# Summary of Model-based Attitude Control of LUVOIR in Modular Dynamic Analysis (MDA) Simulink Environment

William Bentz*, Lia Sacks**

June 2018

This document overviews the work I completed in the second half of my summer 2018 internship experience in Code 591 at NASA Goddard Space Flight Center. Please see the former memorandum [1] for additional context on the project. The take away point is that LUVOIR A has been modeled as three rigid bodies linked by 1-DOF rotary joints. An LQR attitude controller was designed for precision pointing of the spacecraft with the design requirement that steady state oscillations have amplitude less than a miliarcsecond. The performance of the algorithm was tested in the Modular Dynamic Analysis Simulink library developed by J. Roger Chen at NASA Goddard. While the initial simulations were of rigid bodies, subsequent simulations included flexible body modes on all three bodies of the model. The simulated structural deformations initially destabilized the LQR controller thus requiring a redesign of the K gain matrix. The final simulation demonstrated a stabilizing feedback gain with miliarcsecond precision within 400 minutes. This run used 20 modes on the spacecraft, 9 on the bus, and 100 on the payload. Future recommended work includes the development of a Vibration Isolation and Precision Pointing System (VIPPS) Simulink model. Armed with this model, the system should be modeled as four bodies whose associated flexible files must be generated from the spacecraft through Gimbal 1, Gimbal 1 through Gimbal 2, Gimbal 2 through the VIPPS interface, and the VIPPS interface through the payload.

## I. Modeling and Equations of Motion

We have added an additional level of complexity to the LUVOIR model which is now composed of three rigid bodies connected in serial by two rotary joints as illustrated in Fig. 1. The bodies are referred to as the spacecraft, tower, and payload respectively. Note that in the MDA codes the tower is often referred to interchangeably as the boom. The derivation of the equations of motion (EOMs) for this model follows the process outlined in [2] which we described in detail in our former memorandum for the two body system [1].

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

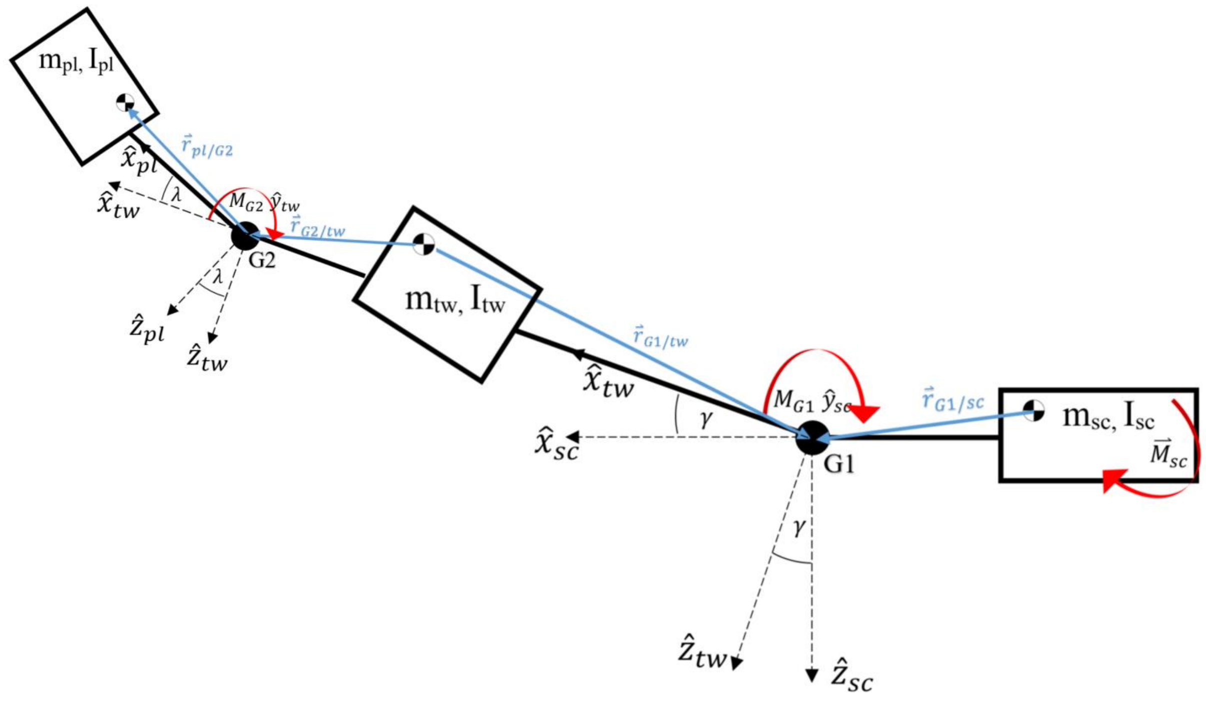
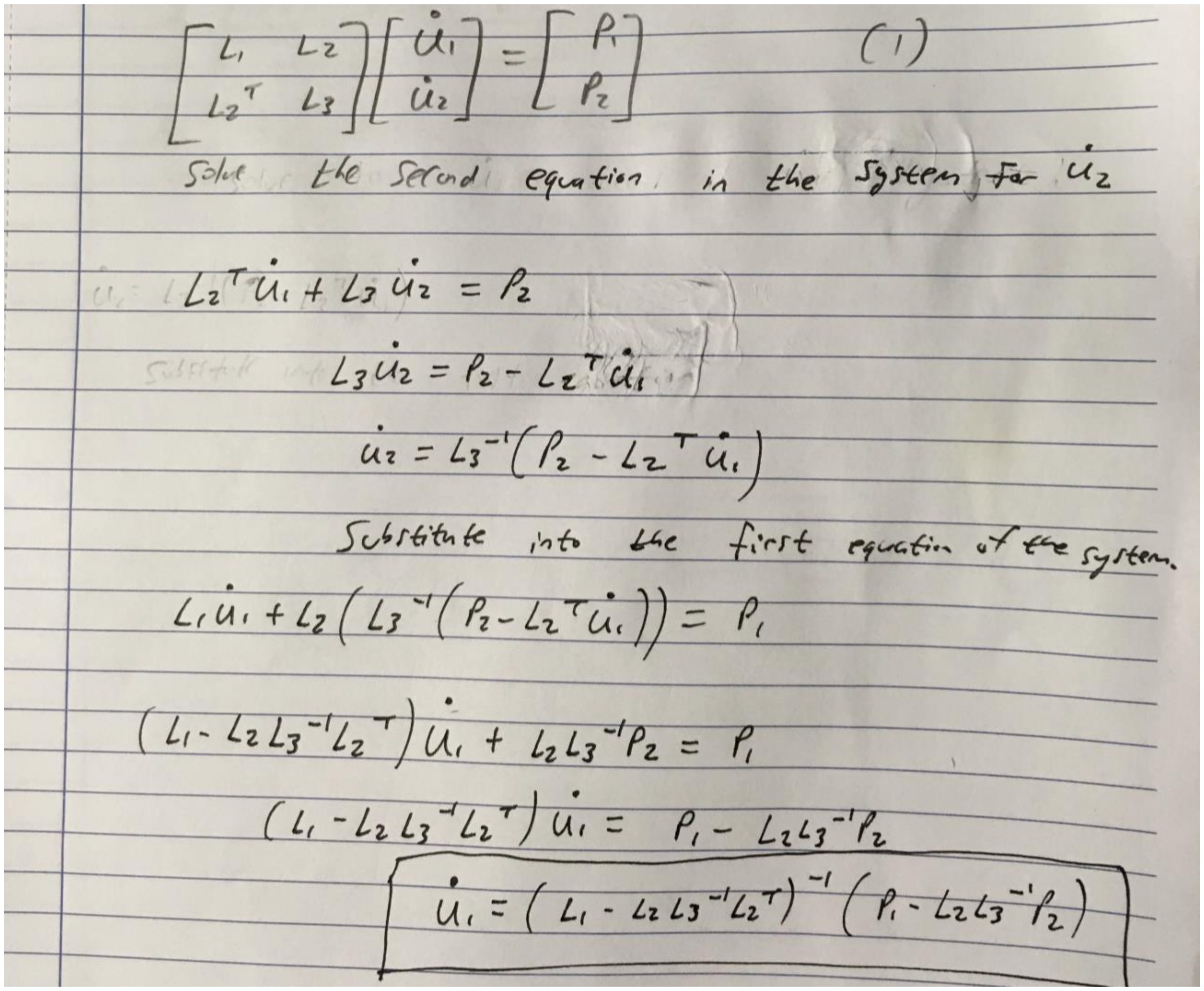


Fig. 1. LUVOIR A modeled as three rigid bodies connected by two 1-DOF rotary joints.

We used symbolic manipulation and symbolic differentiation to derive the three body EOMs and linearize about arbitrary desired 3-2-1 Euler angles ($\psi - \theta - \phi$), tower gimbal angle $\gamma$, and payload gimbal angle $\lambda$. The linearization point for Euler angle rates and gimbal angle rates is zero. The MATLAB script that performs these computations is titled "ThreeDeeModeling_Kane_Two_1DOF_joints_InertiaFast_CountTorq.m" and is provided in Appendix A. Note in this document that I define all orientation matrices (AKA Direction Cosine Matrices) in a manner consistent with [1] where "O_BO_SC" is equivalent to $\mathcal{O}_{bo/sc}$. Note that "sc" is the spacecraft frame, "bo" is the boom frame (AKA tower frame), and "pl" is the payload frame referred to as serial linked bodies 1, 2 and 3 respectively in Stoneking's notation [2].

The script is more or less an extension of [1] until we reach the definition for "T_squig" on line 64 of the .m file (line numbers differ in the Appendix due to page width constraints). Note here the terms "T_sc2-T_bo2" and "T_bo2-T_pl2" associated with Y-axis torques on the spacecraft and the boom. Stoneking's formulation considers only external torques that are imposed on the rigid bodies while our input terms "T_bo2" and "T_pl2" are actually internal torques applied on an outer body by an inner body at the two respective gimbals. Thus when defining the net torques applied to the spacecraft and boom one must subtract the reaction torques from the outer bodies within Stoneking's equation. This correction is not present in [1] as it was discovered in July.

"LargeMat" defined on line 90 of the .m file refers to $(\Omega^T[I]\Omega + V^T[m]V)$ in equation (41) of [2]. At this point, it is too computationally complex to invert the 8x8 symbolic $(\Omega^T[I]\Omega + V^T[m]V)$ expression in order to obtain an explicit $\dot{u}$. Thankfully, we're only really interested in the first five states of $u$, i.e., $u_1 = [\vec{\omega}_{sc/N}\ \dot{\gamma}\ \dot{\lambda}]$. Let us refer to $(\Omega^T[I]\Omega + V^T[m]V)$ as L, and the right hand side of (41) in [2] as P. Lines 90 through 119 essentially perform the following operations:



$$\begin{bmatrix} L_1 & L_2 \\ L_2^T & L_3 \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \quad (1)$$

Solve the second equation in the system for $\dot{u}_2$

$$L_2^T \dot{u}_1 + L_3 \dot{u}_2 = P_2$$

$$L_3 \dot{u}_2 = P_2 - L_2^T \dot{u}_1$$

$$\dot{u}_2 = L_3^{-1}(P_2 - L_2^T \dot{u}_1)$$

Substitute into the first equation of the system.

$$L_1 \dot{u}_1 + L_2\left(L_3^{-1}\left(P_2 - L_2^T \dot{u}_1\right)\right) = P_1$$

$$\left(L_1 - L_2 L_3^{-1} L_2^T\right) \dot{u}_1 + L_2 L_3^{-1} P_2 = P_1$$

$$\left(L_1 - L_2 L_3^{-1} L_2^T\right) \dot{u}_1 = P_1 - L_2 L_3^{-1} P_2$$

$$\boxed{\dot{u}_1 = \left(L_1 - L_2 L_3^{-1} L_2^T\right)^{-1} \left(P_1 - L_2 L_3^{-1} P_2\right)}$$

Lines 123-132 of the script combine our expressions for $\dot{u}_1$, the second order rotational dynamics, with the standard kinematic differential equations for Euler angles (see Appendix II of [3]) to arrive at a set of ten explicit nonlinear differential equations fully characterizing the system's rotation dynamics. The remainder of the script performs symbolic differentiation in order to generate the linearized EOMS which it saves to a .mat file in the form of an "A" and "B" matrix.

## II. Rotating Parameters into the MDA Frame

The script "Compute_MDA_parameters_Two1DOFGimbals.m" rotates parameters from the CS 0 Vertex Frame as provided by structural team images into the MDA frames defined in this document. This is included in Appendix B. Note that in MDA, the spacecraft bus has its origin at the spacecraft center of mass while the outer bodies each have their origins at the joint linking them to their inner body, i.e., gimbal 1 and gimbal 2 respectively. Note in this .m file that SC, BO, and PL refer to center of mass positions and LV refers to the launch vehicle interface. The values generated by this script should be copy/pasted into the following scripts where appropriate: “bus_body_params_Two1DOFJoints.m”, “boom_body_params.m”, and “pay_body_params.m” in the Parameters folder and “FlexGenerator_Bentz.m” in the FlexData folder. The values generated by “Compute_MDA_parameters_Two1DOFGimbals.m” were also used to populate the powerpoint slides “LUVOIR Mass Properties_Bentz.ppt” which contains an additional diagram illustrating the MDA reference frame on a realistic structural model.

Note that the three #_body_params.m files also contain the variables defining initial conditions for orientation. The reference trajectory that you wish for the system to track is held in the guidance folder. For my purposes, I was interested in precision pointing and thus my reference trajectories (“gimbal_traj_Will.mat” and “SC_traj_Will.mat”) are defined as constant vectors.

## III. Notes on Flexible Body Generation

The flexible body files are generated by the script “FlexGenerator_Bentz.m” in the FlexData folder. This file reads in a set of Matlab files which Christine Collins of code 542 has provided. These are described in the FlexData folder in the excel spreadsheets. My flexible body files, like Alex’s before me, only used the eigenvalue frequencies from the third column as xx_oeigen.mat as these have the greatest effect upon the modal shapes. Roger’s MDA flex body powerpoint indicates that Ei and Gi also effect the shape as well and I believe that these are columns 1-3 and 4-6 of the xx_rbmphi.mat files respectively (The matrix seems to be transposed compared to what Christine lists in the excel sheet). As I stated, my results do not contain the effects of Ei and Gi; however, I have created an untested version of my generation script “FlexGenerator_Bentz_Beta.m” that reads in these components as well. I have not simulated the performance of the system with these additional files and I assume that more tuning of the gains would be required after the inclusion of these two matrices.

**BEFORE RUNNING FlexGenerator_Bentz.m, ensure that you have set the desired number of modes for each of the three bodies (n=#; on lines 11, 117, and 169) and that you also set the correct desired number of modes in bus_flexible_params.m, boom_flexible_params.m and pay_flexible_params.m**

In defining the parameters, rso for all three bodies (according to Roger) should be the undeformed center of mass position relative to the reference point of the rigid body frame. As the frame for the spacecraft has its origin at the COM, and both outerbody frames are at the gimbals, my three definitions for rso as [0,0,0]’, [6.937…,0.000362…,0.004103..]', and [4.193781…, 0.0141…, 2.734634…]' are consistent with Compute_MDA_parameters_Two1DOFGimbals.m. However, some confusions seemed to arise with respect to the definition of Io. Roger felt that Io for the two outer bodies should be the moment of inertia of that body about the reference point of that frame (i.e., gimbal 1 and gimbal 2). However, I found that the system response seemed totally erratic here even for very small controller gains, e.g., Q=I. Gimbal angle 1 would begin to correct in the wrong direction before very quickly exploding and causing the simulation to crash due to integration error. When I commented out both parallel axis theorem lines which invoke the “transfer.m” script, it allowed for the system to be stabilized and the performance for very small gains now seemed to be consistent with the rigid body model.

Looking under the mask of the flexible body blocks in MDA, it seems that there would be no way of telling the block that Io is about any other point aside from the center of mass anyway. It’s not like we are setting any sort of “free-free” or “clamped-free” flag in the MDA block to tell it which point to sum the moments about. Furthermore, the rso vector is still defined as center of mass position relative to reference point. Therefore, I am assuming that Iso should be about the center of mass for all of three rigid body files as this resulted in simulations that did not crash and ran in a manner consistent with their rigid body counterparts for very small gains. My opinion on Iso is in direct contradiction to Roger’s and thus this should be explored more.

Perhaps the culprit could also lie in a misunderstanding on the definition of rso (maybe these should all 0??). Roger was on travel for a good portion of the latter half of my internship and thus I was unable to really resolve this with him. A future intern will need to screen share the FlexGenerator_Bentz script with Roger and really nail down every instance of rso and Iso and come to a 100% agreement on exactly what each definition should be and then go over exactly where Iso is used under the mask of the Flexible blody MDA block to ensure that his present understanding of the flex body Iso definition is consistent with when he designed the block.

An additional issue lies in the files that were provided by Christine for the simulation. The three files are defined from the bus through gimbal 1, gimbal 1 through VIPPS, and VIPPS through payload. This is incorrect because in MDA we define the second and third bodies as interfacing at gimbal 2 so I’m not entirely sure what is happening with the mode shapes due to this inconsistency. I had discussed with Christine about how we should address this and she stated that she cannot generate a file that passes through the VIPPS as the bodies are physically separated there in her current model. I believe that these inconsistencies can be resolved through a 4 body model which includes an interface at the VIPPS and I briefly outline this in the Future Work section.

## IV. Simulation Initiation and LQR Design

After ensuring that the appropriate numerical values for the mass and inertia properties have been set (see section II), and that the number of modes in “bus_flexible_params.m”, “boom_flexible_params.m”, and “pay_flexible_params.m” are correct (relative to their associated flexible files), one should enter the “Initiation and Parameters” directory and run the script “init_sim_LUVOIR_Two1DOFJoints.m” which loads the LUVOIR parameters into memory. Note that the files names of reference trajectories are also specified in this script.

This initiation script launches many functions in sequence. One of which, “bus_body_params_Two1DOFJoints.m” also contains the code that generates the LQR controller. It will load the symbolic A and B matrices described at the end of section I and substitute numerical values in for all of the symbolic ones. Note that lines 104-108 of “bus_body_params_Two1DOFJoints.m” defines the desired orientation of the spacecraft in terms of the final values associated with the reference trajectory files (which are constant anyway). This means that the entire definition of the reference trajectory file is more or less a remnant of the old simulation which was defined for a time varying slew trajectory. The reference trajectory file is still required to allow for the simulation to compile; however, the values of that file are really only used here in defining the linearization point of the controller. Line 114 is where the actual LQR control gains are specified. It currently has R as identity and $Q = \begin{bmatrix} 1000\, I_{8\times 8} & 0_{8\times 2} \\ 0_{8\times 8} & 2000\, I_{2\times 2} \end{bmatrix}$. This worked well for the flex SC, flex Boom, Flex Pay model.

## V. Simulation Results and Design Iterations

As described in [1], my initial method was to plot the impulse responses of the linearized system to inform the choice of Q matrix. However, it soon became apparent that the introduction of flexible modes was destabilizing to the system and caused the Euler angles to explode as illustrated in Fig. 2.

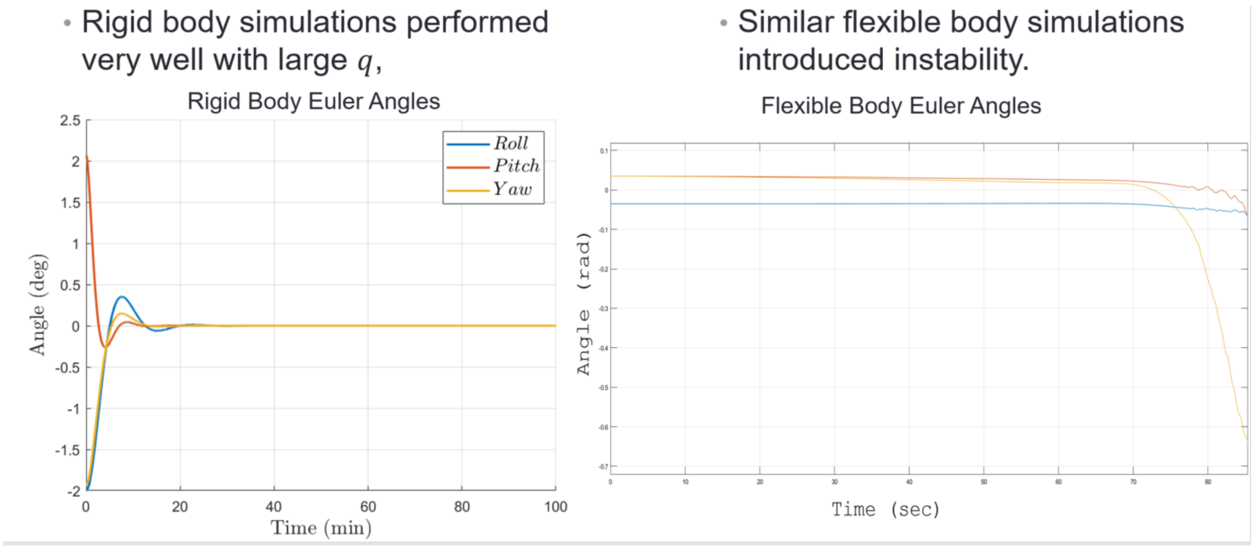


Fig. 2. Values for Q which are stabilizing in the rigid body model may result in an unstable flexible body system.

The instability seems to have grown out from a rapid oscillation in gimbal angle 1 which would gradually build in amplitude over the first few seconds of simulation before completely destabilizing (see Fig. 3). My approach was to reduce Q by orders of magnitude (down from $Q = 5 \times 10^6 I$) and then to place additional penalty on the gimbal angle rates (states 9 and 10 of the system). This stabilized the system; however, it also increased the settling time from hundreds of seconds to hundreds of minutes. Achieving miliarcsecond precision in less than 400 seconds was deemed “still reasonable” for a system with potentially multi-week science observations.

## VI. Notes on the MDA Implementation

"LUVOIR_MDA_FlexSC_FlexPay_FlexBoom_WillBentz_V4.slx" is the final and deliverable MDA model. My LQR controller is implemented in two locations:

1) LUVOIR_MDA_FlexSC_FlexPay_FlexBoom_WillBentz_V4->Controllers->CMG Steering Control->Torque control->Control

2) LUVOIR_MDA_FlexSC_FlexPay_FlexBoom_WillBentz_V4->Controllers->Payload Controller-> Boom Gimbal Control

Location 1) generates $\vec{M}_{sc}$ while Location 2) generates both $M_{G1}$ and $M_{G2}$. Note that Location 1) includes a gain of -1 before output. This sign change is included because the controller is outputting a desired CMG torque and thus a reaction torque is exerted on the spacecraft bus. Note that Location 2) includes additional deviations from our ideal model. The gimbal is modeled as a harmonic drive that accepts a rotor joint input and contains stiffness and damping parameters that are functions of an internal torsional deflection $\bar{\gamma}$ (on the order of $10^{-8}$ radians). Therefore, the rotor joint input is computed by feeding forward a restoring torque of the form $\tau_r = \kappa\bar{\gamma} + C\dot{\bar{\gamma}}$ into our control signal $M_G$ before dividing by the gear ratio of the harmonic drive. Location 1) and Location 2) are illustrated in Fig. 3 and Fig. 4.

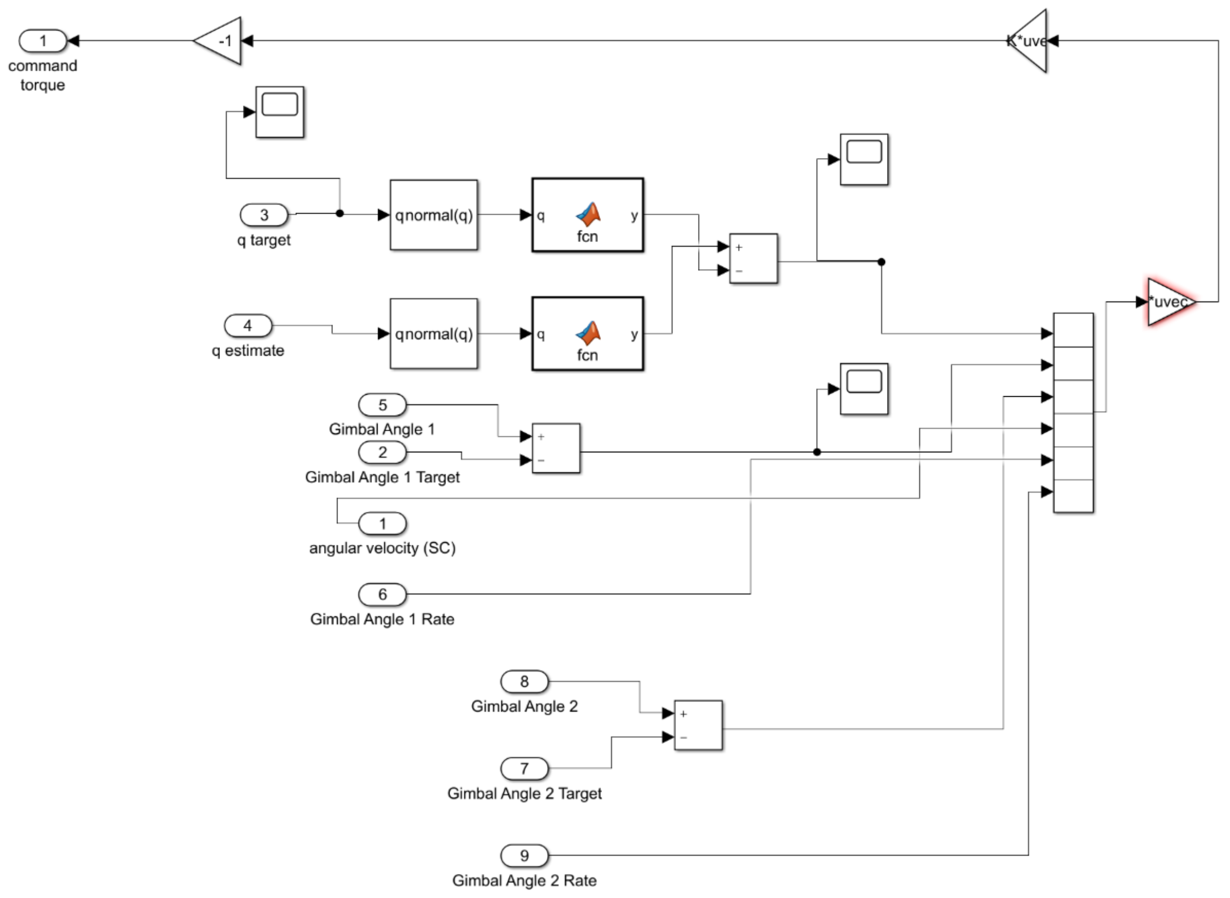


Fig. 3. The CMG commanded torque is illustrated above. Note that the custom MATLAB function block converts the quaternions to Euler angles. Also note the quaternion normalization

blocks. One should always normalize a quaternion before attempting to convert it to Euler angles as even slight deviations from norm can result in substantial errors in the Euler angles.

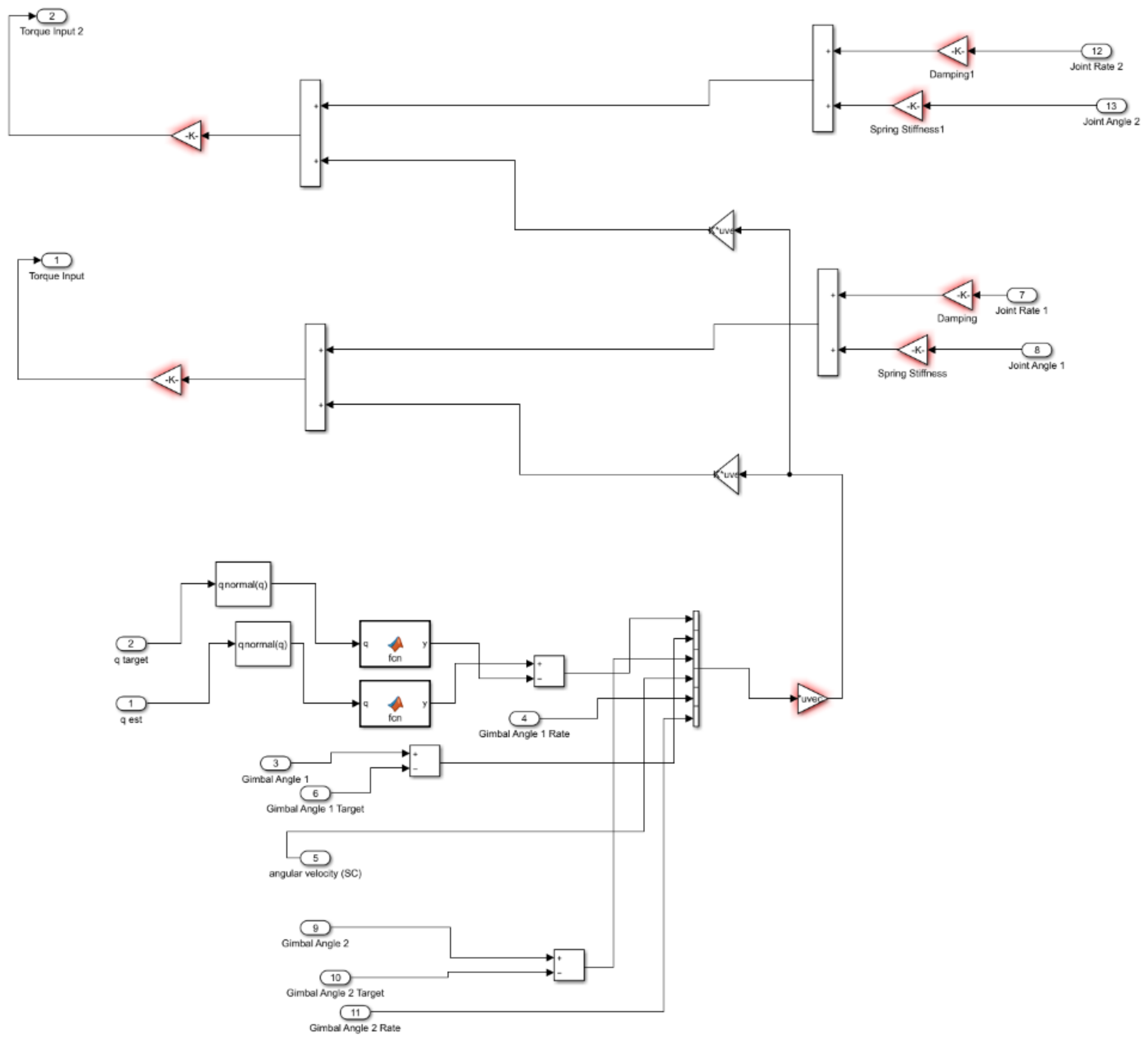


Fig. 4. The gimbal torque commands are illustrated above. Note in the upper right that the reaction torques associated with stiffness and damping in the harmonic drive are fed forward into the control signal.

## VII. Concluding Notes and Future Work

I should briefly mention that this is not the only simulation model I completed this summer. I also tested LQR on two rigid bodies sharing a single joint (detailed in [1]), and then on two bodies where the first was flexible, and then on three rigid bodies, etc. Thus many iterations of my models exist in the models folder (all with my name in the file names). However, I only detailed the final one as it is the culmination of the work and running the prior models are logical extensions. When running a two body problem, please use init_sim_LUVOIR instead of init_sim_LUVOIR_Two1DofJoints,  and also note that in this case the changing the mass and inertia properties of the system (and the control gains) can be done through “bus_body_params.m”.

I recommend the development of a Vibration Isolation and Precision Pointing System (VIPPS) Simulink model. Armed with this model, the system should be modeled as four bodies whose associated flexible files must be generated from the spacecraft through Gimbal 1, Gimbal 1 through Gimbal 2, Gimbal 2 through the VIPPS interface, and the VIPPS interface through the payload. It is also worth exploring the use of robust control strategies such as sliding mode control or H-infinity which may be better suited for the flexible body perturbations. At the very least, it may be worth designing a controller based on these techniques and plugging it into my existing Simulink model the evaluate the performance relative to LQR.

Additionally, a question was raised in my intern presentation as to how realistic it is to measure a torsional deflection on the order of 10^-8 radians in order to feed forward reaction torques of the gimbals into the control signal. This is an open question that ought to be discussed with the mechanical or systems team. In general, state estimation is an entirely open question on this project and the MDA model ought to be further augmented to include sensor models.

## Appendix:

### A. ThreeDeeModeling_Kane_Two_1DOF_joints_InertiaFast_CountTorq.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 and gimbal angle lambda about the boom Y axis..
syms psi theta phi gamma lambda gammadot lambdadot a_1 a_2 a_3 b_1 b_2 b_3 c_1 c_2 c_3 d_1 d_2
d_3 w_sc1 w_sc2 w_sc3;
Gamma_1=[0;1;0];
Gammadot_1=[0;0;0];
Gamma_2=[0;1;0];
Gammadot_2=[0;0;0];
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_BO_SC=[cos(gamma),0,-sin(gamma);0,1,0;sin(gamma),0,cos(gamma)];
O_PL_SC=[cos(gamma+lambda),0,-sin(gamma+lambda);0,1,0;sin(gamma+lambda),0,cos(gamma+lambda)];
O_PL_BO=[cos(lambda),0,-sin(lambda);0,1,0;sin(lambda),0,cos(lambda)];
Omega=[eye(3),zeros(3,5);
       O_BO_SC,Gamma_1,zeros(3,4);
       O_PL_SC,O_PL_BO*Gamma_1,Gamma_2,zeros(3,3)];


%position vectors expressed in inertial frames

O_N_PL=transpose([cos(gamma+lambda),0,-
sin(gamma+lambda);0,1,0;sin(gamma+lambda),0,cos(gamma+lambda)]*[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_BO=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]);
r_11=O_N_SC*[a_1;a_2;a_3];
r_21=O_N_BO*[b_1;b_2;b_3];
r_22=O_N_BO*[c_1;c_2;c_3];
r_32=O_N_PL*[d_1;d_2;d_3];
r_31=r_32-r_22+r_21;

beta_2=r_21-r_11;
```

```
beta_3=r_31-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];
beta_3_skew=[0,-beta_3(3),beta_3(2);
             beta_3(3),0,-beta_3(1);
             -beta_3(2),beta_3(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];
r_31_skew=[0,-r_31(3),r_31(2);
             r_31(3),0,-r_31(1);
             -r_31(2),r_31(1),0];
r_32_skew=[0,-r_32(3),r_32(2);
             r_32(3),0,-r_32(1);
             -r_32(2),r_32(1),0];

V=[zeros(3,3),zeros(3,1), zeros(3,1), eye(3);
   beta_2_skew*O_N_SC,r_21_skew*O_N_BO*Gamma_1,zeros(3,1),eye(3);
   beta_3_skew*O_N_SC,r_31_skew*O_N_BO*Gamma_1,r_32_skew*O_N_PL*Gamma_2,eye(3)];


wo_BO=[0;gammadot;0]+O_BO_SC*[w_sc1;w_sc2;w_sc3];
alpha_N_BO_r=cross(wo_BO,Gamma_1*gammadot);
a_N_BO_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_BO*wo_BO,cross(O_N_BO*wo_BO,r_21))-cross(O_N_BO*alpha_N_BO_r,r_21);

wo_PL=O_PL_BO*wo_BO+[0;lambdadot;0];
alpha_N_PL_r=O_PL_BO*alpha_N_BO_r+cross(wo_PL,Gamma_2*lambdadot);
a_N_PL_r=a_N_BO_r+cross(O_N_BO*wo_BO,cross(O_N_BO*wo_BO,r_22))+cross(O_N_BO*alpha_N_BO_r,r_22)-
cross(O_N_PL*wo_PL,cross(O_N_PL*wo_PL,r_32))-cross(O_N_PL*alpha_N_PL_r,r_32);
%Control inputs are T_sc1, T_sc2, T_sc3, T_bo2 and T_pl2
syms T_sc1 T_sc2 T_sc3 T_bo2 T_pl2
T_squig=[T_sc1;T_sc2-T_bo2;T_sc3;0;T_bo2-T_pl2;0;0;T_pl2;0];
%Independent moment of inertia elements are:
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
syms I_bo_1_1 I_bo_1_2 I_bo_1_3 I_bo_2_2 I_bo_2_3 I_bo_3_3
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
syms m_sc m_bo m_pl
I_brack=[I_sc_1_1, I_sc_1_2, I_sc_1_3, 0, 0, 0, 0, 0, 0;
         I_sc_1_2, I_sc_2_2, I_sc_2_3, 0, 0, 0, 0, 0, 0;
         I_sc_1_3, I_sc_2_3, I_sc_3_3, 0, 0, 0, 0, 0, 0;
         0, 0, 0, I_bo_1_1, I_bo_1_2, I_bo_1_3, 0, 0, 0;
         0, 0, 0, I_bo_1_2, I_bo_2_2, I_bo_2_3, 0, 0, 0;
         0, 0, 0, I_bo_1_3, I_bo_2_3, I_bo_3_3, 0, 0, 0;
         0, 0, 0, 0, 0, 0, I_pl_1_1, I_pl_1_2, I_pl_1_3;
         0, 0, 0, 0, 0, 0, I_pl_1_2, I_pl_2_2, I_pl_2_3;
         0, 0, 0, 0, 0, 0, I_pl_1_3, I_pl_2_3, I_pl_3_3];
m_brack=[m_sc*eye(3),zeros(3,3), zeros(3,3);
         zeros(3,3),m_bo*eye(3), zeros(3,3);
         zeros(3,3),zeros(3,3),m_pl*eye(3)];
alpha_r_squig=[0;0;0;alpha_N_BO_r;alpha_N_PL_r];

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]);
               cross(wo_BO,[I_bo_1_1, I_bo_1_2, I_bo_1_3;I_bo_1_2, I_bo_2_2, I_bo_2_3;I_bo_1_3,
I_bo_2_3, I_bo_3_3]*wo_BO);
               cross(wo_PL,[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_PL)];
F_squig=zeros(9,1);
a_r_squig=[0;0;0;a_N_BO_r;a_N_PL_r];

LargeMat=transpose(Omega)*I_brack*Omega+transpose(V)*m_brack*V; %LargeMat
P=(transpose(Omega)*(T_squig-I_brack*alpha_r_squig-wcrossH_squig)+transpose(V)*(F_squig-
m_brack*a_r_squig));
L1=simplify(LargeMat(1:5,1:5));
L2=simplify(LargeMat(1:5,6:8));
L3=simplify(LargeMat(6:8,6:8));
```

```
P1=simplify(P(1:5));
P2=simplify(P(6:8));
fprintf('L1 through p2 are simplified, computing udot\n')
L3_inv=inv(L3);
FiveByFive_Inv=sym('A_%d_%d',[5,5]);
FiveByFive_Inv=inv(FiveByFive_Inv);

Term_1=sym(zeros(5,5));
Term_1_preinvert=L1-L2*L3_inv*transpose(L2);
syms A_1_1
for j=1:5
for k=1:5
tic
Term_1(j,k)=subs(FiveByFive_Inv(j,k),A_1_1,Term_1_preinvert(1,1));
for jj=1:5
    for kk=1:5
   A_jj_kk=strcat(strcat(strcat('A_',num2str(jj)),'_'),num2str(kk));
        Term_1(j,k)=subs(Term_1(j,k),A_jj_kk,Term_1_preinvert(jj,kk));
    end
end
toc
end
end
u_dot=Term_1*(P1-L2*L3_inv*P2);
fprintf('udot computed!\n')

%Okay Here is the desired uncoupled EOMS
%u_new=[phi;theta;psi;gamma;lambda;w_sc1;w_sc2;w_sc3;gammadot;lambdadot];
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)=lambdadot;
u_new_dot(6)=u_dot(1);
u_new_dot(7)=u_dot(2);
u_new_dot(8)=u_dot(3);
u_new_dot(9)=u_dot(4);
u_new_dot(10)=u_dot(5);
%Now Let us linearize about our operating point [phi_d; theta_d; psi_d;
%gamma_d; lambda_d; 0; 0; 0; 0; 0];
syms phi_d theta_d psi_d gamma_d lambda_d

fprintf('Okay Time to Run the differentiation loop\n')
parfor j=1:10
tic
A_1(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),phi),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_2(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),theta),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_3(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),psi),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_4(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),gamma),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_5(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),lambda),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),w_sc2,0),w_sc3,0),gammado
t,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_6(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),w_sc1),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),lambda,lambda_d),lambdadot,0),T_bo2,0);
```

```
A_7(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),w_sc2),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_8(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),w_sc3),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_9(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),gammadot),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),w_sc1,0),w_sc2,0),w_sc3,0),gamma
dot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0),lambda,lambda_d),lambdadot,0),T_bo2,0);

A_10(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot
(j),lambdadot),phi,phi_d),theta,theta_d),psi,psi_d),gamma,gamma_d),lambda,lambda_d),w_sc1,0),w_sc
2,0),w_sc3,0),gammadot,0),lambdadot,0),T_sc1,0),T_sc2,0),T_sc3,0),T_pl2,0),T_bo2,0);

B_1(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),T_sc1),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

B_2(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),T_sc2),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

B_3(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),T_sc3),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

B_4(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),T_bo2),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),lambda,lambda_d),lambdadot,0),T_bo2,0);

B_5(j)=subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(subs(diff(u_new_dot(
j),T_pl2),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),lambda,lambda_d),lambdadot,0),T_bo2,0);
toc
end
fprintf('Finished Parallel Stuff\n')
for j=1:10
    tic
    A(j,1)=A_1(j);
    A(j,2)=A_2(j);
    A(j,3)=A_3(j);
    A(j,4)=A_4(j);
    A(j,5)=A_5(j);
    A(j,6)=A_6(j);
    A(j,7)=A_7(j);
    A(j,8)=A_8(j);
    A(j,9)=A_9(j);
    A(j,10)=A_10(j);
    B(j,1)=B_1(j);
    B(j,2)=B_2(j);
    B(j,3)=B_3(j);
    B(j,4)=B_4(j);
    B(j,5)=B_5(j);
    toc
end

save('State_Space_TWO_1DOF_Joints_CountTorq.mat','A','B','-v7.3')
fprintf('Save Complete\n')
```

## B. Compute_MDA_parameters_Two1DOFGimbals.m

```
%Generate Parameters for Two 1-DOF gimbal sim taken from Version 2.0
%LUVOIR Model 2.10 07 03 1 Coordinate System, Inertias, and Modes provided
%by Christine Collins.
%Rotates parameters into the frame appropriate for MDA which is illustrated
%in the LUVOIR Mass Properties FEM Model powerpoint from 7/9/2018 which credits Lia Sacks,
%Christine Collins, and William Bentz
```

```
mass_sc=11175.44;
mass_bo=314.552;
mass_pl=34618.22;
r_SC_CS=[-0.0515905;-10.24503;-2.314444]; %CS 0 Vertex frame
r_BO_CS=[-3.62e-4;-2.325406;-5.661404]; %CS 0 Vertex frame
r_PL_CS=[-1.41e-2;-3.555562;-1.471726]; %CS 0 Vertex frame
r_G1_CS=[0.000000;  -9.262678;  -5.665507]; %CS 0 Vertex
r_G2_CS=[0.000000;  -0.820928;  -5.665507]; %CS 0 Vertex
r_LV_CS=[-2.7227e-9;-11.45788;-2.328014]; %CS 0 Vertex
I_PL_CS=[1495501,5725.815,26731.22;
        5725.815,903967.1,112233;
        26731.22,112233,1453949];
I_BO_CS=[2906.019,0.0359722,0.000466717;
         0.0359722,8.062613,3.00417;
         0.000466717,3.00417,2909.028];
I_SC_CS=[288614,914.809,26111.78;
         914.809,553914.7,-597.5039;
         26111.78,-597.5039,287576.5];
O_PL_CS=[1,0,0;0,cosd(90),sind(90);0,-sind(90),cosd(90)]*[cosd(-90),0,-sind(-90);0,1,0;sind(-
90),0,cosd(-90)];%Rotate CS Frame -90 by y then 90 by x to get PL frame.
O_SCandBO_CS=[cosd(90),sind(90),0;-sind(90),cosd(90),0;0,0,1];  %Rotate CS Frame +90 degrees
about Z to get SC frame

I_PL=O_PL_CS*I_PL_CS*transpose(O_PL_CS) %Moment of inerita of payload in payload frame
I_BO=O_SCandBO_CS*I_BO_CS*transpose(O_SCandBO_CS) %Moment of inertia of boom in boom frame
I_SC=O_SCandBO_CS*I_SC_CS*transpose(O_SCandBO_CS) %Moment of inertia of SC in SC frame.

r_PL_G2=O_PL_CS*(r_PL_CS-r_G2_CS)
r_G1_SC=O_SCandBO_CS*(r_G1_CS-r_SC_CS)
r_BO_G1=O_SCandBO_CS*(r_BO_CS-r_G1_CS)
r_G2_G1=O_SCandBO_CS*(r_G2_CS-r_G1_CS)
r_SC_LV=O_SCandBO_CS*(r_SC_CS-r_LV_CS)
r_G1_LV=O_SCandBO_CS*(r_G1_CS-r_LV_CS)
```

## References:

[1] William Bentz, Lia Sacks. Derivation of Equations of Motion and Model-based Control for 3D Rigid Body Approximation of LUVOIR. Unpublished technical note, 2018.

[2] 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.

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