}_{pl/N} \times \left(\vec{\omega}_{pl/N} \times \vec{r}_{G/pl}\right) - \vec{\alpha}^{r}_{pl/N} \times \vec{r}_{G/pl}. \tag{11}$$

We can now write our system accelerations compactly:

$$\begin{bmatrix}\vec{\alpha}_{sc/N}|_{sc}\\ \vec{\alpha}_{pl/N}|_{pl}\end{bmatrix} = \Omega\dot{u} + \begin{bmatrix}0_{3\times1}\\ \vec{\alpha}^r_{pl/N}|_{pl}\end{bmatrix}, \tag{12}$$

$$\begin{bmatrix}\vec{a}_{sc/N}|_{N}\\ \vec{a}_{pl/N}|_{N}\end{bmatrix} = V\dot{u} + \begin{bmatrix}0_{3\times1}\\ \vec{a}^r_{pl/N}|_{N}\end{bmatrix}. \tag{13}$$

From [1], we have that Kane's equation for our system may be written in matrix form:

$$\Omega^T\left(\begin{bmatrix}\vec{M}_{sc}|_{sc}\\ \vec{M}_G|_{pl}\end{bmatrix} - \begin{bmatrix}I_{sc} & 0_{3\times3}\\ 0_{3\times3} & I_{pl}\end{bmatrix}\begin{bmatrix}\vec{\alpha}_{sc/N}|_{sc}\\ \vec{\alpha}_{pl/N}|_{pl}\end{bmatrix} - \begin{bmatrix}\vec{\omega}_{sc/N}|_{sc}\times I_{sc}\vec{\omega}_{sc/N}|_{sc}\\ \vec{\omega}_{pl/N}|_{pl}\times I_{pl}\vec{\omega}_{pl/N}|_{pl}\end{bmatrix}\right) + V^T\left(\begin{bmatrix}\vec{F}_{sc}|_{N}\\ \vec{F}_{pl}|_{N}\end{bmatrix} - \begin{bmatrix}m_{sc}U_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & m_{pl}U_{3\times3}\end{bmatrix}\begin{bmatrix}\vec{a}_{sc/N}|_{N}\\ \vec{a}_{pl/NN}\end{bmatrix}\right) = 0, \tag{14}$$

where one may recall that $\vec{M}_G|_{pl} = M_G\hat{y}_{pl} = M_G\hat{y}_{sc}$. For our system, we assume that our external forces obey $\vec{F}_{sc} = \vec{F}_{pl} = 0$. Substituting (12-13) into (14), we have:

$$\underbrace{\left(\Omega^T\begin{bmatrix}I_{sc} & 0_{3\times3}\\ 0_{3\times3} & I_{pl}\end{bmatrix}\Omega + V^T\begin{bmatrix}m_{sc}U_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & m_{pl}U_{3\times3}\end{bmatrix}V\right)}_{L}\dot{u} = \Omega^T\left(\begin{bmatrix}\vec{M}_{sc}|_{sc}\\ \vec{M}_G|_{pl}\end{bmatrix} - \begin{bmatrix}I_{sc} & 0_{3\times3}\\ 0_{3\times3} & I_{pl}\end{bmatrix}\begin{bmatrix}0_{3\times1}\\ \vec{\alpha}^r_{pl/N}|_{pl}\end{bmatrix} - \begin{bmatrix}\vec{\omega}_{sc/N}|_{sc}\times I_{sc}\vec{\omega}_{sc/N}|_{sc}\\ \vec{\omega}_{pl/N}|_{pl}\times I_{pl}\vec{\omega}_{pl/N}|_{pl}\end{bmatrix}\right) - V^T\begin{bmatrix}m_{sc}U_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & m_{pl}U_{3\times3}\end{bmatrix}\begin{bmatrix}0_{3\times1}\\ \vec{a}^r_{pl/N}|_{N}\end{bmatrix}, \tag{15}$$

where we have introduced the definition L to simplify notation. Moving forward, we shall refer to the entire right hand side of (15) as P.

For our system, L is a 7×7 invertible matrix. Thus, it is possible to compute an explicit form of L^{-1} (though incredibly computationally expensive). We have constructed a generic 7×7 matrix in MATLAB composed of 49 symbolic variables "A_1_1", "A_1_2" etc. generated using the "syms" command. An analytic expression of the inverse of this matrix has been saved to the file "SevenBySevenInverse.mat" which is available upon request. Thus, one may compute the desired analytic expression for L^{-1} by substituting the algebraic expressions defining each element of L for the variables "A_1_1", "A_1_2" etc. using the symbolic substitution "subs" command in MATLAB. We have found that this approach saves considerable computational time over attempting to invert L directly and can be computed in less than a day by a modern desktop computer.

Noting that the rotational dynamics decouple from the translational dynamics in this parameterization, we can discard the $\vec{v}_{sc/N}|_N$ component of our state vector u and redefine our system in terms of fully explicit (i.e., $\dot{x} = f(x)$) nonlinear rotational equations of motion:

$$\dot{\phi} = \vec{\omega}_{sc/N}|_{sc}\cdot\begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\end{bmatrix}^T, \tag{16}$$

$$\dot{\theta} = \vec{\omega}_{sc/N}|_{sc}\cdot\begin{bmatrix}0 & \cos\phi & -\sin\phi\end{bmatrix}^T, \tag{17}$$

$$\dot{\psi} = \vec{\omega}_{sc/N}|_{sc}\cdot\begin{bmatrix}0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}^T, \tag{18}$$

$$\dot{\gamma} = \sigma, \tag{19}$$

$$\dot{\vec{\omega}}_{sc/N}|_{sc} = \begin{bmatrix}U_{3\times3} & 0_{3\times4}\end{bmatrix}\left(L^{-1}P\right), \tag{20}$$

$$\dot{\sigma} = \begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}\left(L^{-1}P\right). \tag{21}$$

4 Linearization and Control Design

We can now linearize our equations of motion (16-21) about an equilibrium state. We assume that this state consists of an arbitrary spacecraft orientation and gimbal angle with zero associated angular velocities, i.e. $x_d = [\phi_d \ \ \theta_d \ \ \psi_d \ \ \gamma_d \ \ 0 \ \ 0 \ \ 0 \ \ 0]^T$. Let us define our system input vector $\bar{u} = \left[\vec{M}_{sc}|_{sc} \ \ M_G\right]^T$ Noting our implicitly defined rotational dynamics state variable x, the linearized dynamics are:

$$\dot{x} = \left.\frac{\partial f}{\partial x}\right|_{\substack{x = x_d, \\ \bar{u} = 0}} (x - x_d) + \left.\frac{\partial f}{\partial \bar{u}}\right|_{\substack{x = x_d, \\ \bar{u} = 0}} \bar{u}, \tag{22}$$

where f is the vector concatenated from the right hand sides of (16-21). In the sequel, we shall refer to the two matrices of partial derivatives in (22) as A and B respectively.

As alluded to above, our nonlinear equations of motion are intractable to deal with by hand. Thus we resort to symbolic differentiation in MATLAB via the "diff" command. For details on implementation, please see lines 96-141 in the script "ThreeDeeModeling_Kane_3DCOM.m" included in the Appendix.

Having generated our A and B matrices. It was straightforward to design a stabilizing feedback law of the form $\bar{u} = -Kx$ in MATLAB using the "lqr" command. To inform the design, we analyzed the impulse responses in MATLAB that resulted from various choices of the Q matrix in the LQR cost function. We iterated through progressively larger and larger Q matrices while keeping R fixed at identity until the settling time had reached an acceptable value of 842 seconds for $Q = 50,000U$. These iterations are presented in Figs. 2-7. The settling time approximately halves with each order of magnitude increase in Q. Note that although this trend continues beyond $Q = 50,000U$, we selected $Q = 50,000U$ as the weight matrix for our initial nonlinear simulation out of a concern that we would eventually saturate the actuators. We also tried increasing the penalties on orientation error above those of rate error as presented in Figs. 8-9; however, this did not make a noticeable difference.

Following our impulse response figures, we have also included a direct comparison of this controller's performance against the slew controller designed by Alejandro Hernandez as a part of his former NASA Goddard internship work on LUVOIR. For this simulation, the initial spacecraft orientation is $[\phi \ \ \theta \ \ \psi] = [2^\circ \ \ 2^\circ \ \ 88^\circ]$ and the desired spacecraft orientation is $[\phi_d \ \ \theta_d \ \ \psi_d] = [0^\circ \ \ 0^\circ \ \ 90^\circ]$. The initial gimbal angle is 1.54 radians with desired angle of $\pi/2$ radians. The direct performance comparisons are presented in Figs. 10-15. The takeaway points are that Alejandro's PID slew controller stabilizes faster (though at the expense of tremendous initial torque inputs) while the LQR controlled spacecraft follows smoother trajectories with more conservative actuation requirements. The LQR controller is generally free of high frequency oscillations with the exception of those of the gimbal controller as presented in Fig. 13.

Moving forward, we plan to evaluate the performance of this controller in more complex spacecraft and payload models. The nonlinear simulations presented in this report were generated using a multi-rigid body model for LUVOIR. In order to accurately predict the performance of this control strategy, we will need to simulate a flexible body spacecraft and flexible body payload subject to disturbances. If the LQR strategy does not perform well when we add these layers of complexity to the model, then perhaps we ought to consider a robust nonlinear control strategy e.g., *sliding mode control* or Lyapunov based control.

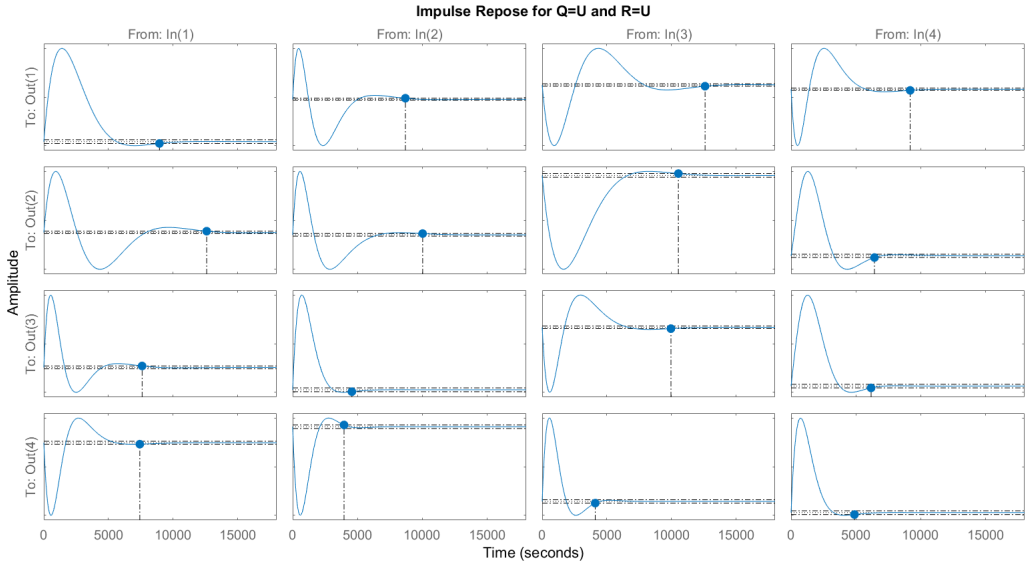

Figure 2: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & U_{4\times4} \end{bmatrix}$. The maximum settling time of 12600 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

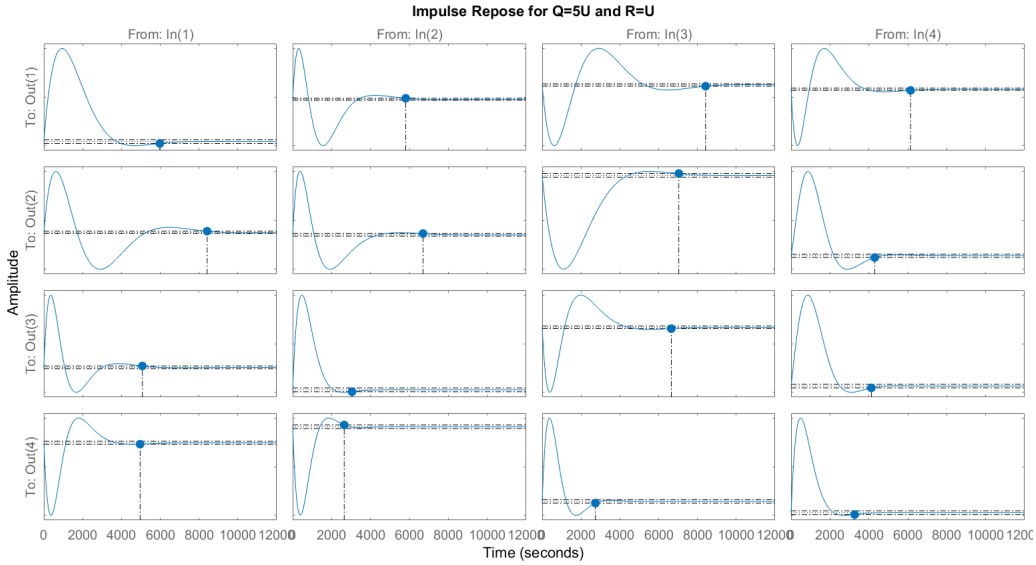

Figure 3: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 5U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 5U_{4\times4} \end{bmatrix}$. The maximum settling time of 8420 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

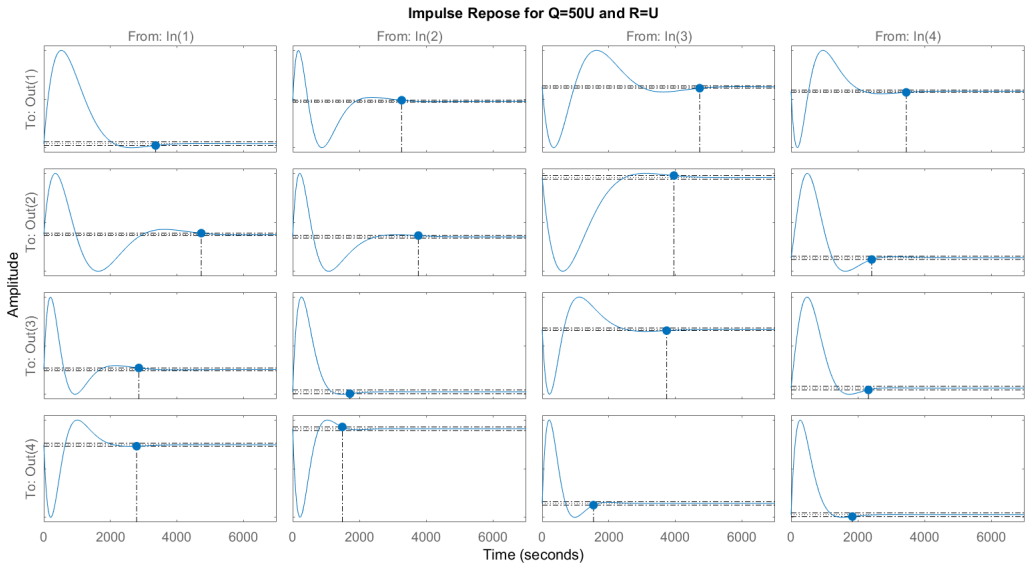

Figure 4: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 50U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 50U_{4\times4} \end{bmatrix}$. The maximum settling time of 4730 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

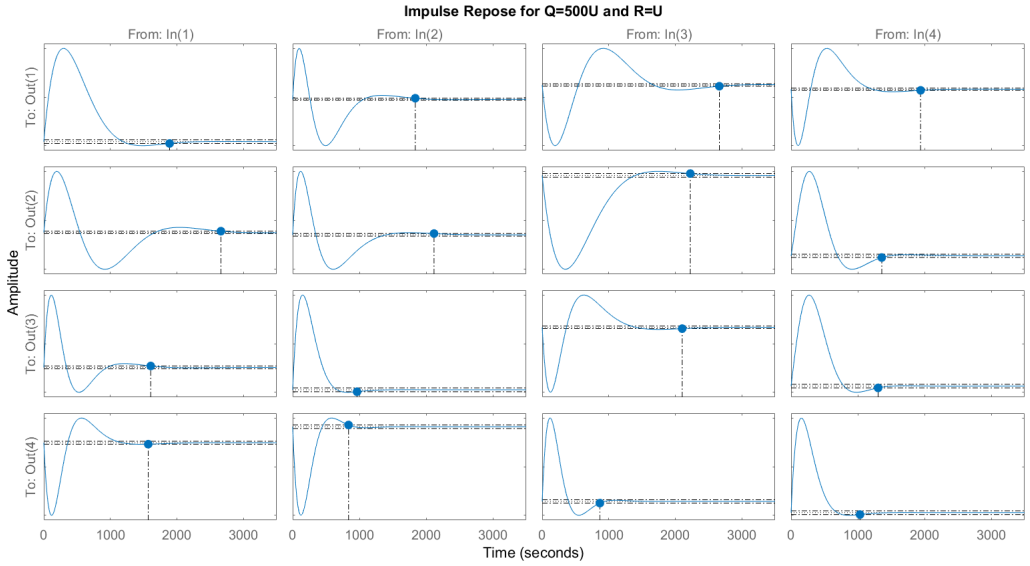

Figure 5: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 500U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 500U_{4\times4} \end{bmatrix}$. The maximum settling time of 2660 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

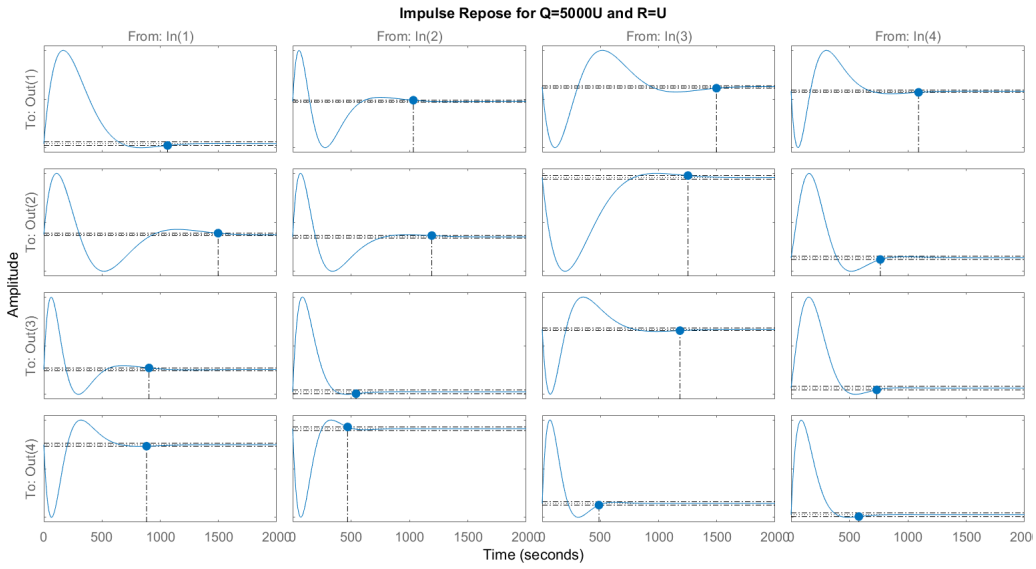

Figure 6: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 5000U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 5000U_{4\times4} \end{bmatrix}$. The maximum settling time of 1500 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

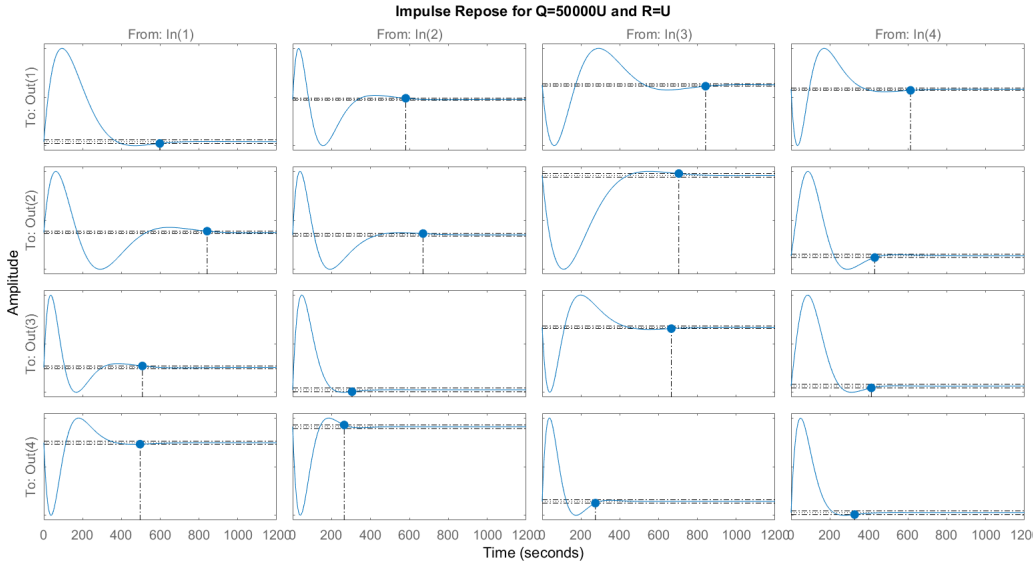

Figure 7: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 50000U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 50000U_{4\times4} \end{bmatrix}$. The maximum settling time of 842 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

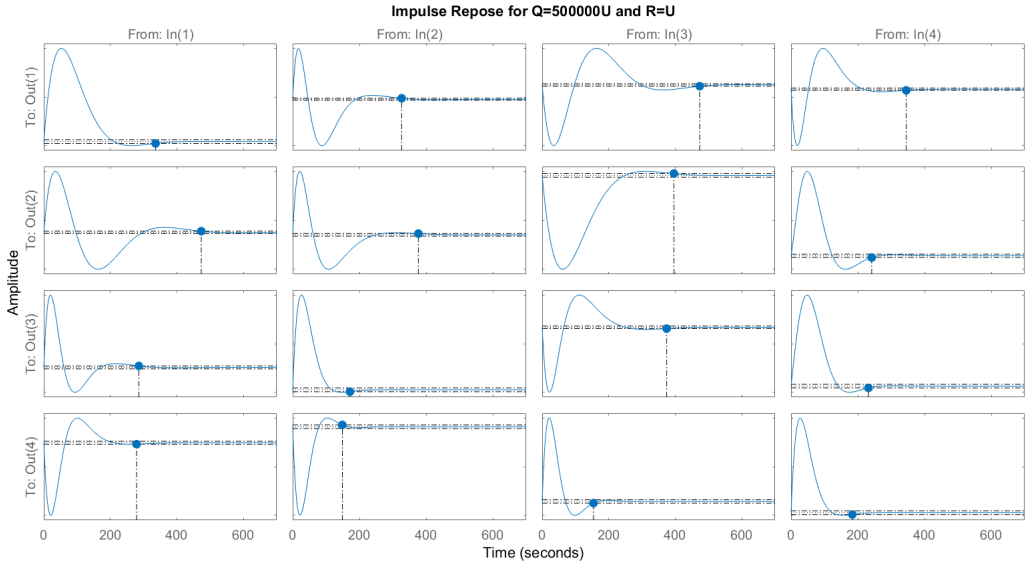

Figure 8: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 500000U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 500000U_{4\times4} \end{bmatrix}$. The maximum settling time of 473 seconds is associated with state ϕ subject to a disturbance impulse torque along $\hat{z}_{sc}$.

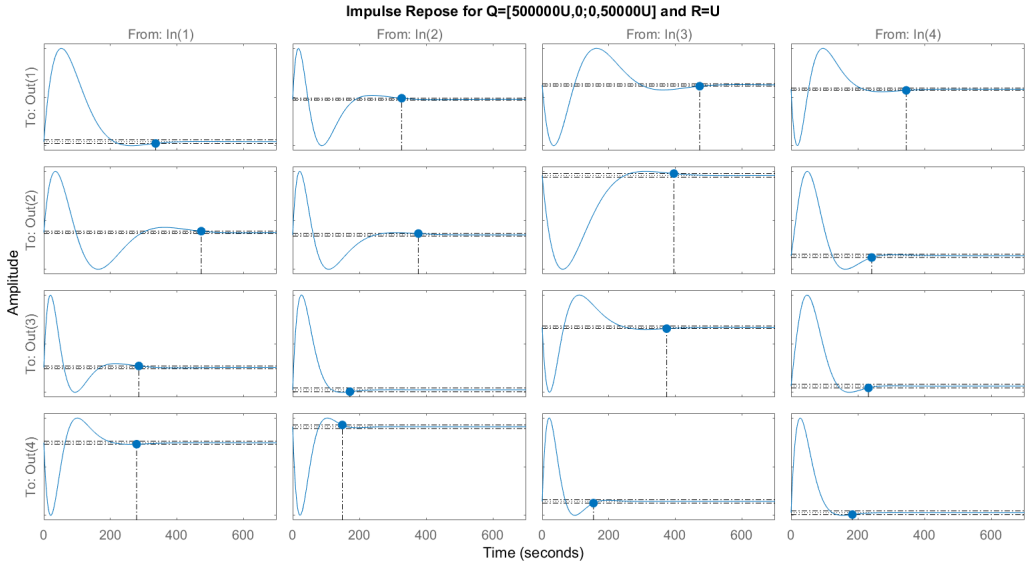

Figure 9: The normalized impulse responses of the first four states of the closed loop linearized dynamics are illustrated for $Q = \begin{bmatrix} 500000U_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 50000U_{4\times4} \end{bmatrix}$. The maximum settling time remains unchanged from Fig. 8 despite the differing weight on angular velocity.

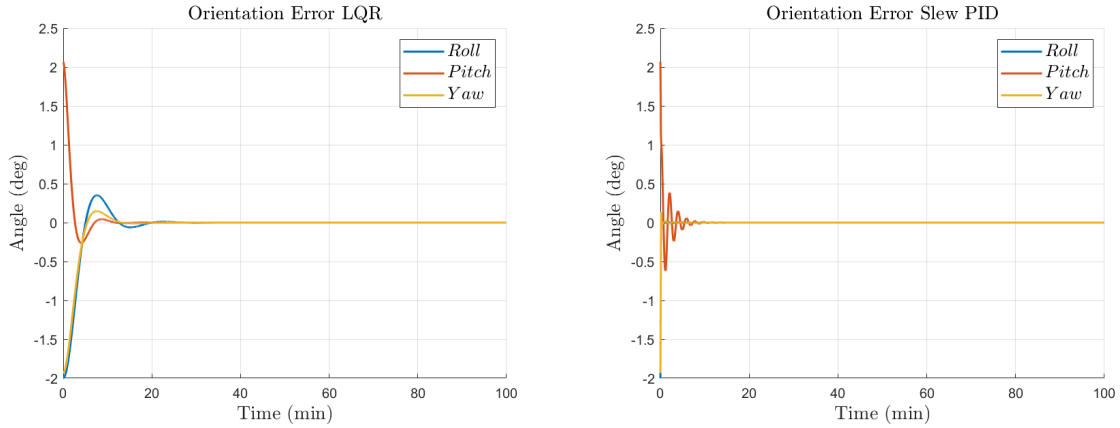

Figure 10: The baseline slew controller stabilizes the orientation errors faster, though at the expensive of initial high frequency oscillations.

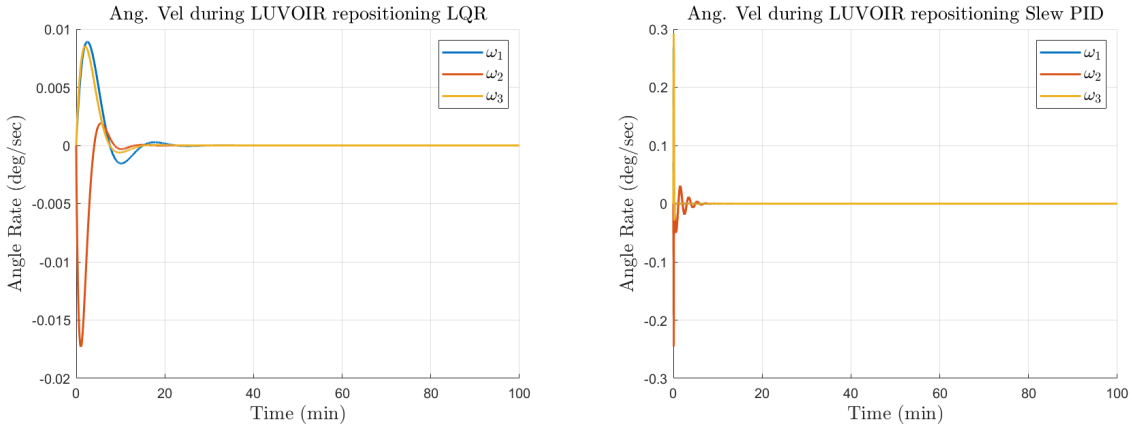

Figure 11: Peak angular velocities are orders of magnitude smaller in the LQR controller.

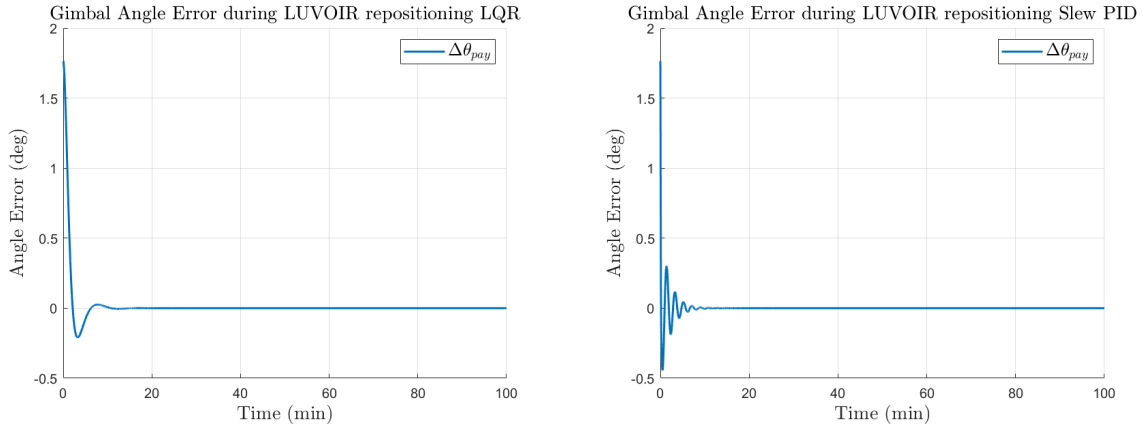

Figure 12: High frequency oscillations present in the PID slew controller are not present in the LQR based method.

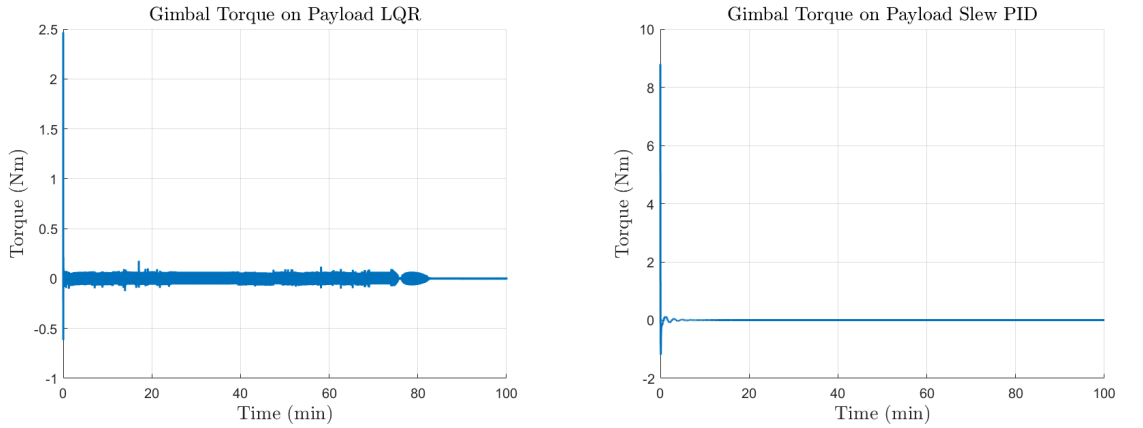

Figure 13: The PID slew controller sends higher peak torque commands to the gimbal. The LQR controller has higher frequency oscillations as the gimbal error stabilizes.

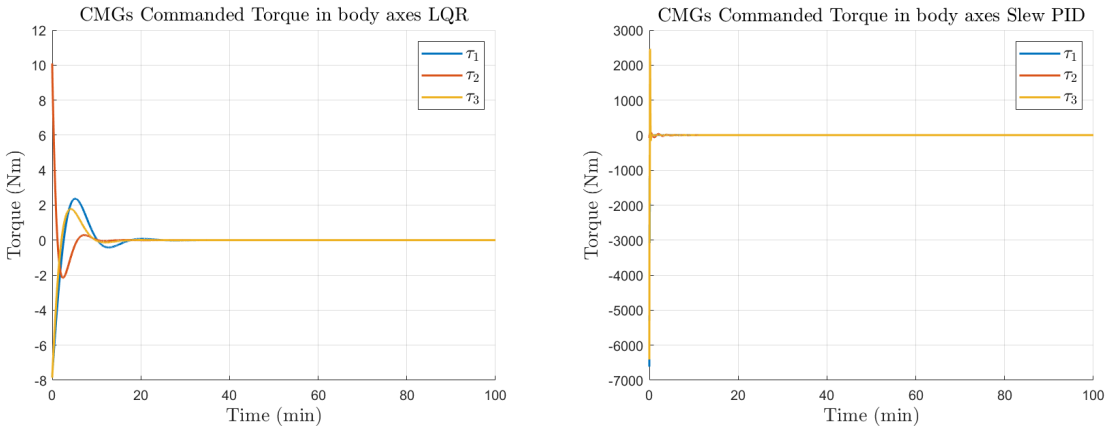

Figure 14: Peak CMG torques are orders of magnitude higher in the PID controller.

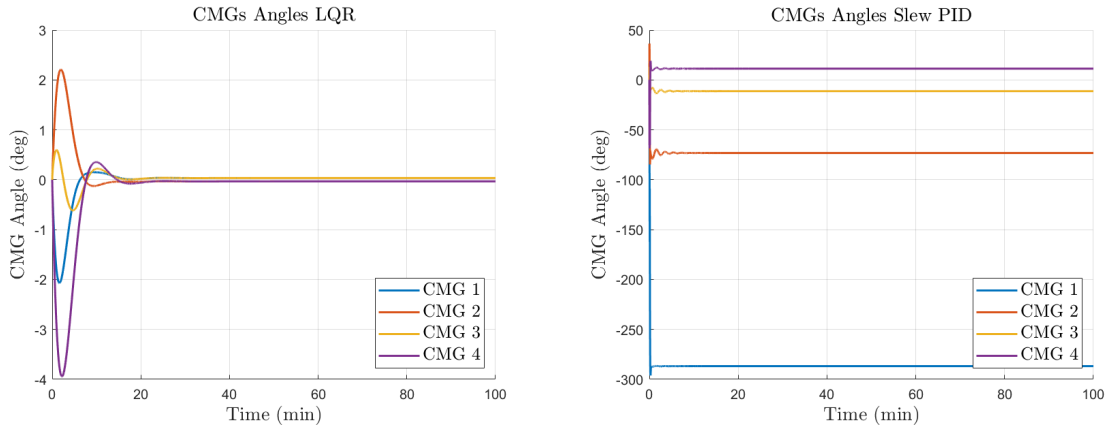

Figure 15: CMG angles are nearly zero in steady state for the LQR controller.

5 Appendix

ThreeDeeModeling_Kane_3DCOM.m:

```
%Let's take Kane's approach as in Stoneking 2013
clear
%I will use the standard 3-2-1 Euler angles psi theta phi with gimbal
    angle
%gamma about spacecraft Y axis.
syms psi theta phi gamma gammadot a_1 a_2 a_3 b_1 b_2 b_3 w_sc1 w_sc2
    w_sc3;
Gamma=[0;1;0];
Gammadot=[0;0;0];
C_21=[cos(gamma),0,-sin(gamma);
        0, 1, 0;
        sin(gamma),0,cos(gamma)];
Omega=[eye(3),zeros(3,4);
        C_21,Gamma,zeros(3,3)];
%position vectors expressed in inertial frames
O_N_SC=transpose([1,0,0;0,cos(phi),sin(phi);0,-sin(phi),cos(phi)]*[cos(
    theta),0,-sin(theta);0,1,0;sin(theta),0,cos(theta)]*[cos(psi),sin(
    psi),0;-sin(psi),cos(psi),0;0,0,1]);
O_N_PL=transpose([cos(gamma),0,-sin(gamma);0,1,0;sin(gamma),0,cos(gamma
    )]*[1,0,0;0,cos(phi),sin(phi);0,-sin(phi),cos(phi)]*[cos(theta),0,-
    sin(theta);0,1,0;sin(theta),0,cos(theta)]*[cos(psi),sin(psi),0;-sin(
    psi),cos(psi),0;0,0,1]);
O_PL_SC=[cos(gamma),0,-sin(gamma);0,1,0;sin(gamma),0,cos(gamma)];
r_21=O_N_PL*[a_1;a_2;a_3];
r_11=O_N_SC*[b_1;b_2;b_3];
beta_2=r_21-r_11;
beta_2_skew=[0,-beta_2(3),beta_2(2);
                beta_2(3),0,-beta_2(1);
                -beta_2(2),beta_2(1),0];
r_21_skew=[0,-r_21(3),r_21(2);
                r_21(3),0,-r_21(1);
                -r_21(2),r_21(1),0];
V=[zeros(3,4),eye(3);
     beta_2_skew*O_N_SC,r_21_skew*O_N_PL*Gamma,eye(3)];
wo=[0;gammadot;0]+O_PL_SC*[w_sc1;w_sc2;w_sc3];
alpha_N_o_r=cross(wo,Gamma*gammadot);
a_N_o_r=cross(O_N_SC*[w_sc1;w_sc2;w_sc3],cross(O_N_SC*[w_sc1;w_sc2;
    w_sc3],r_11))-cross(O_N_PL*wo,cross(O_N_PL*wo,r_21))-cross(O_N_PL*
    alpha_N_o_r,r_21);
%Control inputs are T_sc1, T_sc2, T_sc3, and T_pl2
syms T_sc1 T_sc2 T_sc3 T_pl2
T_squig=[T_sc1;T_sc2;T_sc3;0;T_pl2;0];
%Independent moment of inertia elements are:
```

```
35 syms I_sc_1_1 I_sc_1_2 I_sc_1_3 I_sc_2_2 I_sc_2_3 I_sc_3_3
36 syms I_pl_1_1 I_pl_1_2 I_pl_1_3 I_pl_2_2 I_pl_2_3 I_pl_3_3
37 syms m_sc m_pl
38 I_brack=[I_sc_1_1 , I_sc_1_2 , I_sc_1_3 , 0, 0, 0;
39           I_sc_1_2 , I_sc_2_2 , I_sc_2_3 , 0, 0, 0;
40           I_sc_1_3 , I_sc_2_3 , I_sc_3_3 , 0, 0, 0;
41           0, 0, 0, I_pl_1_1 , I_pl_1_2 , I_pl_1_3 ;
42           0, 0, 0, I_pl_1_2 , I_pl_2_2 , I_pl_2_3 ;
43           0, 0, 0, I_pl_1_3 , I_pl_2_3 , I_pl_3_3 ];
44 m_brack=[m_sc*eye(3) ,zeros(3,3);
45           zeros(3,3) ,m_pl*eye(3) ];
46 alpha_r_squig =[0;0;0;alpha_N_o_r ];
47 wcrossH_squig=[cross([w_sc1;w_sc2;w_sc3] ,[I_sc_1_1 , I_sc_1_2 , I_sc_1_3 ;
       I_sc_1_2 , I_sc_2_2 , I_sc_2_3 ;I_sc_1_3 , I_sc_2_3 , I_sc_3_3 ]*[w_sc1;
       w_sc2;w_sc3]);
48                 cross(wo,[I_pl_1_1 , I_pl_1_2 , I_pl_1_3 ;I_pl_1_2 ,
                       I_pl_2_2 , I_pl_2_3 ;I_pl_1_3 , I_pl_2_3 , I_pl_3_3 ]*wo)
                       ];
49 F_squig=zeros(6,1);
50 a_r_squig =[0;0;0;a_N_o_r ];
51 %u_dot=(transpose(Omega)*I_brack*Omega+transpose(V)*m_brack*V)\(
       transpose(Omega)*(T_squig-I_brack*alpha_r_squig-wcrossH_squig)+
       transpose(V)*(F_squig-m_brack*a_r_squig))
52
53 %%Line 51 is taking an eternity to run so let's substitute values into
       a symbolic representation
54 %%of a 7x7 inverted matrix below. This symbolic file is
       SevenbySevenInverse.mat
55 %We will skip running lines 58-81 and just load the result in the
       following
56 %line.
57
58 LargeMat=transpose(Omega)*I_brack*Omega+transpose(V)*m_brack*V; %
       LargeMat
59 %is the matrix we seek to invert.
60 load('SevenbySevenInverse.mat')
61 for j=1:7
62     for k=1:7
63    syms(strcat(strcat(strcat('A_',num2str(j)),'_'),num2str(k)));
64     end
65 end
66 LargeMatInv=sym(zeros(7,7));
67
68 for j=1:7
69 for k=1:7
70 tic
```

```
LargeMatInv(j,k)=subs(PLEASESAVE(j,k),A_1_1,LargeMat(1,1));
for jj=1:7
    for kk=1:7
   A_jj_kk=strcat(strcat(strcat('A_',num2str(jj)),'_'),num2str(kk));
        LargeMatInv(j,k)=subs(LargeMatInv(j,k),A_jj_kk,LargeMat(jj,kk))
            ;
    end
end
toc
end
end
%Okay That worked.
%load('LargeMatInv.mat')
u_dot=LargeMatInv*(transpose(Omega)*(T_squig-I_brack*alpha_r_squig-
    wcrossH_squig)+transpose(V)*(F_squig-m_brack*a_r_squig));
%Okay Here is the desired uncoupled EOMS
%u_new=[phi;theta;psi;gamma;w_sc1;w_sc2;w_sc3;gammadot];
u_new_dot(1)=w_sc1+sin(phi)*tan(theta)*w_sc2+cos(phi)*tan(theta)*w_sc3;
u_new_dot(2)=cos(phi)*w_sc2-sin(phi)*w_sc3;
u_new_dot(3)=sin(phi)*sec(theta)*w_sc2+cos(phi)*sec(theta)*w_sc3;
u_new_dot(4)=gammadot;
u_new_dot(5)=u_dot(1);
u_new_dot(6)=u_dot(2);
u_new_dot(7)=u_dot(3);
u_new_dot(8)=u_dot(4);
%Now Let us linearize about our operating point [phi_d; theta_d; psi_d;
%gamma_d; 0; 0; 0; 0];
syms phi_d theta_d psi_d gamma_d
for j=1:8
tic
A(j,1)=diff(u_new_dot(j),phi);
A(j,1)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,1),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,2)=diff(u_new_dot(j),theta);
A(j,2)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,2),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,3)=diff(u_new_dot(j),psi);
A(j,3)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,3),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,4)=diff(u_new_dot(j),gamma);
```

```
A(j,4)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,4),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,5)=diff(u_new_dot(j),w_sc1);
A(j,5)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,5),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,6)=diff(u_new_dot(j),w_sc2);
A(j,6)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,6),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,7)=diff(u_new_dot(j),w_sc3);
A(j,7)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,7),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

A(j,8)=diff(u_new_dot(j),gammadot);
A(j,8)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(A(j
    ,8),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);
toc
end
%Guess what T_pl_2 is a VIRTUAL INPUT. I'm working now in terms of
    T_pl2
% DON'T FORGET THAT True Input M_G follows state feedback law
% M_G=T_pl2+k*gamma+c*gammadot
for j=1:8
tic
B(j,1)=diff(u_new_dot(j),T_sc1);
B(j,1)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(B(j
    ,1),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

B(j,2)=diff(u_new_dot(j),T_sc2);
B(j,2)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(B(j
    ,2),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

B(j,3)=diff(u_new_dot(j),T_sc3);
B(j,3)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(B(j
    ,3),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);

```

```
B(j,4)=diff(u_new_dot(j),T_pl2);
B(j,4)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(B(j
    ,4),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),
    w_sc2,0),w_sc3,0),gammadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0);
toc
end
save('State_Space_3DCOM.mat','A','B','-v7.3')
fprintf('Save Complete\n')
A=simplify(A);
B=simplify(B);
```

References

[1] Eric Stoneking. Implementation of Kane's Method for a Spacecraft Composed of Multiple Rigid Bodies. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4649, 2013.

[2] Dennis Bernstein. *GEOMETRY, KINEMATICS, STATICS, AND DYNAMICS*. 2012. University of Michigan Graduate Dynamics Course Notes, http://ruina.tam.cornell.edu/Courses/ME4730 Fall 2013/BernsteinDynamicsVDec282012.pdf.

[3] Thomas R. Kane, Peter W. Likins, and David A. Levinson. *Spacecraft dynamics*. McGraw-Hill Book Co., 1983.