Fast Maneuvering Spacecraft with RCS and Single Gimbal CMGs

In this example we have a spacecraft that requires high precision pointing for optical instruments, such as a telescope, or a laser beam. Its Line-of-Sight (LOS) which is pointing at targets is in the body x direction. The spacecraft points at a target for a period of time and then it rotates to point at another target, and then another. The spacecraft is described to be an agile spacecraft because under normal operations it is constantly maneuvering between targets and the retargeting must be completed as fast as possible. The on board Attitude Control System (ACS) operates in different modes depending on the circumstances. It uses a combination of reaction control (RCS) jets and Single-Gimbal Control Moment Gyros (SG-CMG). In normal mode of

operation the ACS uses four Control Moment Gyros which provide high torque required for fast maneuvering between targets. The CMGs use solar energy and they do not require fuel, they also provider smoother operation than RCS. The spacecraft uses the RCS jets for momentum desaturation and also as an ACS backup. It also has a fixed thruster engine for orbital maneuvering. The spacecraft has a flexible structure because of the solar arrays and other communication appendages that require finite element modeling and detailed flex analysis. To further complicate the analysis there is also a tank on board containing fuel for the main engine and the RCS jets which effects stability during re-boost and introduces oscillatory disturbances that may degrade the LOS pointing accuracy as the fuel is sloshing inside the tank.

The following analysis focuses mainly in the two main modes of operation, the RCS and the CMG attitude control. We will develop several dynamic models for this flexible spacecraft, design control laws and analyze the ACS stability and performance in various modes of operation. The analysis begins with simple rigid-body models and it gradually becomes more complex as we include structural flexibility and fuel sloshing dynamics. We also gradually increase the complexity of the control laws, starting with a simple phase-plane 3-dof logic and upgrading it to a 6-dof logic that provides simultaneous translation and attitude control. The modified logic is also designed to reduce fuel consumption by pulse-width-modulating the RCS jets. Other control ideas, such as, using blended CMGs and Reaction Wheels (RW), and combined RCS with RW configurations are also evaluated.

In section 1 we use the Flixan program to prepare various flexible spacecraft dynamic models that will be used in later chapters. Two sets of models are created using the “Flexible Spacecraft FEM” program (FEM), and the “Flight Vehicle Modeling” program (FVP), and they are saved in two separate folders. The modeling section can be skipped if the user is already familiar with vehicle modeling in Flixan and may jump to the more interesting analysis sections. In section 2 we design and analyze the RCS system. We begin with a simple 3-dof phase-plane, combined with a dot-product jet-selection logic, and gradually upgrade it into a more advanced logic that minimizes fuel usage. The method is augmented to 6-dof by including also translational control. In section 2.6 we develop a non-linear slosh model for a partially filled fuel tank. This model is used for zero g or low g environment. It is combined with the spacecraft model and used to perform stability analysis. In section 3 we develop the Max Energy, non-linear control law for a 4 SG-CMG array with a singularity avoidance algorithm and implement it in various simulation models. Finally, in chapter 4 we present simulation models that demonstrate multi-mode operations.

1.0 Preparation of the Structure Flexibility Models

Structural flexibility is an important factor to consider when analyzing stability and performance of this spacecraft. It has solar arrays, antennas, and other appendages mounted on its relatively solid bus structure that create low damped flex resonances which induce disturbances on the spacecraft when they get excited by the spacecraft motion as it maneuvers around. This causes degradation in LOS pointing, and jitter in optical imaging. Flexibility may also cause structural instability if not filtered properly. In this section we are describing how to use the Flixan program to generate linear models of the spacecraft that include structural flexibility and fuel sloshing. If you are already familiar with the Flixan modeling process you may bypass this section and go to the more interesting stuff in Section 2.

The first 46 structural modes of this spacecraft are saved in file "*FlexSc_Rcs.Mod*". They were obtained from a finite element modeling (FEM) program. The title of the modal data is "*Flexible Spacecraft with Solar Array, RCS and CMGs*". There is a block of data for each mode (resonance) and there are 46 blocks of data. Each block contains the mode shapes and slopes of a particular resonance at 19 vehicle locations (nodes). The first six modes are rigid-body modes at zero frequency and they define the rigid-body motion of the spacecraft. We typically throw away the first 6 rigid-body modes because we have our own rigid-body models which are more efficient. The real structural modes start with mode number 7 which is at 0.5 Hz. Another important file for flex model preparation is the nodes map, file "*FlexSc_Rcs.Nod*", which describes the identity of the 19 structural locations (nodes) included in the modal data file, in the same order as they are listed there. The map file is used in menus by the model preparation programs.

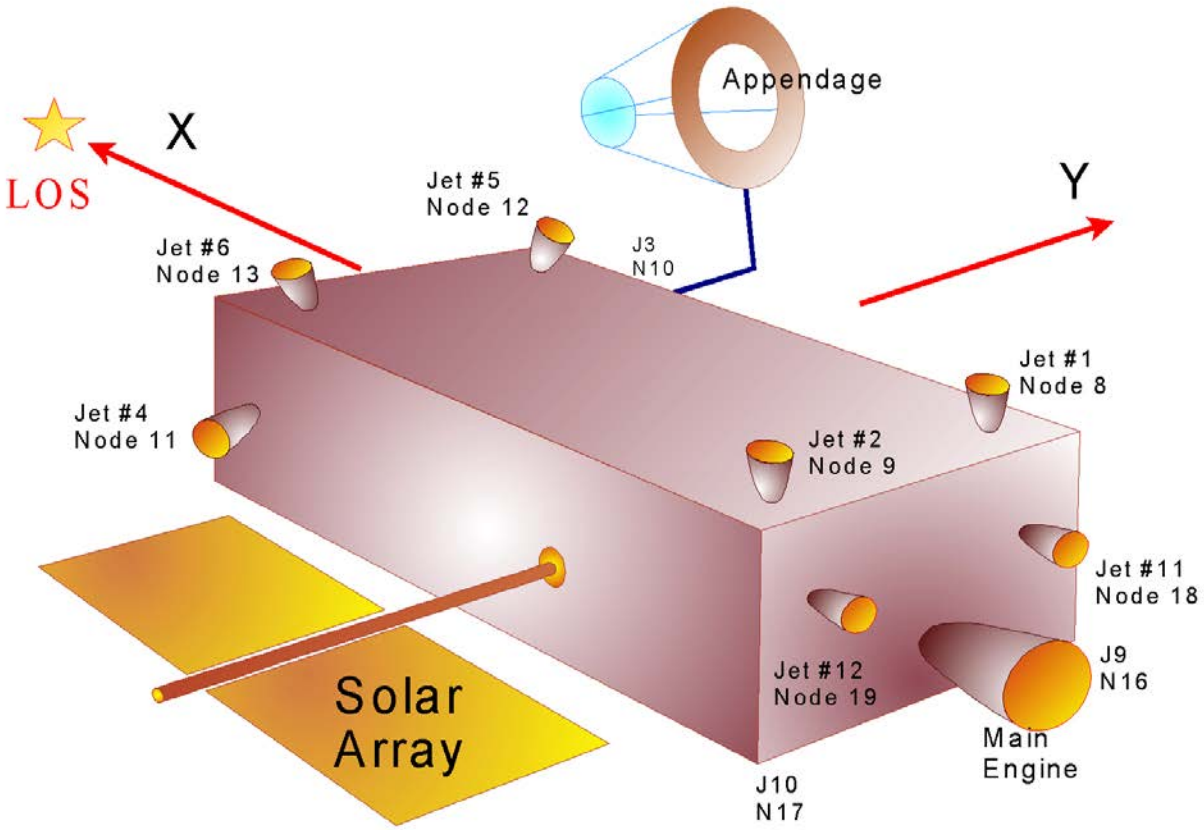
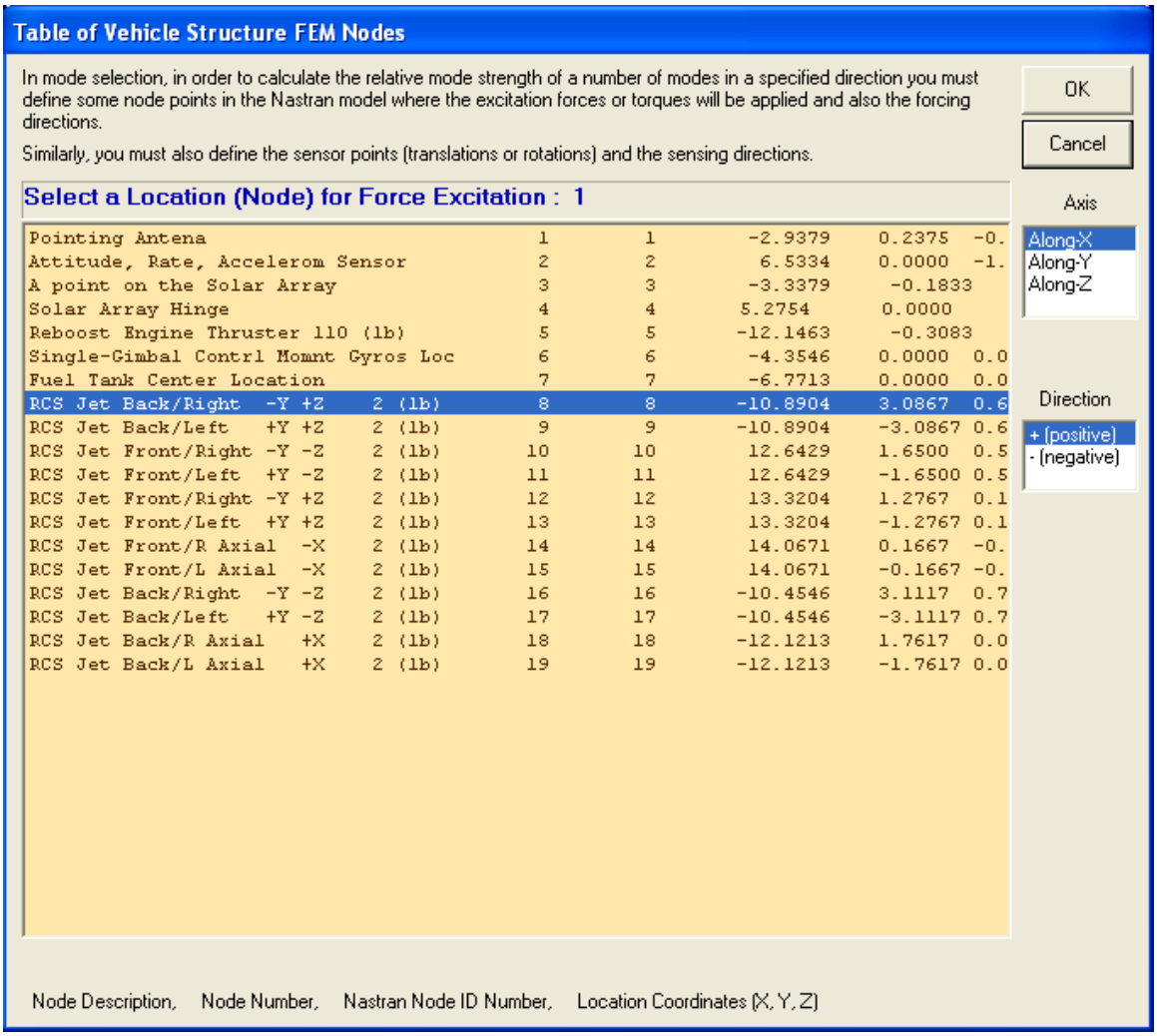
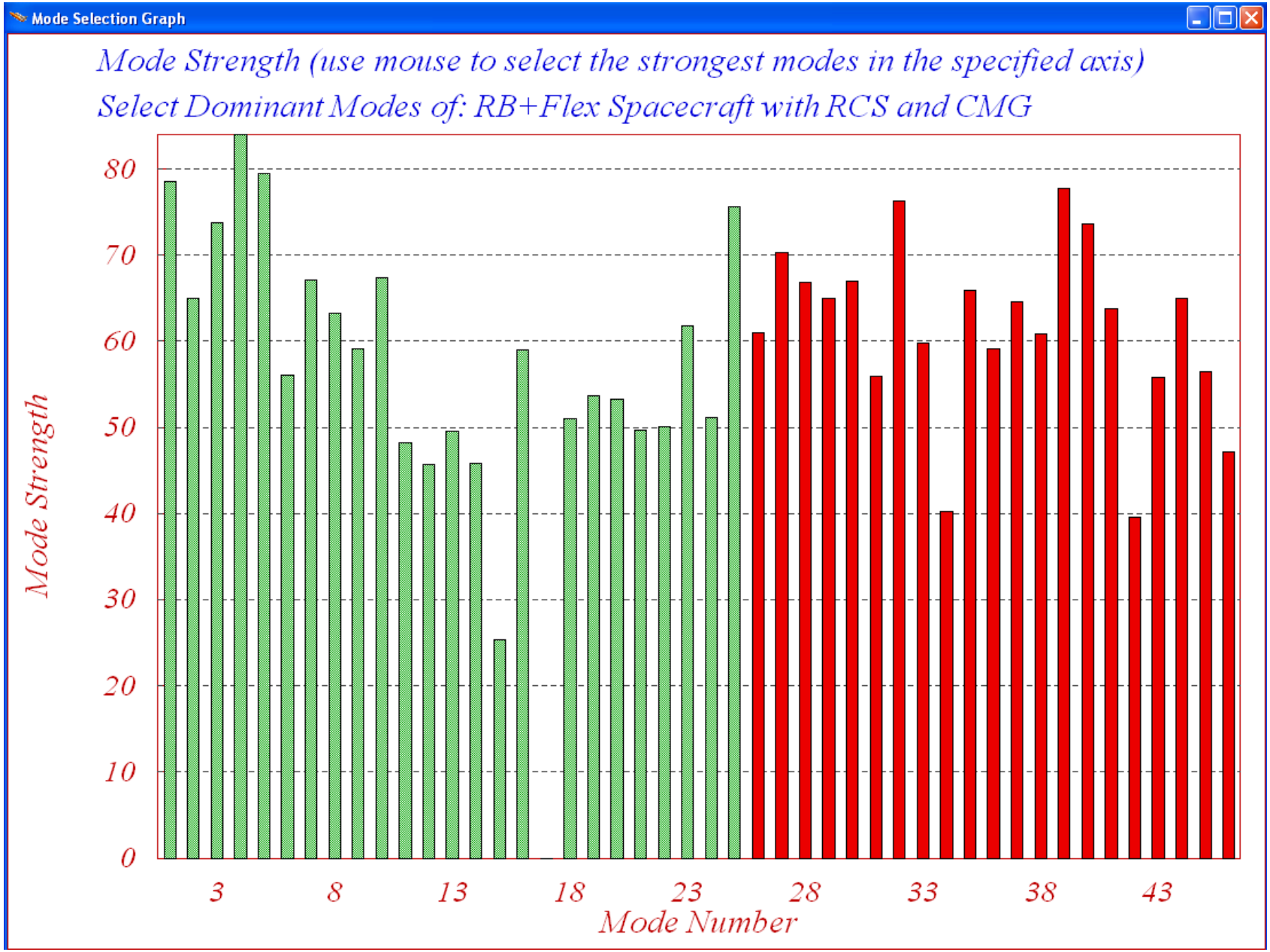


Figure 1 Locations of the RCS Jets and the Main Engine

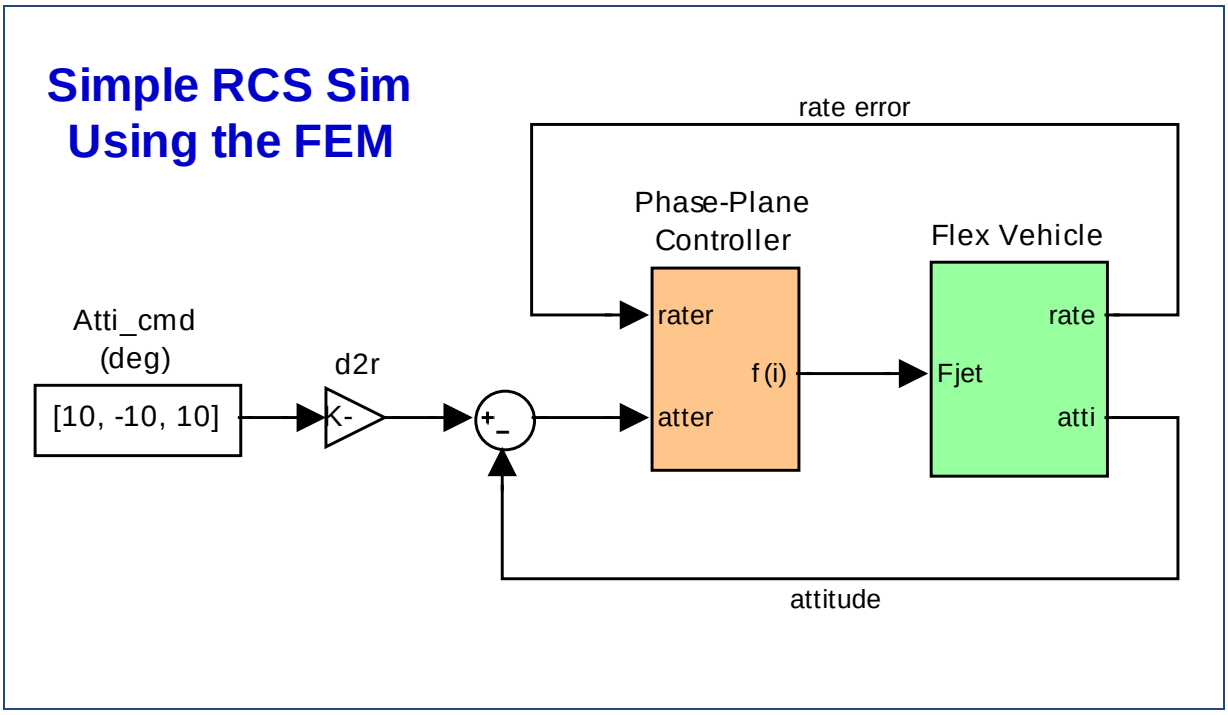


The program finally calculates the modal strengths for the 46 modes and displays it in bar-chart. Each bar corresponds to a mode number. The user must select all the modes from the chart, shown below, by clicking on the bar with the mouse. When a mode is selected its color changes from red to green.

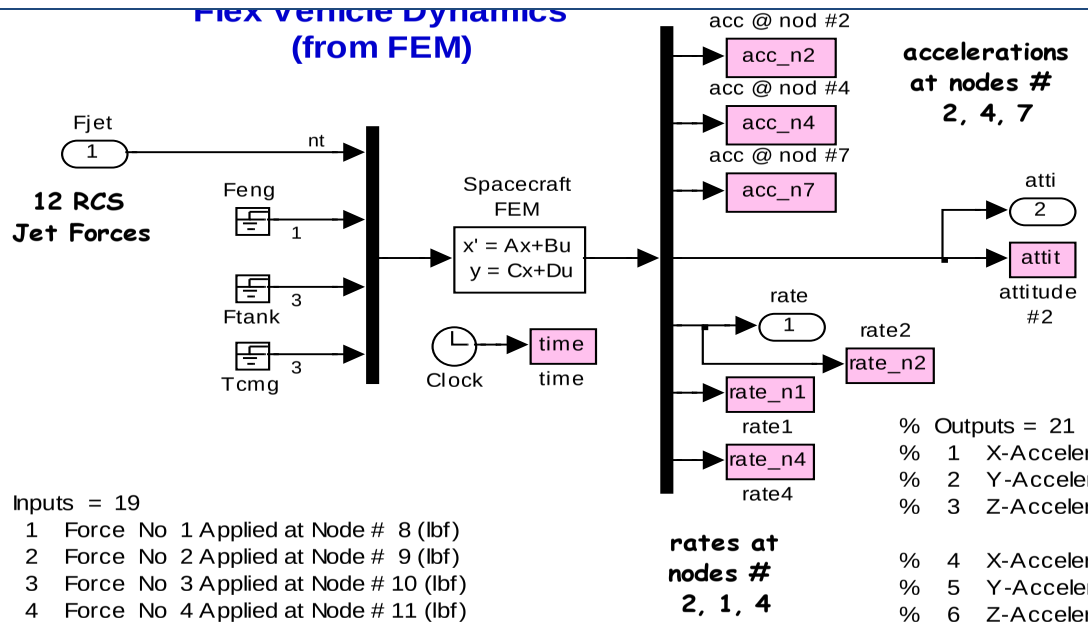

Mode Selection Graph
Mode Strength (use mouse to select the strongest modes in the specified axis)
Select Dominant Modes of: RB+Flex Spacecraft with RCS and CMG
Mode Strength
0
10
20
30
40
50
60
70
80
3
8
13
18
23
28
33
38
43
Mode Number

Simple Simulation Using the Finite Element Model

The following simulation model "*Flex_Sim.mdl*" uses the system title "*RB+Flex Spacecraft with RCS and CMG*", which was created using modal data as described earlier. It is saved in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis\(a) Flex Models from Spacecraft FEM\Matan*".



The simulation consists of the spacecraft dynamics and a simple phase-plane controller "*Phase-Plane.m*". It is initialized by file "*start.m*" which loads the spacecraft state-space system from file "*flex_spacecraft_fem_s.m*". The rigid-body modes are included in the finite element model. The following figure shows the spacecraft attitude response to 10 (deg) attitude commands in different directions. It also shows accelerometer and rate gyro responses in three spacecraft locations. This is a simple model to begin. We will continue with more complex models in the following sections.



% Inputs = 19
% 1 Force No 1 Applied at Node # 8 (lbf)
% 2 Force No 2 Applied at Node # 9 (lbf)
% 3 Force No 3 Applied at Node # 10 (lbf)
% 4 Force No 4 Applied at Node # 11 (lbf)
% 5 Force No 5 Applied at Node # 12 (lbf)
% 6 Force No 6 Applied at Node # 13 (lbf)
% 7 Force No 7 Applied at Node # 14 (lbf)
% 8 Force No 8 Applied at Node # 15 (lbf)
% 9 Force No 9 Applied at Node # 16 (lbf)
% 10 Force No 10 Applied at Node # 17 (lbf)
% 11 Force No 11 Applied at Node # 18 (lbf)
% 12 Force No 12 Applied at Node # 19 (lbf)

% 13 Force No 13 Applied at Node # 5 (lbf)

% 14 Force No 14 Applied at Node # 7 (lbf)
% 15 Force No 15 Applied at Node # 7 (lbf)
% 16 Force No 16 Applied at Node # 7 (lbf)

% 17 Torque No 1 Applied at Node # 6 (ft-lb)
% 18 Torque No 2 Applied at Node # 6 (ft-lb)
% 19 Torque No 3 Applied at Node # 6 (ft-lb)

% Outputs = 21
% 1 X-Accelerat. Sensed at Node # 2 (ft)
% 2 Y-Accelerat. Sensed at Node # 2 (ft)
% 3 Z-Accelerat. Sensed at Node # 2 (ft)

% 4 X-Accelerat. Sensed at Node # 4 (ft)
% 5 Y-Accelerat. Sensed at Node # 4 (ft)
% 6 Z-Accelerat. Sensed at Node # 4 (ft)

% 7 X-Accelerat. Sensed at Node # 7 (ft)
% 8 Y-Accelerat. Sensed at Node # 7 (ft)
% 9 Z-Accelerat. Sensed at Node # 7 (ft)

% 10 X-Rotation Sensed at Node # 2 (radian)
% 11 Y-Rotation Sensed at Node # 2 (radian)
% 12 Z-Rotation Sensed at Node # 2 (radian)

% 13 X-Rot. Rate Sensed at Node # 2 (radian)
% 14 Y-Rot. Rate Sensed at Node # 2 (radian)
% 15 Z-Rot. Rate Sensed at Node # 2 (radian)

% 16 X-Rot. Rate Sensed at Node # 1 (radian)
% 17 Y-Rot. Rate Sensed at Node # 1 (radian)
% 18 Z-Rot. Rate Sensed at Node # 1 (radian)

% 19 X-Rot. Rate Sensed at Node # 4 (radian)
% 20 Y-Rot. Rate Sensed at Node # 4 (radian)
% 21 Z-Rot. Rate Sensed at Node # 4 (radian)

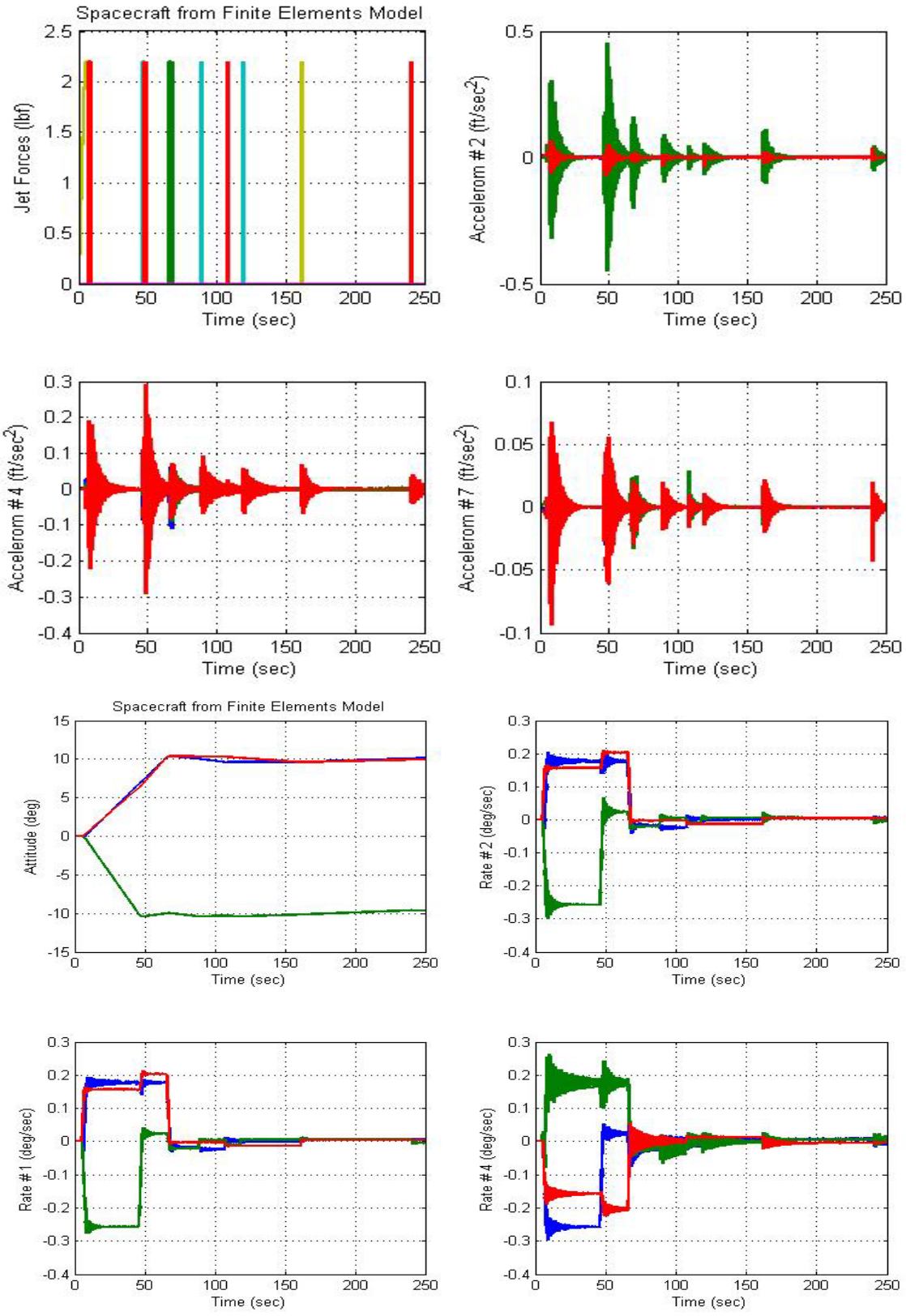


Figure 1.1.1 Flexible spacecraft response to 10 (deg) attitude commands.

Frequency Domain Stability Analysis

In the same folder there is also a Simulink model "*Open_Loop_RCS.mdl*" used for analyzing RCS flex mode stability using the Describing Function (DF) method.

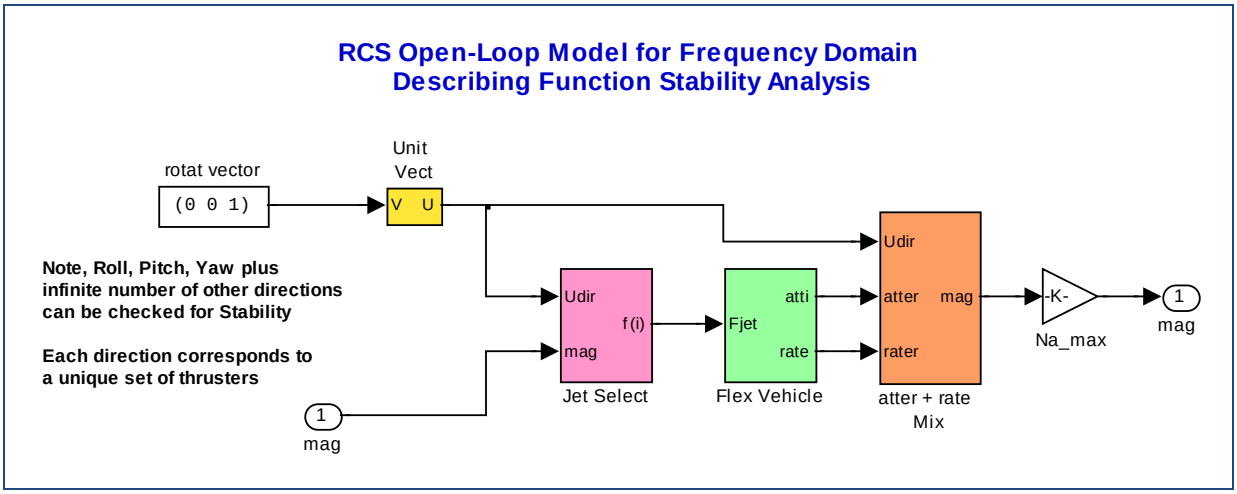


Figure 1.1.2 Frequency domain model for RCS stability analysis using Describing Function

This model is an open-loop linearized version of the previous closed-loop model. It consists of the same basic subsystems, slightly modified for frequency domain analysis. Stability is evaluated one axis (rotational direction) at a time. The rotational direction is an input to the model. In the example above we are analyzing the yaw axis. By modifying the rotation vector we can analyze pitch (0, 1, 0) and roll (1, 0, 0). It is also possible to analyze stability in many other skewed directions such as: (0, 0.7, -0.7) or (0.5, -0.3, 0.6), since every direction corresponds to a unique set of thrusters exciting the flex modes differently. For linear analysis the complex non-linear phase-plane logic is approximated with a linear combination of attitude plus rate errors. The linearized jet selection logic is also an approximation because it averages the positive and negative accelerations about a specific rotational vector. It selects not only the positive direction jets (with half thrust) but also the jets that accelerate in the opposite direction but assuming negative half thrusts, thus, exercising both positive and negative jets. The system output is multiplied by the max value of the dead-band DF to scale the Nichols charts so that the Nichols critical point (+) corresponds to the min of the DF inverse, that is -1/N(a). The Matlab file "frequ.m" uses the above model to calculate the frequency response and plot the Nichols charts as shown in Figure (1.1.3) below.

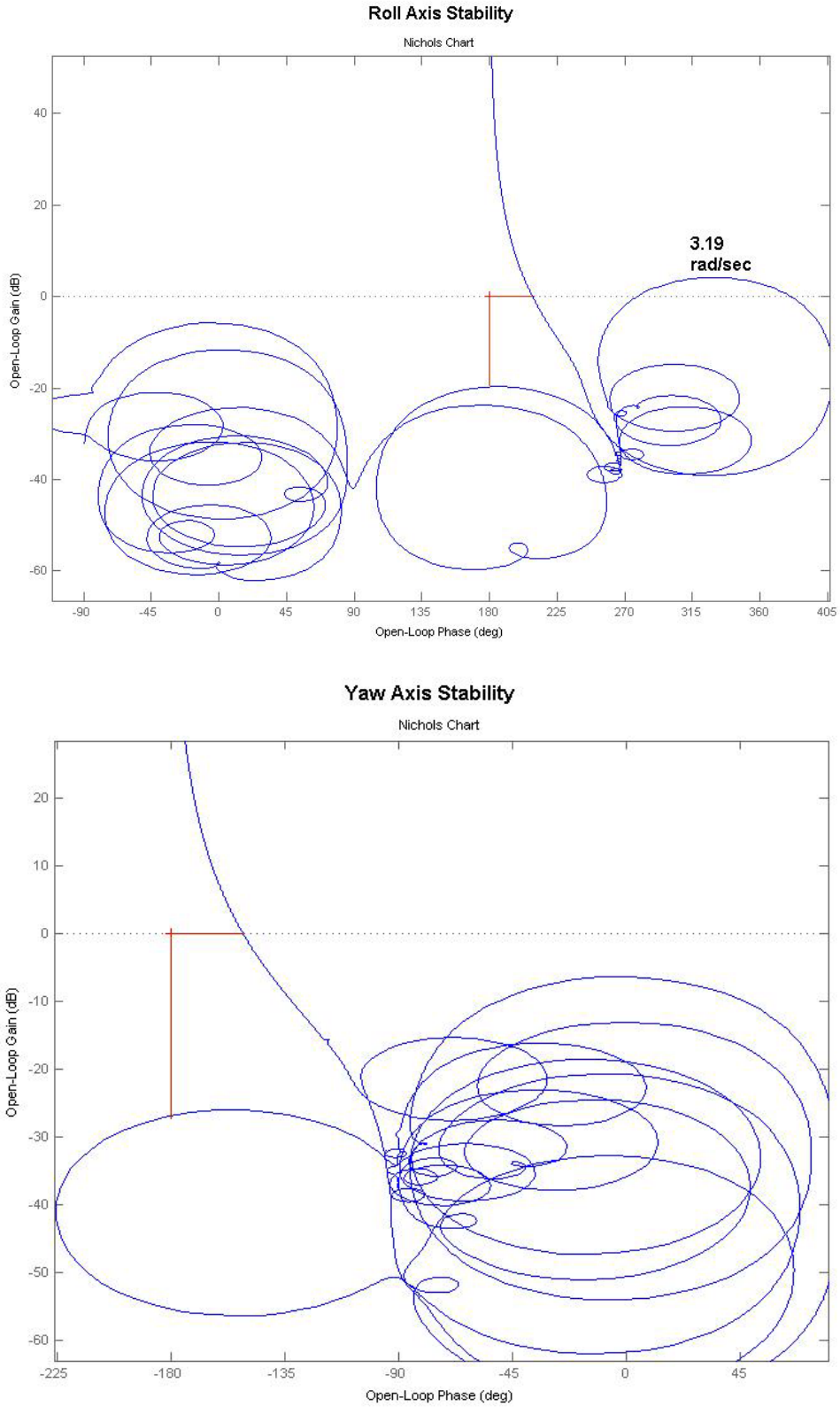


Figure 1.1.3 Nichols Charts showing stability margins from the minimum point of -1/N(a)

2.0 RCS Control

In this section we will design the RCS control laws, starting with a simple phase-plane logic and advancing to a more complex jet selection that minimizes fuel usage. We will also perform analysis and simulations, starting with a simple rigid-body non-linear simulation, and gradually advance to more complex models that include structural flexibility and fuel sloshing. We will present a non-linear model for modeling propellant sloshing at zero or low g, and finally analyze RCS stability in the frequency domain with flexibility and fuel sloshing.

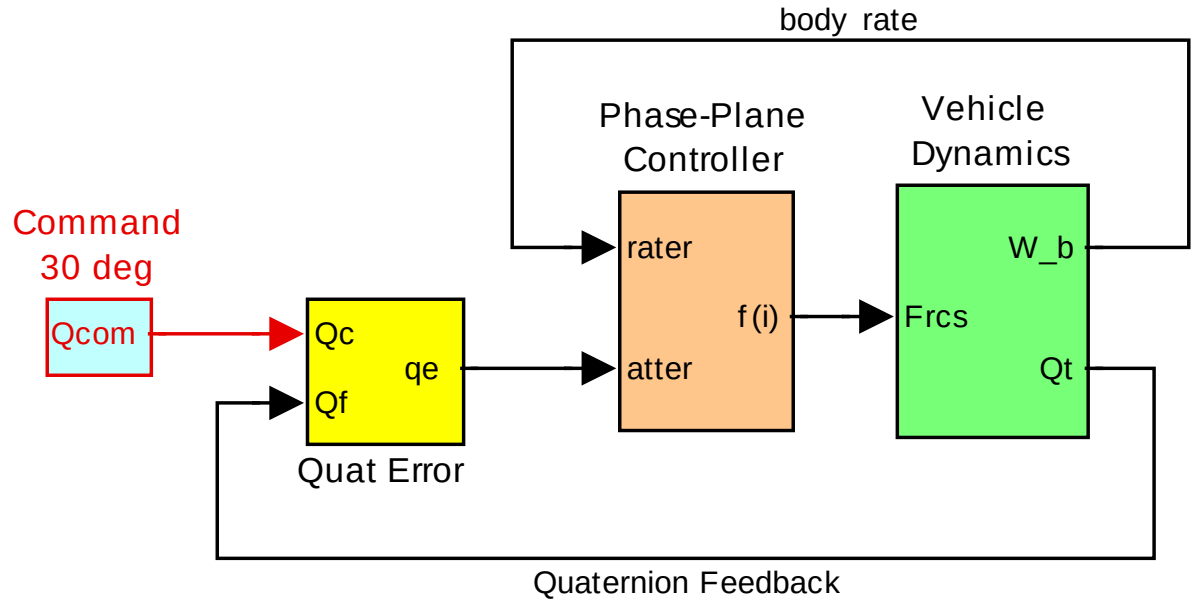


Figure 2.1 Attitude Control System

Let us begin with a simplified version of the RCS Attitude Control System shown in Figure (2.1). It consists of the spacecraft dynamics inside the green block, a phase-plane attitude control system (orange block), a quaternion attitude command generator, and a quaternion error block (yellow).

2.1 Attitude Control System

The orange block in Figure (2.1) is the Attitude Control System (ACS) which consists of the phase-plane logic and the jet-selection logic. The yellow block calculates the attitude error. The inputs to the phase-plane logic are attitude errors, and vehicle rates. The phase-plane calculates the demanded change in vehicle rate, which is a vector about which the vehicle must rotate in order to move from the initial orientation to the commanded attitude. The jet selection logic translates the rate command vector into jet firing. It uses dot product to calculate the torque contributions from all 12 jets in the commanded direction and it selects a few jets that contribute the biggest moment in that direction. The output from jet-select is a vector of 12 jet forces. Most of them are "off". The logic fires between 2 to 4 jets at a time, depending on the commanded direction. Figure (2.2) shows the phase-plane logic in one axis. It determines the acceleration direction from the rate and attitude errors. There are 3 separate phase-planes operating simultaneously for roll, pitch, and yaw axes. Each plane consists of three regions, a region of zero firing, a region of positive jet firing, and a region of negative jet firing in the corresponding direction. The firing decision is made based on the combined rate and attitude error in the direction that reduces error.

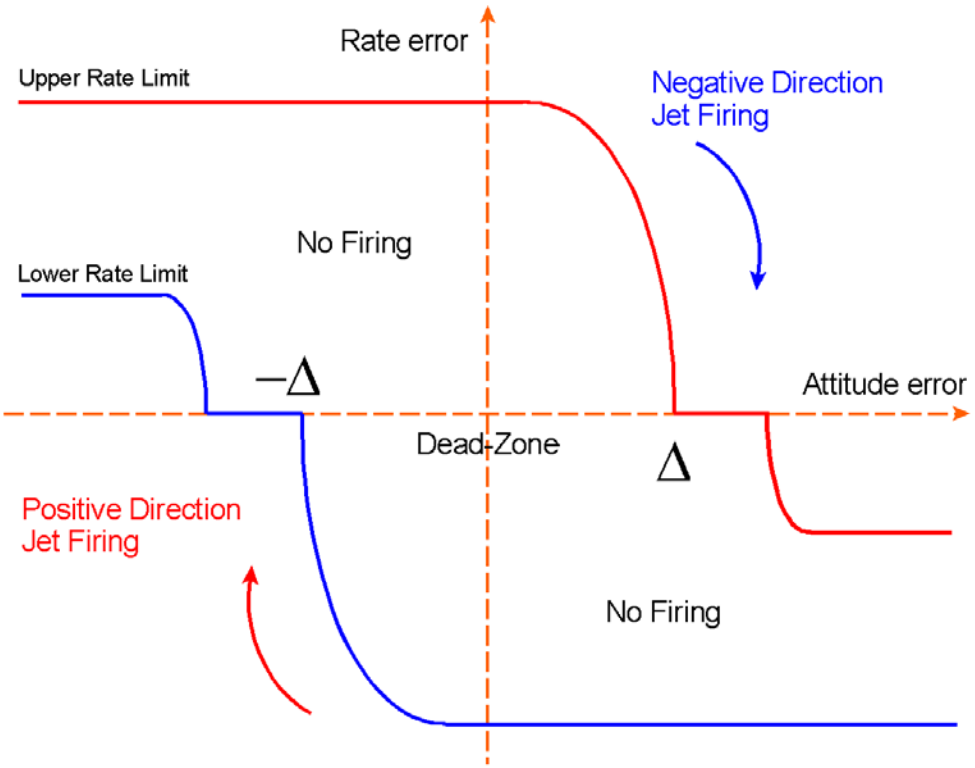


Figure 2.2 Phase-Plane Shows Conditions for Jet Firing

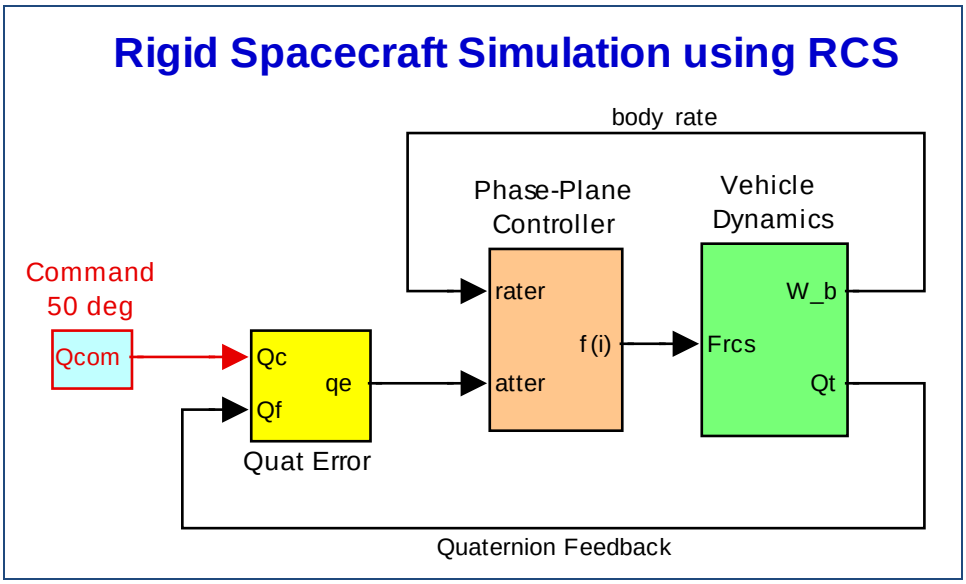
2.2 RCS Simulation Models Using the Dot-Product Jet Selection Logic

In this design example we begin with simple models and gradually upgrade them to more complex ones. As the design progresses we are going to analyze various simulation models with increasing complexity in terms of spacecraft dynamics and also in control design. To avoid confusion, therefore, each simulation model with the associated Simulink and data files are placed in separate folders and they will be analyzed separately. We start with a 3-dof rigid-body ACS simulation that uses dot-product jet selection logic. Then we apply the same ACS control system to the flex models created in Section 1 using two separate Flixan methods. We compare the simulation results obtained from the two models, both in time and also in frequency domain.

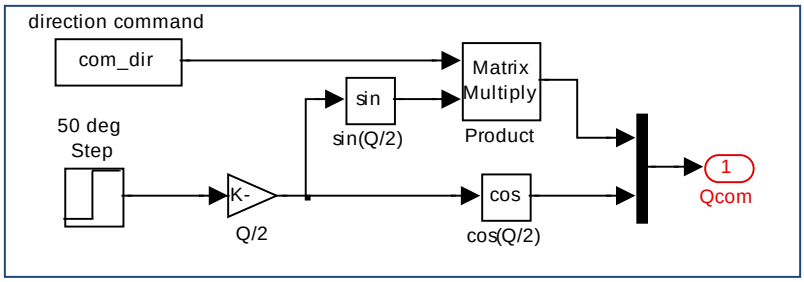
The next step is to upgrade the ACS design and to replace it with a fuel optimal jet selection logic that significantly reduces fuel usage. Following that, we further upgrade the minimum fuel control logic to accommodate also translational in addition to rotational control. We finally test the full 6-dof minimum fuel control logic using a non-linear spacecraft model with flexibility, the flex dynamics being connected in parallel with the previously used non-linear rigid-body model. This demonstrates a wide variety of options which are available for reaction control modeling and analysis.

2.2.1 Rigid-Body Non-Linear Simulation

We start the RCS analysis with a simple rigid-body non-linear simulation model in order to demonstrate the phase-plane and jet-selection logic. The files used in this model are in folder "…*\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis****(c) NonLin RigBody RCS Attitude Control***". The Simulink model is "*RB_Sim_RCS.Mdl*" shown in Figure (2.2.1).



The spacecraft dynamics are implemented in Matlab function "*SV_Dynamics.m*". The spacecraft outputs are body rate and attitude quaternion. The quaternion error (yellow) block receives the attitude quaternion command and the quaternion feedback from the spacecraft and calculates the attitude error which consists only of the 3-axis vector part of the quaternion error, representing attitude errors in roll, pitch and yaw. The magnitude part of the 4-dimensional quaternion error is ignored. The Matlab function "*qerror2.m*" calculates the quaternion error. The phase-plane and jet-selection logic are coded in Matlab functions "*Phase_Plane.m*", and "*Jet_Select_dot.m*". Jet-select is called by the phase-plane logic. The phase-plane parameters such as the dead-band and rate limits are loaded into Matlab workspace by the initialization file "start.m" which must be executed prior to the simulation. Other parameters are also loaded, such as, number of jets, thrust, jet locations, thrust directions, moments of inertia, and cg location. They are used by the jet selection logic to determine which jets should be fired in order to provide rotation in the direction commanded by the phase-plane logic.



The attitude command is defined as a quaternion rotation from the current attitude, which is assumed to be zero, i.e. (0,0,0,1). The quaternion command is a 4-dimentional vector consisting of a direction about which the vehicle should rotate, and the angle of rotation. The block [Qcom] is shown in detail in Figure (2.2.2). The direction "com_dir" is defined in file "start.m", and the rotation angle (50 deg) is defined inside the block. After running the Simulink model, execute file "pl.m" which will plot the simulation results, as shown in figure (2.2.3) below.

2.2.2 Comparing the Linear Models with Structural Flexibility in Closed-Loop Sims

The next step in our RCS analysis is to check out by means of simulations the two types of flex spacecraft models that were created in sections 1.1 and 1.2. The first one was created by the flex spacecraft FE modeling program, and the second one was created by the flight vehicle modeling program. We want to make sure that similar results are obtained from both models before we continue any further. The first one has the rigid-body dynamics represented by rigid-body modes from the FEM, while the second one is using its own rigid-body model and only the flex modes are taken out of the FEM. The files for the Simulink models are in the same folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis* ***(d) RCS Attitude Control w Flex***". It is a good practice to test and compare these two systems before we move on to more complex models.

Let us begin with the first simulation which is in file "*Sim_Flex_fem_z.mdl*" and uses the FEM, see Figure (2.2.4). The spacecraft system is in file "*flex_spacecraft_fem_z.m*". It includes 40 structural modes and 6 rigid-body modes, a total of 46 modes. It was discretized with a sampling period of T_s=5 msec, and its title is "*RB+Flex Spacecraft with RCS and CMG (Z-Transf)*".

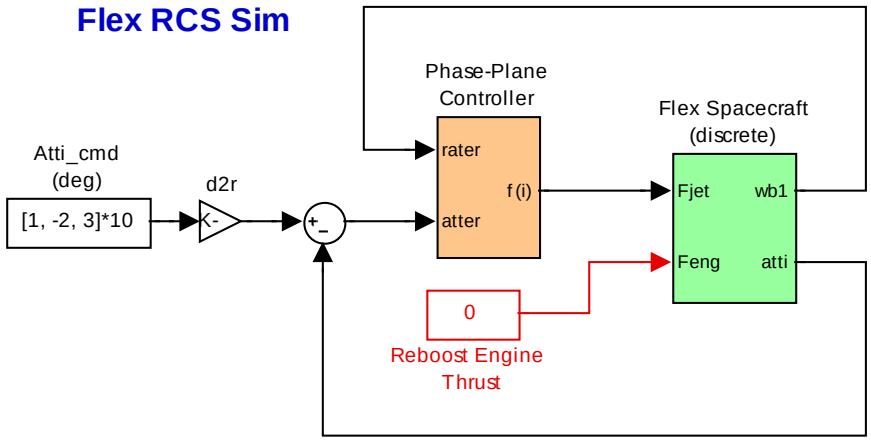


Figure 2.2.4 Linear Flex Spacecraft Simulation "Sim_Flex_fem_z.mdl"

The ACS consists of the phase-plane and the dot-product jet selection logic which is sampled slower, at 20*Ts= 0.1 sec. There is a fuel counter which integrates the sum of jet thrusts: $\int \sum_{i=1,nt} F_{(i)} dt$. The rate transition blocks separate the flex spacecraft (which is sampled at 5 msec) from the ACS (which is sampled every 100 msec). The ACS logic is implemented as a Matlab Function in file "*Phase_Plane.m*", and it is similar to the one described in section 2.2.1. The jet-select output is normalized to unity and it is, therefore, divided by the jet thrust (Th). There is no translation control yet.

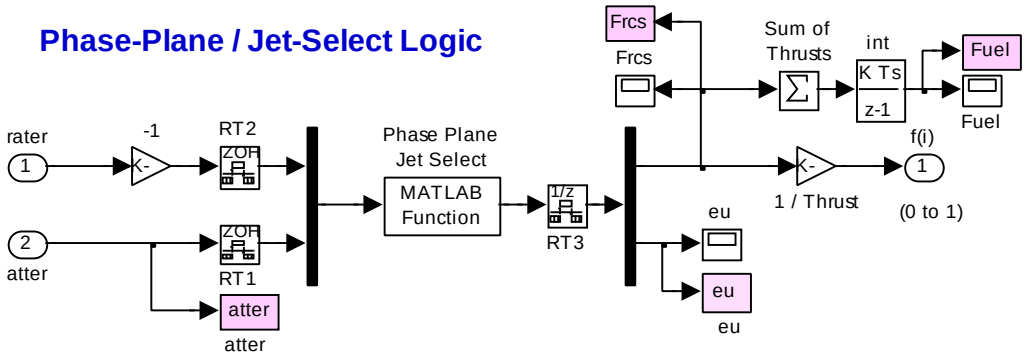


Figure 2.2.5 Phase-Plane and Jet Selection Logic

The discrete flex spacecraft block is shown in detail in Figure (2.2.6). Its inputs are 12 RCS jet forces, and the orbital maneuvering engine thrust (which is not used in this case). The outputs are attitudes, rates, and accelerations at specific vehicle locations defined in section 1.1 during model preparation. File "start.m" initializes the spacecraft and ACS parameters for the simulations, and file "pl.m" plots the simulation data, as shown in figure (2.2.7).

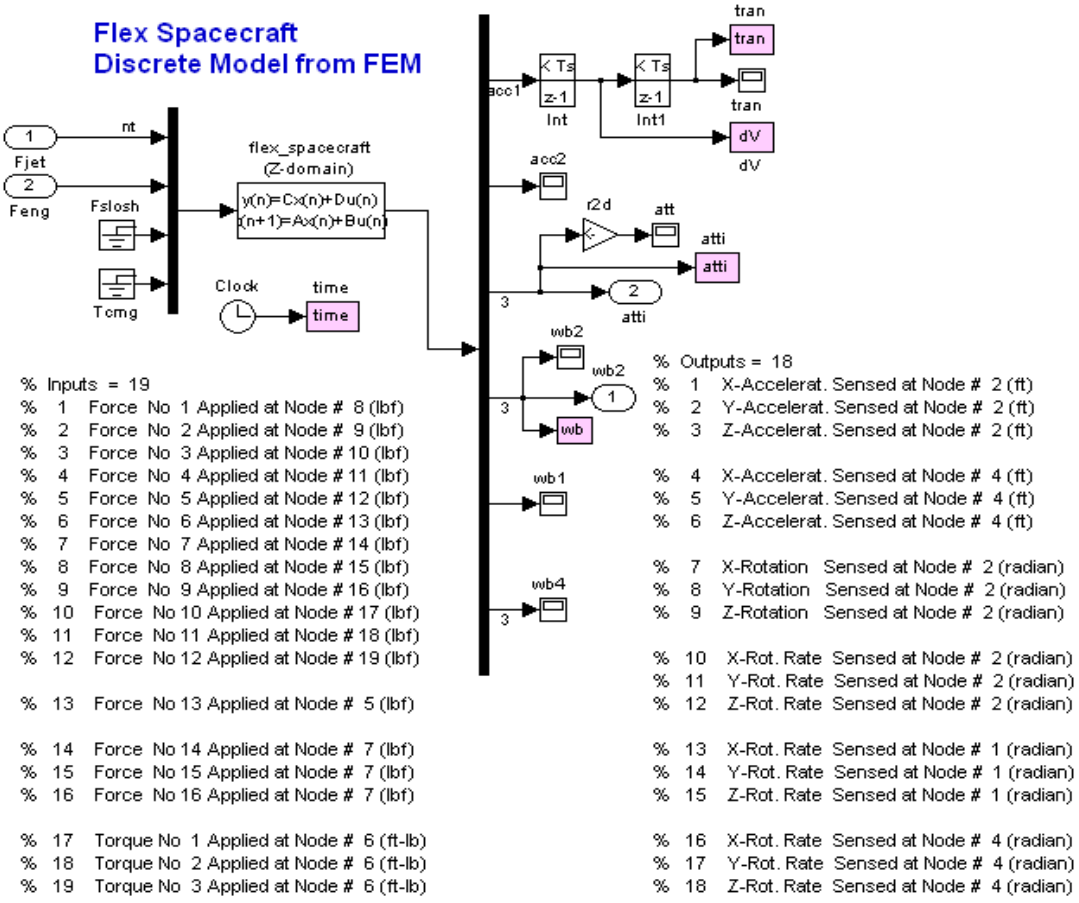
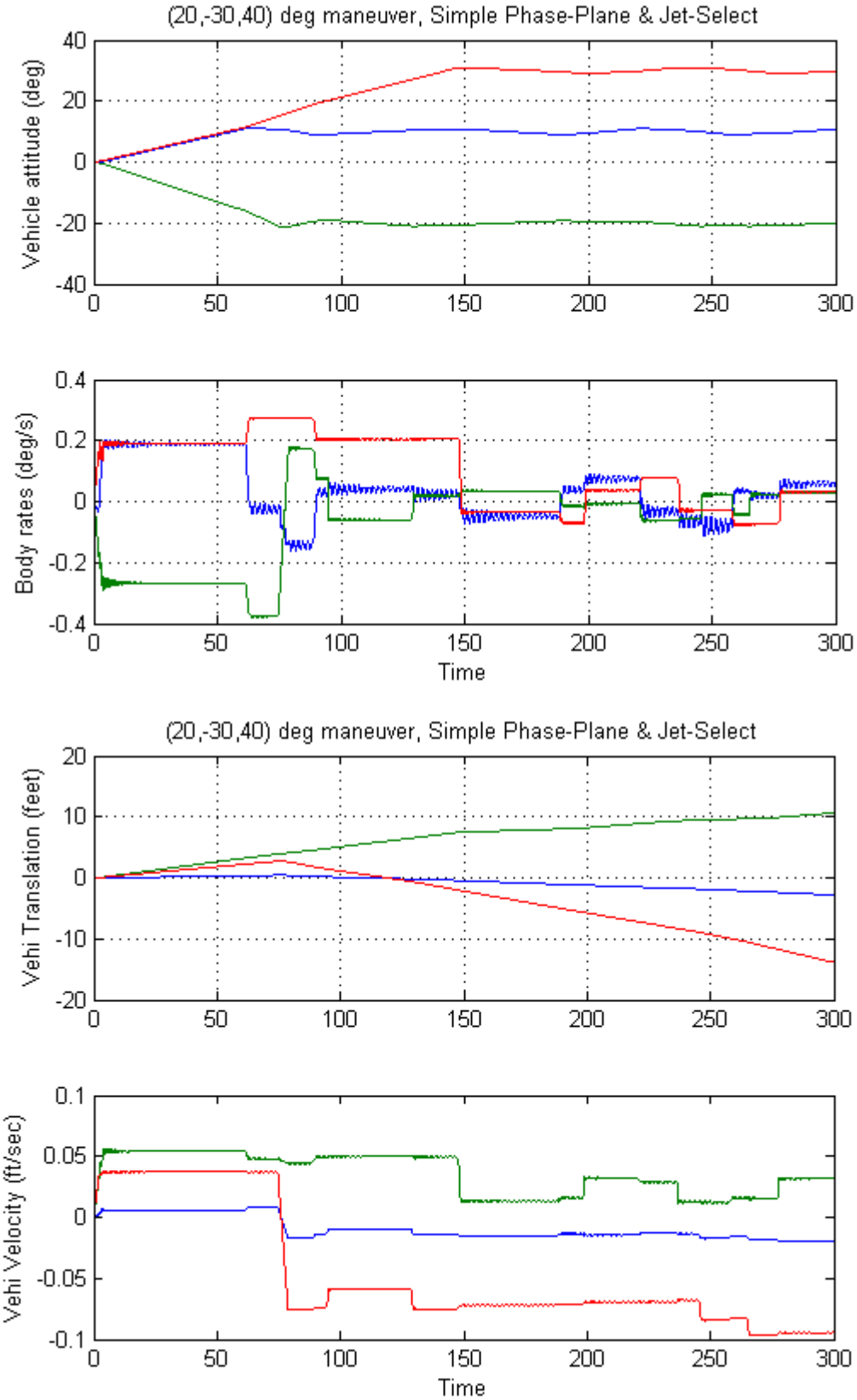


Figure 2.2.6 Discrete Spacecraft State-Space System from file "flex_spacecraft_fem_z.m", system originated from the Flex Spacecraft Modeling Program.


(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehicle attitude (deg)
Body rates (deg/s)
Time
(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehi Translation (feet)
Vehi Velocity (ft/sec)
Time

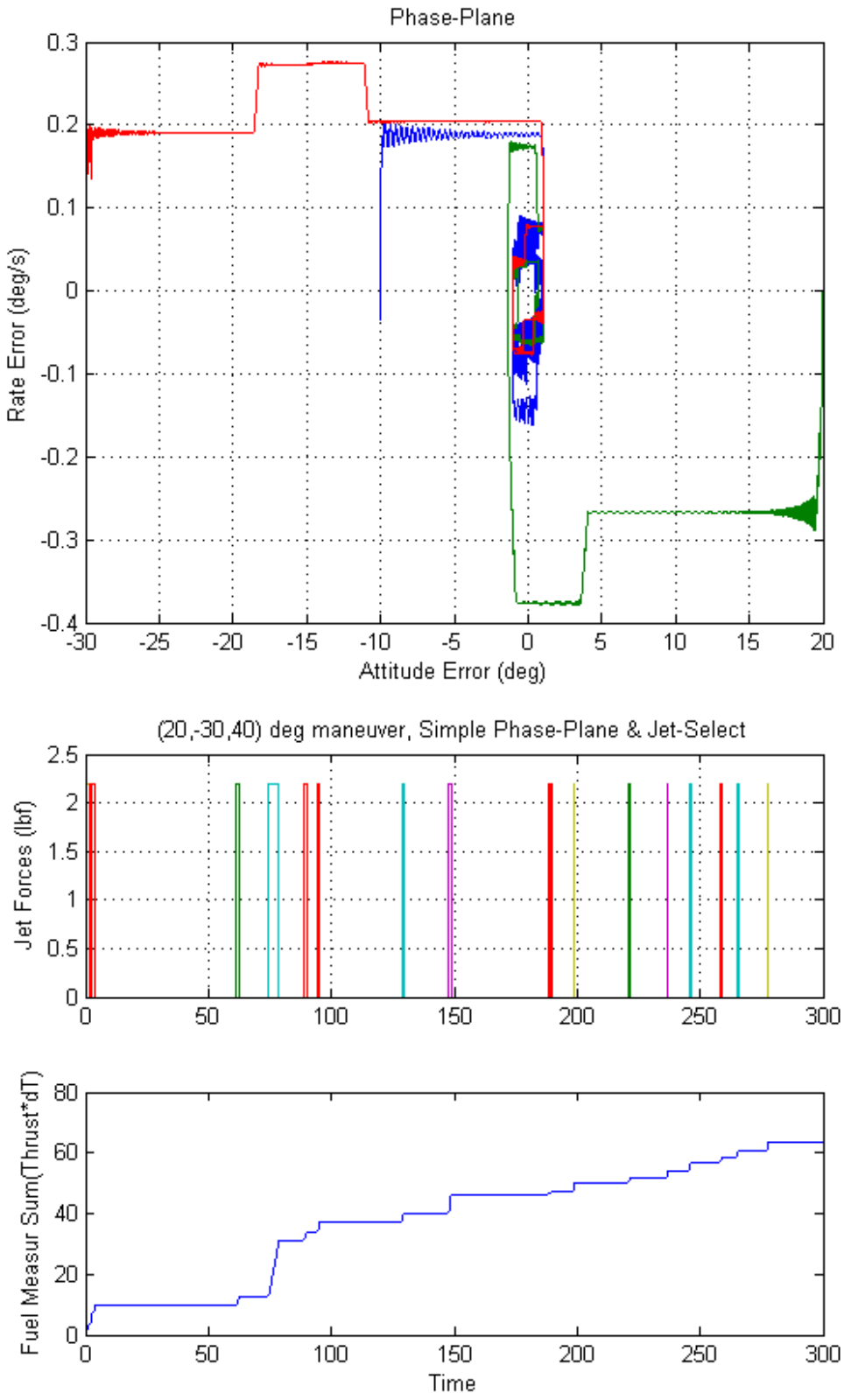


Figure 2.2.7 Simulation Results from "Sim_Flex_fem_z.mdl"

The second Simulink model "*Sim_Flex_fvp_z.mdl*" is almost identical to the first one, but uses the discrete system "*flex_satellite_fvp_z.m*", which was created by the "Flight Vehicle Modeling Program" in Section 1.2. The spacecraft dynamics block is shown in detail in Figure (2.2.7).

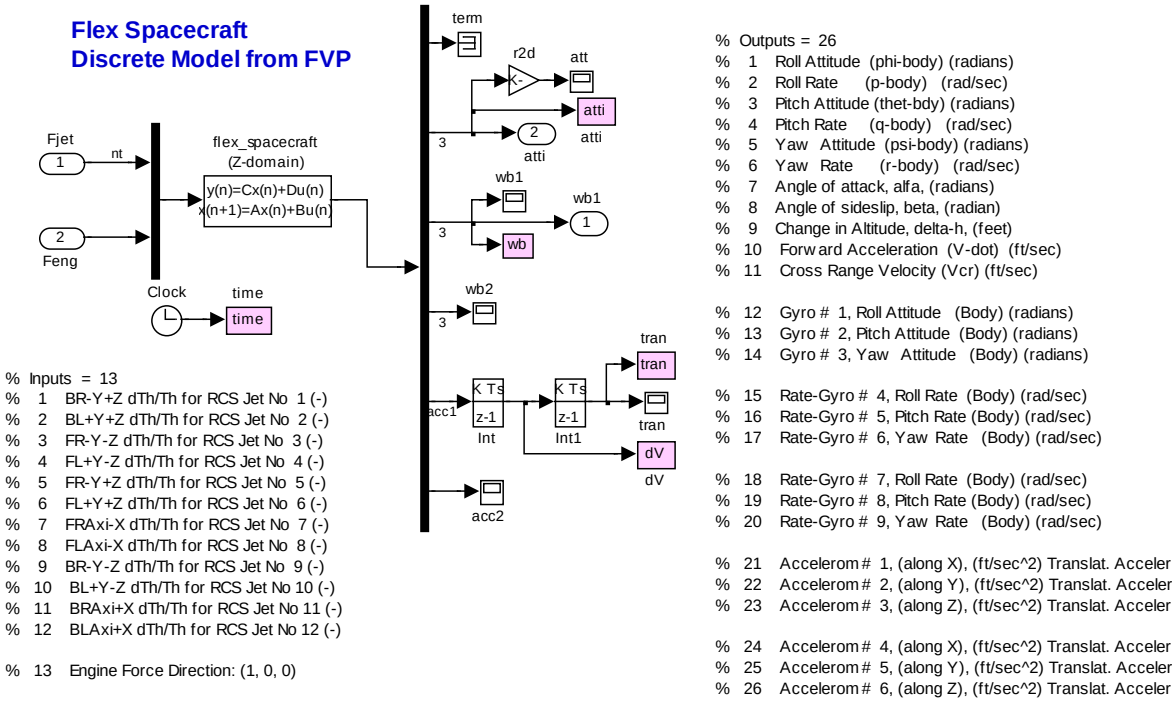
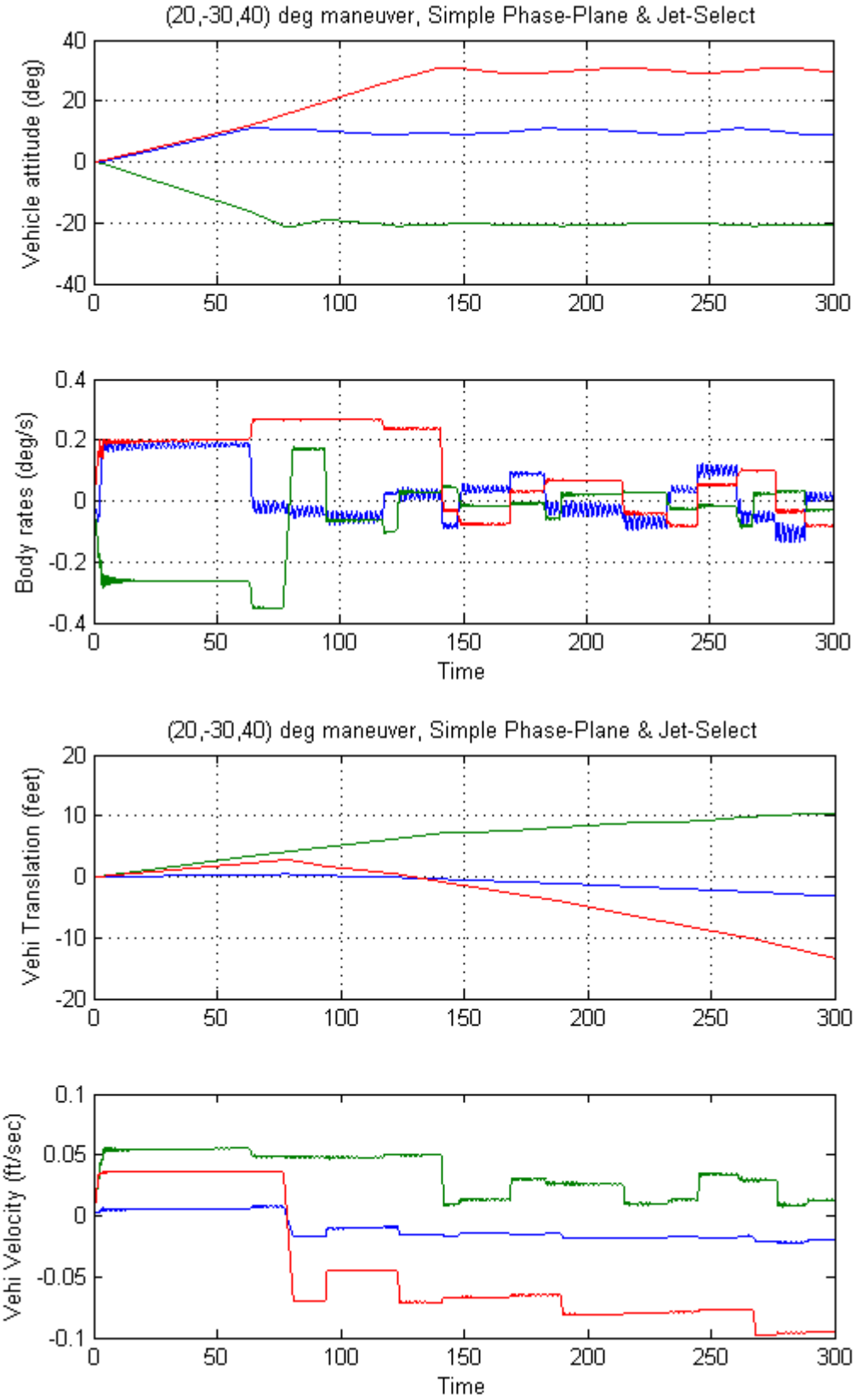


Figure 2.2.7 Discrete Spacecraft State-Space System from file "flex_satellite_fvp_z.m", system originated from the Flight Vehicle Modeling Program.

The jet-selection logic output in this model is a little different, because, it uses the FVP spacecraft system. This system requires the jet thrust inputs to vary between zero and one (one representing max thrust). The actual thrust value is integrated in the state-space system data. The file "start.m" initializes both models, and file "pl.m" plots the results from either simulation after completion.

The following plots show the spacecraft response to attitude commands. The attitude converges to its commanded position. The rates are limited to approximately 0.2 (deg/sec). It is significant to notice that at the end of the maneuver the vehicle linear velocity and position are not zero. This will be corrected later by including a translational control logic. The plots show that the responses of the two models to attitude commands are almost identical. There may be negligible differences between the two models in the rigid-body data.


(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehicle attitude (deg)
Body rates (deg/s)
Time
(20,-30,40) deg maneuver, Simple Phase-Plane & Jet-Select
Vehi Translation (feet)
Vehi Velocity (ft/sec)
Time

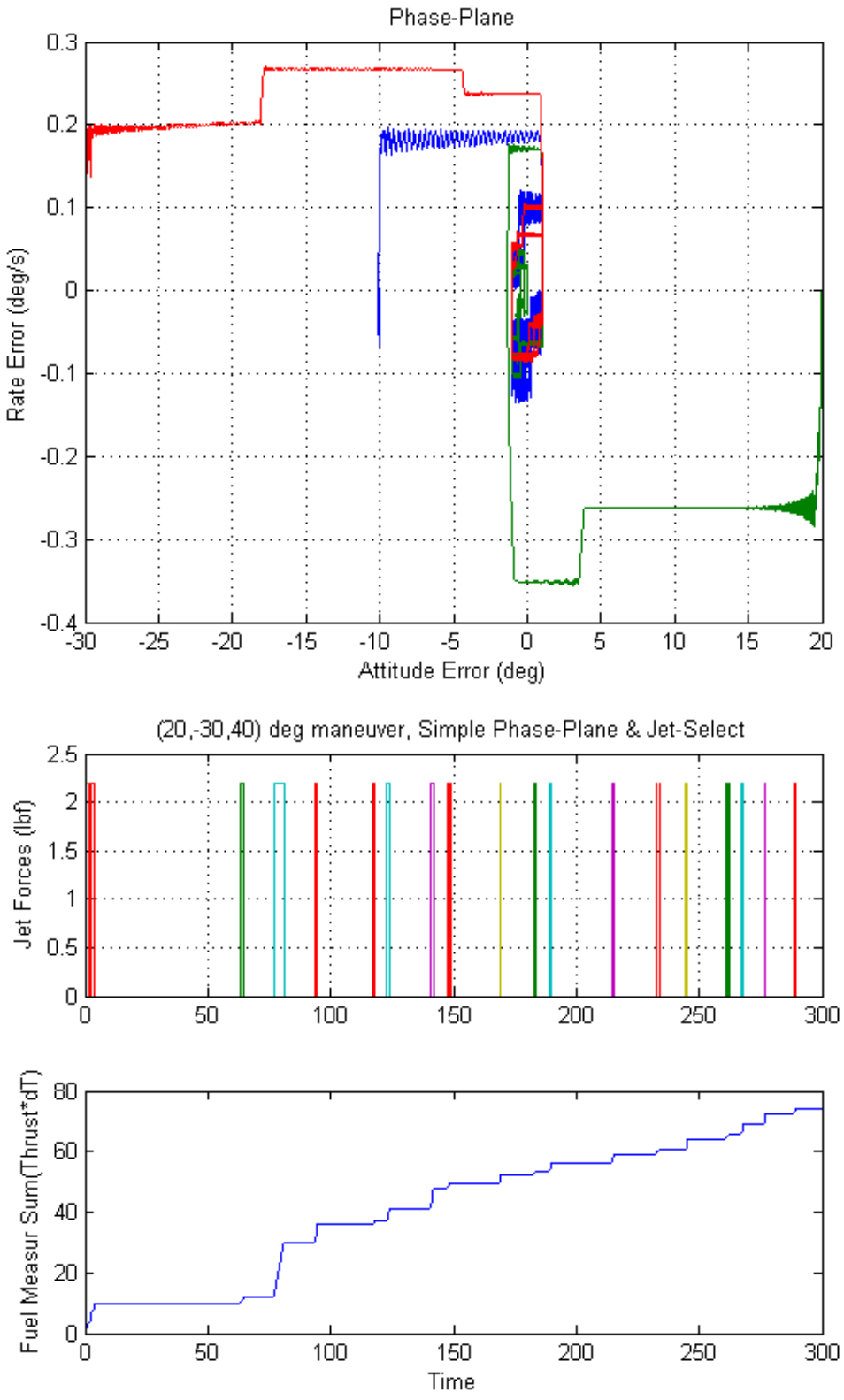
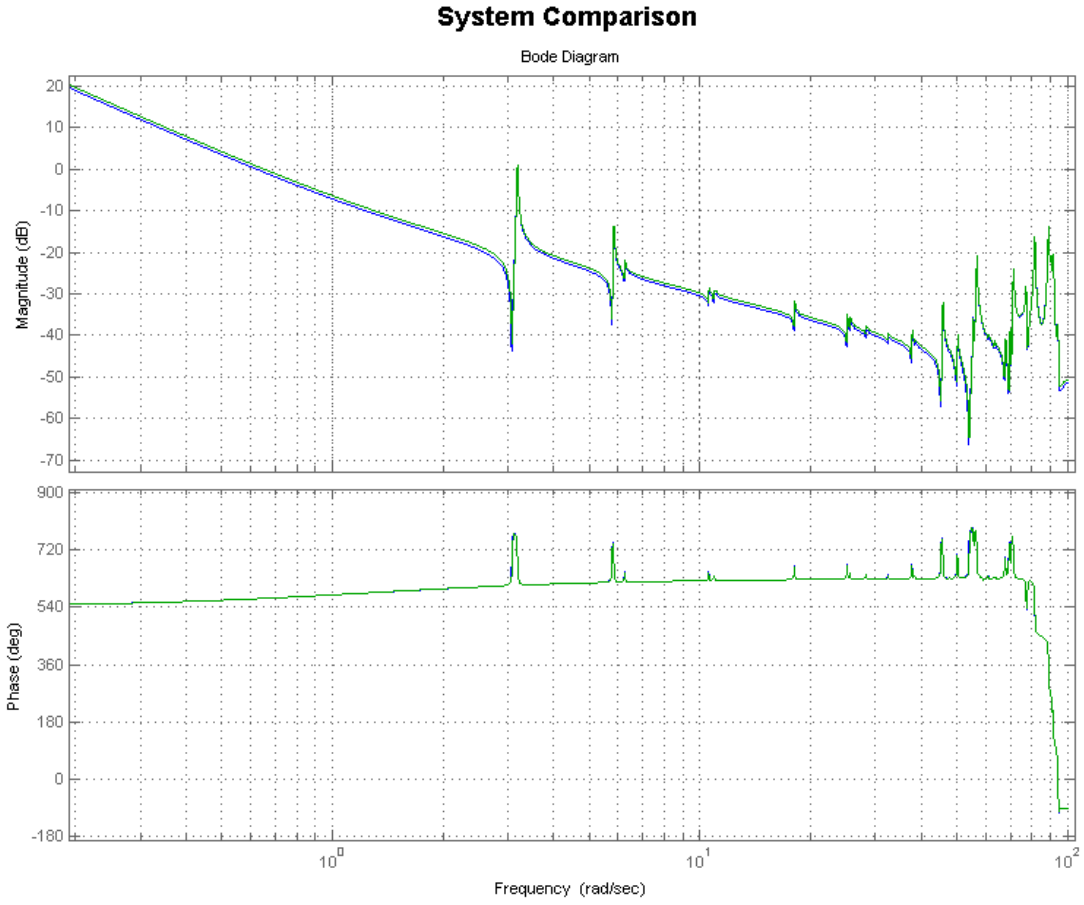


Figure 2.2.7 Simulation Results from "Sim_Flex_fvp_z.mdl"

System Comparison in the Frequency Domain

Move down to the subfolder "*Frequency Domain Comparison*" and run the Matlab file "*run_frequ.m*". This file loads the two spacecraft systems "*flex_spacecraft_fem_s.m*" and "*flex_spacecraft_fvp_s.m*" that were created using different methods and calculates the frequency responses of the open-loop systems including linearized controls. The frequency responses are shown plotted together in Bode and Nichols charts in Figure (2.2.8). The results are almost identical proving a very good match between the two modeling approaches. This is an encouragement to continue the analysis further.



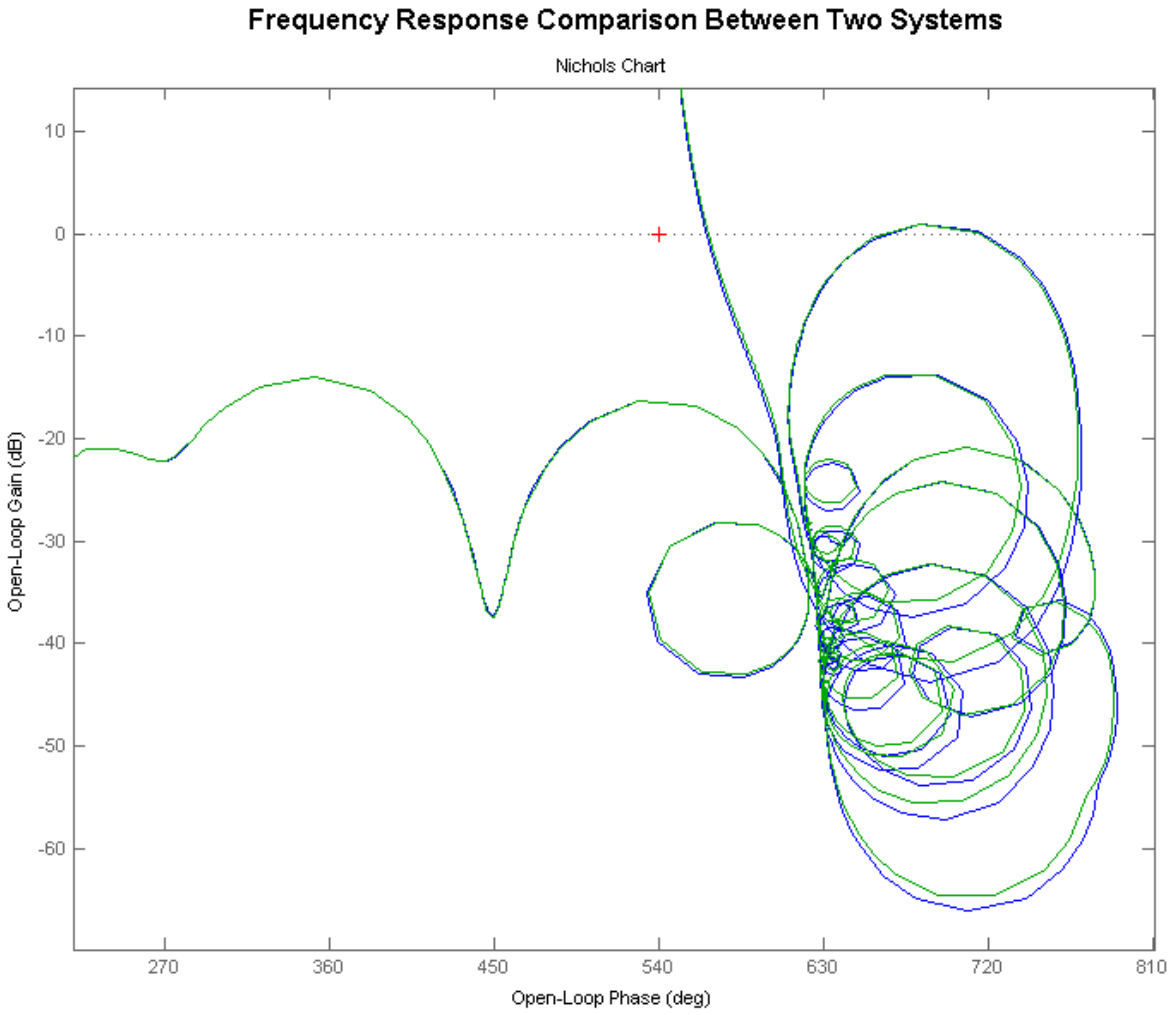


Figure 2.2.8 Frequency domain plots show a very good comparison between the two models

2.4 Jet Selection Logic that Minimizes Fuel Usage

A typical attitude control Reaction Control System (RCS) consists of the phase-plane logic and the jet selection logic. The phase-plane logic determines the direction where to apply a torque on the spacecraft in order to correct for attitude errors and it generates a rate change command to the jet selection logic. In a 12 jet, dot-product based, jet selection logic, 2 to 4 jets are selected which are contributing a positive torque in the demanded direction, and the logic fires those jets for a short period to correct the attitude error. The direction of motion, however, may be near but it is rarely along the commanded direction, which causes an error to develop in a different direction which corresponds to a new set of jets to be selected and fired in the next cycle. It is common sense, therefore, to assume that in order to rotate the spacecraft about a commanded direction more optimally in terms of fuel usage, after selecting the jets which contribute a positive torque in that direction, not all of them should be fired at the same level. Instead, if we assume that the flow rate is proportional to thrust, the ones which contribute more torque should be allowed to fire at higher thrusts than the ones which contribute less in that direction. But this is impossible because most reaction control jets can only fire at constant thrust. They are either "on" or they are "off." In this case we can use pulse width modulation. We assume that the control cycle period is relatively long enough (0.1 sec) to allow sufficient room for a jet pulse width modulation within the cycle. In the beginning of the control cycle we select 3 jets and turn them on together. Then we allow the most contributing jets to stay on longer within the cycle than the less contributing jets, an "on-time" proportional to the amount of contribution of the selected jets. The jet control logic commands the jets every 0.1 sec. When a jet is selected it receives also its "on-time", which is in multiples of 5 msec, a minimum of 5 msec, and a maximum of 95 msec. So the attitude control software does not have the responsibility to turn "off" the jets.

But how do we determine how long should each of the selected jets stay "on" during the control cycle? Let us assume that three jets out of 12 were selected (i, j, and k) and their corresponding vehicle acceleration vectors from each jet are: (a_i , a_j , a_k) respectively. If during the cycle we turn them on for a period of (t_i , t_j , t_k) respectively, and there are no other disturbances, at the end of the cycle the vehicle rate ($\delta\underline{\omega}$) will be

$$\delta\underline{\omega} = \begin{pmatrix} \underline{a}_i & \underline{a}_j & \underline{a}_k \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \end{bmatrix} = \left[A_{i,j,k}\right]\underline{t} \qquad (2.4.1)$$

Now if ($\delta\underline{\omega}_c$) represents the commanded change in vehicle rate at the end of the cycle, and that the vehicle does move much during the short cycle, then we can invert the matrix and solve for the on-times. We assume of course that the jets were properly selected for the commanded direction to provide positive on-times. Otherwise, it may result into negative on-times.

$$\underline{t} = \left[A^{-1}_{i,j,k}\right]\delta\underline{\omega}_c \qquad (2.4.2)$$

Assuming that (f_i , f_j , f_k) are the corresponding amounts of fuel flow rates for the three thrusters selected, the total amount of propellant used by jets (i, j, and k) to achieve a commanded change in rate ($\delta\underline{\omega}_c$) is

$$p_{i,j,k} = \begin{pmatrix} \underline{f}_i & \underline{f}_j & \underline{f}_k \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \end{bmatrix} = \begin{pmatrix} \underline{f}_i & \underline{f}_j & \underline{f}_k \end{pmatrix} \left[A^{-1}_{i,j,k} \right] \delta\underline{\omega}_c \qquad (2.4.3)$$

The propellant usage factor $p_{i,j,k}$ is the criterion used for selecting 3 jets from a total of 12 jets. The thrust directions of the jets on the spacecraft are assumed to be properly selected so that there are always 3 jets, at least, which provide a positive torque in any commanded direction. For a given change in rate command ($\delta\underline{\omega}_c$), the jet-select logic first chooses 3 jets that minimize equation (2.4.3) using function **Jet_Select_3dof**, and it calculates also the "on-times" from equation (2.4.2).

2.4.1 Upgrading the ACS Model with the Min Fuel Logic

The Simulink model "*Sim_Flex_3rot.Mdl*" shown in Figure (2.4.4) implements the minimum fuel attitude control logic. It can be found in folder "*\Flixan\ Examples\Flex Agile Spacecraft with SGCMG & RCS\Analysis* ***(e) Min Fuel RCS Control 3-Rot Flex***". It consists of the RCS attitude control system which operates at 0.1 sec sampling period, and the flexible spacecraft dynamics which is sampled every 5 msec. The roll, pitch, yaw attitude command is applied on the left side.

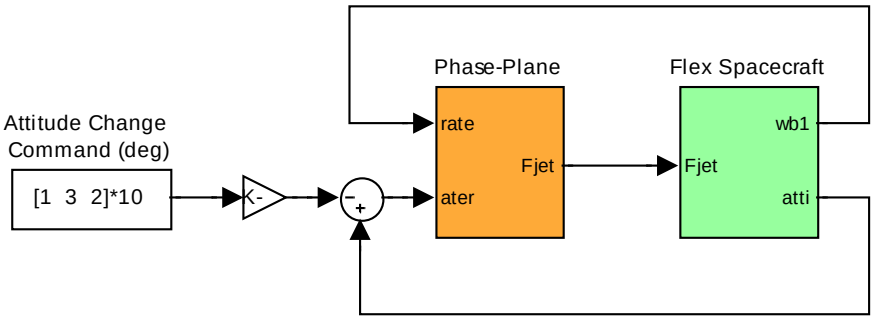


Figure 2.4.4 The 3-dof Simulation Model (Sim_Flex_7R.Mdl) for Min Fuel ACS

The orange ACS block is shown expanded in Figure (2.4.5). It consists of the phase-plane logic which is implemented in Matlab function "*Rotat_PP3.m*". It receives the attitude error and body rate signals. The phase-plane calculates the commanded change ($\delta\underline{\omega}$) in vehicle rate (roll, pitch, and yaw), that feeds into the jet selection logic, shown in Figure (2.4.6), which turns the thrusters "on" or "off" as needed to maneuver the vehicle. Since the system operates at two different rates, 0.005 sec and 0.1 sec, rate transitioning blocks are used in the interfaces between the Simulink blocks.

2.5 Extending the Min Fuel Idea to 6-dof Control

The attitude control logic that we have used so far completely ignores the translational position and velocity state of the spacecraft at the end of a maneuver. In some cases we may want to maneuver the spacecraft attitude while maintaining its current position and velocity, without any linear translation, or we may want to simultaneously command attitude and position changes, such as, for example, during docking. In this section we shall extend the min fuel jet selection logic to 6 degrees-of-freedom (6-dof) control for positioning the spacecraft in both rotational and translational directions. The extended 6-dof logic consists of two phase-planes, a rotational phase-plane similar to the one used in section 2.4, and a translational phase-plane that controls spacecraft linear position and operates very similar. The extended 6-dof jet-select logic receives ($\delta\underline{\omega}$) commands from the rotational phase-plane, and ($\delta\underline{v}$) commands from the translational phase-plane. Then it performs a search to identify 6 jets that provide maximum contribution towards the commanded rotational and translational directions combined.

Let us now assume that six jets out of 12 are selected (i, j, k, l, m, and n) and that their corresponding vehicle angular acceleration vectors from each jet are: (a_i , a_j , a_k , a_l , a_m , a_n) respectively. Also the translational acceleration vectors from each jet are: (b_i , b_j , b_k , b_l , b_m , b_n) respectively. The rotational and translational accelerations for a jet (i) are calculated by the following equations

$$\underline{a}_i = J_v^{-1}\left(\underline{d}_i \times \underline{u}_i\right) f_i \qquad \underline{b}_i = f_i \underline{u}_i \Big/ M_v$$

Where:

$\underline{d}_i$ is the moment arm vector of the thruster from the CG
$\underline{u}_i$ is the thruster direction unit vector
f_i is the jet thrust
M_v J_v is the vehicle mass and moment of inertia matrix

If we turn on the 6 selected jets together in the beginning of the control cycle period and leave them on for periods of (t_i , t_j , t_k , t_l , t_m , t_n) respectively, then when the longest firing jet is turned off the vehicle angular and translational rate ($\delta\underline{\omega}$, $\delta\underline{v}$)' is:

$$\begin{bmatrix} \delta\underline{\omega} \\ \delta\underline{v} \end{bmatrix} = \begin{pmatrix} \underline{a}_i & \underline{a}_j & \underline{a}_k & \underline{a}_l & \underline{a}_m & \underline{a}_n \\ \underline{b}_i & \underline{b}_j & \underline{b}_k & \underline{b}_l & \underline{b}_m & \underline{b}_n \end{pmatrix} \begin{bmatrix} t_i \\ t_j \\ t_k \\ t_l \\ t_m \\ t_n \end{bmatrix} = \left[A_{(6x6)}\right]\underline{t} \qquad (2.5.1)$$

If $(\delta\underline{\omega}$, and $\delta\underline{v})_c$ represent the commanded changes in vehicle angular and translational rate at the end of the 0.1 sec cycle, then we can solve for the on-times of the 6 jets by inverting the A matrix. We assume of course that the jets are properly selected for the commanded directions in order to provide positive on-times, otherwise, it will result into negative on-times.

$$\underline{t} = \left[A^{-1}_{(6x6)}\right]\begin{bmatrix}\delta\underline{\omega}\\ \delta\underline{v}\end{bmatrix}_c \qquad (2.5.2)$$

Assuming that (f_i , f_j , f_k , f_l , f_m , f_n) are the corresponding amounts of fuel flow rates for the six thrusters, the total amount of propellant used by the 6 jets to achieve the commanded ($\delta\underline{\omega}$, and $\delta\underline{v})_c$ is

$$p_{i,j,k,l,m,n} = \begin{pmatrix}\underline{f}_i & \underline{f}_j & \underline{f}_k & \underline{f}_l & \underline{f}_m & \underline{f}_n\end{pmatrix}\left[A^{-1}_{(6x6)}\right]\begin{bmatrix}\delta\underline{\omega}\\ \delta\underline{v}\end{bmatrix} \qquad (2.5.3)$$

The propellant usage factor $p_{i,j,k,l,m,n}$ is the criterion for selecting 6 jets from a total of 12 jets. We also assume that the thrust directions of the jets are properly selected so that there are at least 6 jets which provide a positive torque or force in any commanded direction. For a given commanded ($\delta\underline{\omega}$, and $\delta\underline{v})_c$, the jet-select logic first chooses 6 jets that minimize equation (2.5.3) using function **Jet_Select_6dof**, and it calculates also the "on-times" from equation (2.5.2).

2.5.1 Rotational plus Translational 6-dof Simulation Model, using the Minimum Fuel Jet Selection Logic

The files for this simulation model are in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis* ***(f) Min Fuel RCS Control 6-dof Flex***". The Matlab file "start.m" initializes the simulation parameters. The Simulink model that implements the fuel optimal 6-dof control logic is "*Sim_Flex-6dof.Mdl*" and it is shown in Figure (2.5.4). It consists of the rotational and translational phase-planes (implemented in functions "*Rotat_PP3.m*" and "*Translat_PP.m*") which operate at 0.1 sec and generate the commands ($\delta\underline{\omega}$, and $\delta\underline{v})_c$ driving the 6-dof jet selection logic. The jet selection logic calls functions "*Jet_main_6dof.m*", "*Jet_Select_6dof.m*", "*Jet_Select_rotat*" and "*Jet_Select-transl*" which select 6 jets and their corresponding "on-times", for each 0.1 sec cycle. The jet forces drive the flex spacecraft dynamics (green block) which is the system "*flex_spacecraft_fvp_z.m*" used earlier and sampled at 5 msec. The inputs to the phase-planes are attitude and translation delta commands.

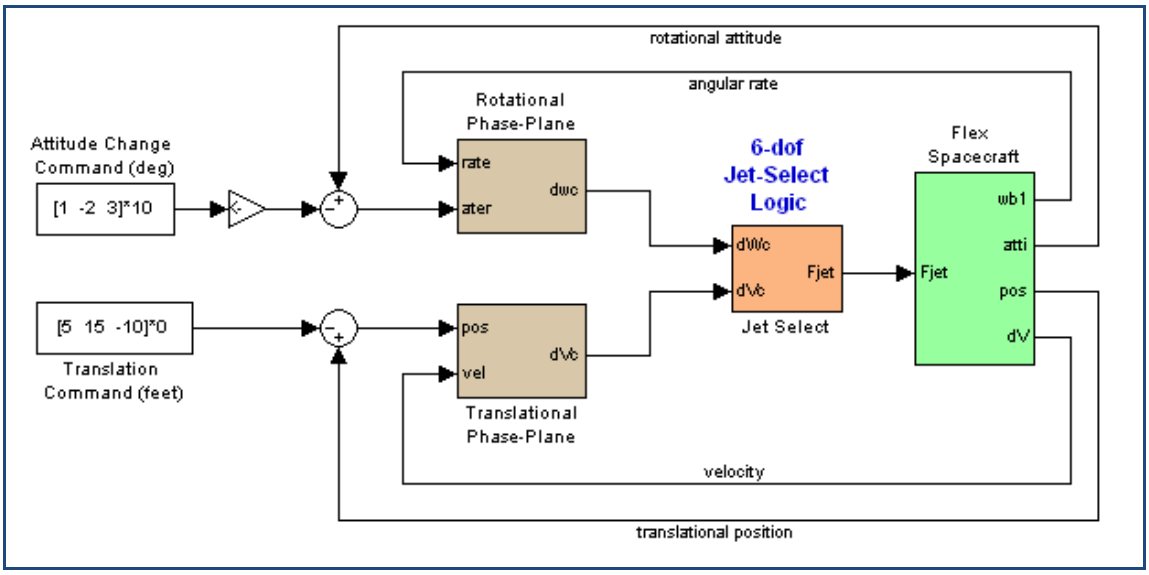


Figure 3.5.4 Minimum Fuel 6-dof Simulation Model "Sim_Flex_6dof.mdl"

In this configuration the RCS controls the vehicle in all 3-translational and 3-rotational directions. The rotational phase-plane issues a change in body rate command and it is implemented in Matlab function "*Rotat_PP3.m*" which. The translational phase-plane issues a change in velocity command and it is implemented in Matlab function "*Translat_PP.m*". Figure (2.5.5) shows the Simulink implementation of the jet selection logic.

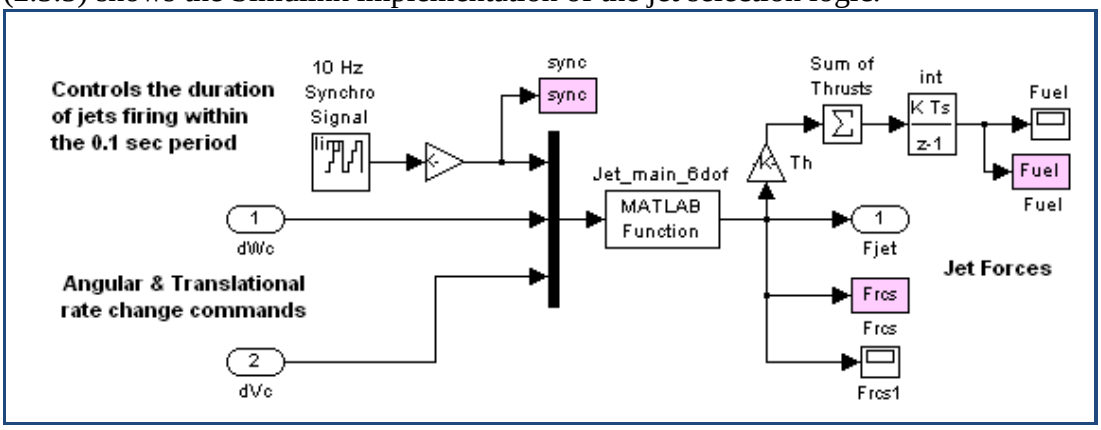


Figure 2.5.5 Minimum Fuel 6-dof Jet Selection Logic

Inside the jet select logic, the change in rate and velocity commands are inputs to Matlab function f= **Jet_main_6dof**(sync, δω, δv), which calculates the jet forces f(i). This function calls the function [js, t]= **Jet_Select_6dof**(δω, δv) which performs a jet search, and selects 6 among the 12 jets that minimize the fuel equation (2.5.3), where (δω, δv) are the change in vehicle rate and velocity commanded by the phase planes. The logic selects 6 jets that best satisfy the combined translational and rotational phase-plane demands. The selected jet numbers are given in the array js(.), and the corresponding on-times in array t(.). The input "sync" is a saw-tooth signal that synchronizes the jet turning-off times in the simulation, not in the actual hardware. In the beginning of the control cycle all 6 selected jets are turned "on". The logic turns them "off"

2.6 Modeling the Fuel Motion inside the Tank

There is a tank inside the spacecraft that has a spherical shape and contains fuel for the RCS jets. The spacecraft motion causes the fuel to move inside the tank and it is creating disturbance forces on the spacecraft which has the potential to interfere with operations or even cause instability in the control system. There is a need, therefore, to capture the fuel sloshing dynamics in a mathematical model that can be combined with the spacecraft model. Linear spring-mass or pendulum models typically used in launch vehicles are not applicable here because the spacecraft is at zero or very low g and the fuel is not behaving like a linear pendulum. It is reasonable, therefore, to assume that the sloshing fuel will induce a bigger disturbance on the spacecraft when it is lumped together like a soft mass m_s that can slide or bounce against the inner surface of the tank as the vehicle translates and rotates in space, and not when it is spread thin inside the tank.

There are a couple of conceptual models to capture the lumped fuel motion and its reaction forces on the tank, both leading to the same equations. One model assumes that the slosh mass m_s is soft and it behaves like a 3-dimensional elastic pendulum. The mass is attached to a tether, and the other end of the tether is attached to the center of the tank, as if as there is a hook at the center of the tank. To capture the elasticity of the mass we insert a spring between the mass and the tether end. When the string is stretched the reaction forces on the vehicle are applied at the tank center through the tether. The soft mass either slides around the inside surface of the tank with the string stretched or it floats inside the tank when the string is slack, in which case it does not apply any force on the tank. When the mass distance from the tank center exceeds the length of the tether (r) the spring stretches and applies a force at the center through the tether. In another visualization the mass softly splashes against the inner surface of the tank without disintegrating.

There is one additional feature needed to prevent excessive deflections of the slosh mass and to contain it within the walls of the tank. We use a non-linear spring that has a stiffness coefficient $k_s(\delta)$ increasing parabolically with extension (δ), that is $k_s = c_1\delta^2 + c_2$. As the mass approaches the tank surface the radial string force becomes sufficiently high to constrain the mass within the tank walls. So the soft mass can either slide parallel to the surface, or float inside the tank, or bounce against the surface. It can be pictured as shown in Figure (2.6.1), where the fixed length of the string is r, where r is about half the size of the tank radius r_0, and there is a non-linear spring between the end of the string and the center of slosh mass m_s.

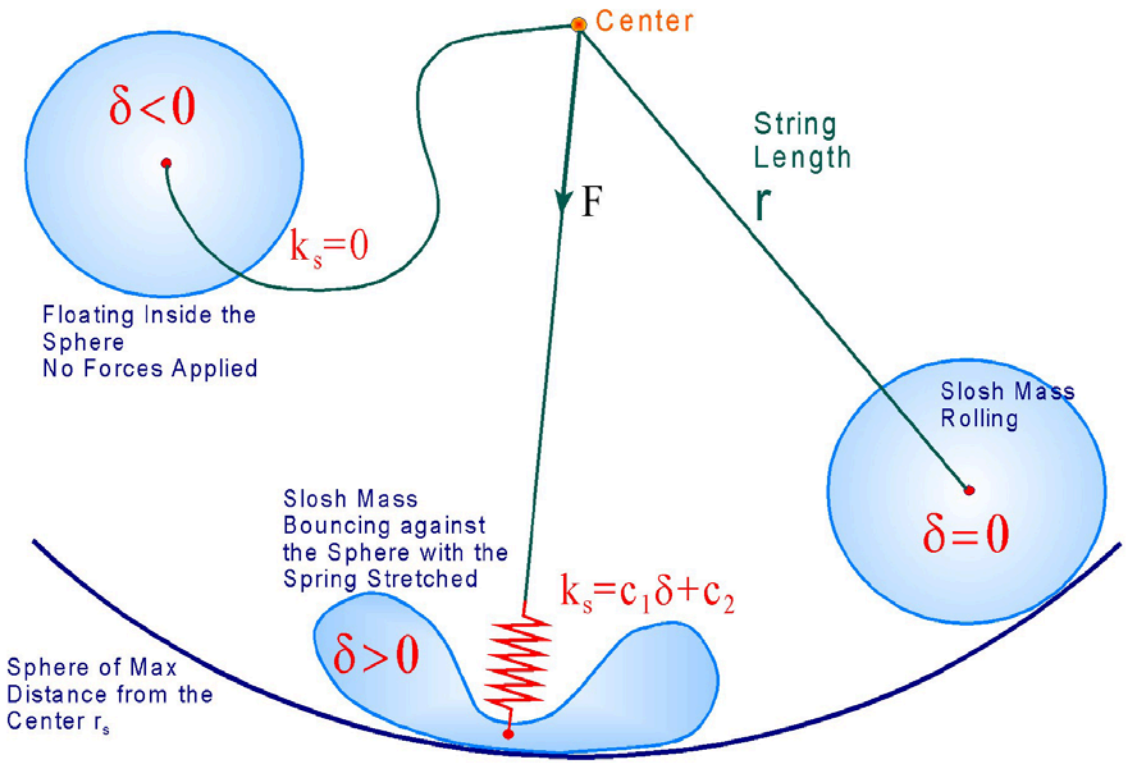


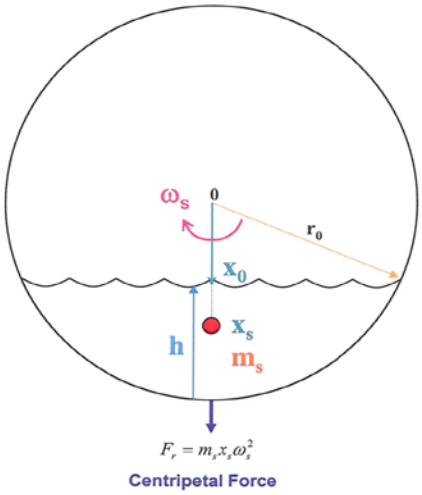
Figure 2.6.1 Conceptual Illustration of the Non-Linear Slosh Pendulum Model

Note, the model parameters, such as the axial and tangential damping coefficients, the length of the tether (r), and the non-linear spring constants are derived from data derived from computational fluid dynamics models. The parameters are adjusted to match the oscillations frequency and rate of decay of the CFD responses. The non-linear spring parameters c_1 and c_2 are adjusted to capture a realistic behavior of the liquid mass as it hits a surface of the tank due to vehicle accelerations.

Another conceptual model that captures the same dynamic effect is to assume that the slosh mass behaves like a soft ball, rolling and bouncing inside a sphere which is approximately half the tank radius, applying forces on the tank perpendicular to the tank surface. Since the tank is spherical the reaction forces always pass through the tank center, same as the soft pendulum model.

2.6.1 What is the worst fuel level?

Before we start analyzing the sloshing problem we must determine: what is the fuel level where the slosh disturbance on the vehicle is maximized. When the tank is almost empty by common sense we know that the sloshing forces are negligible. Also, when the tank is full there is no fuel motion and therefore there is no disturbance. The answer is obviously somewhere in between and we must, therefore, determine what fuel level maximizes the slosh disturbance and use that fuel level in the analysis in order to be on the conservative side.



Let us assume a spherical tank of radius r_0, volume V_0, holding a total fuel mass M_0. The fuel is rotating around the inside surface at an angular rate ω_s held together under the influence of the centripetal force. Its density is (ρ), where:

$$\rho = \frac{M_0}{V_0} = \frac{3M_0}{4\pi r_0^3} \qquad (2.6.1)$$

Assuming that the fuel surface is almost flat, let us calculate the fuel volume as a function of fuel level height (h)

$$V(h) = \pi \int_0^h \left(r_0^2 - x_0^2\right) dh = \pi \int_0^h \left(2r_0 h - h^2\right) dh$$

$$V(h) = \pi h^2 \left(r_0 - \frac{h}{3}\right) \qquad (2.6.2)$$

The fuel mass can be calculated as a function of the fuel surface height or as a function of the surface distance from the tank center x_0.

$$m_s = \pi\rho h^2 \left(r_0 - \frac{h}{3}\right) = \pi\rho \left(\frac{2}{3} r_0^3 + \frac{1}{3} x_0^3 - r_0^2 x_0\right) \qquad (2.6.3)$$

We can also derive equations for the tank fill ratio f_r as a function of fuel height (between zero and one) and the pendulum length l_p between the center of the tank and the fuel center of mass.

$$f_r = \frac{h^2(3r_0 - h)}{4r_0^3} \qquad l_p = \frac{(2r_0 - h)^2}{4\left(r_0 - \frac{h}{3}\right)} \qquad (2.6.4)$$

It seems reasonable to assume that the disturbance on the vehicle will be maxed when the slosh moment of the liquid from the tank center is maximized. The slosh moment is equal to the liquid mass times the distance of its center of mass from the tank center.

$$m_s x_s = \int_{x_0}^{r_0} x\,dm = \pi\rho \int_{x_0}^{r_0} x\left(r_0^2 - x^2\right)dx$$
$$m_s x_s = \frac{\pi\rho}{4}\left(r_0^2 - x_0^2\right)^2 \qquad (2.6.5)$$

The disturbance on the vehicle is maximized when

$$\frac{d}{dx} moment = x_0\left(x_0^2 - r_0^2\right) = 0 \qquad (2.6.6)$$

This happens when x_0=0, that is, when the tank is half full. By combining equations (2.6.3) and (2.6.5) we can solve for the slosh mass distance from the tank center x_s

$$x_s = \frac{\left(r_0^2 - x_0^2\right)^2}{4h^2\left(r_0 - \frac{h}{3}\right)} \qquad (2.6.7)$$

As an alternative approach to calculating max disturbance conditions, let us assume that the liquid mass is spinning inside the tank. It is obvious to assume that the disturbance on the vehicle is maximized when the moment of inertia (I_{slosh}) of the liquid mass about the center of the tank is maximized, where:

$$I_{slosh} = \pi\rho \int_{x_0}^{r_0} x^2\left(r_0^2 - x^2\right)dx = \pi\rho\left[\frac{r_0^2 x^3}{3} - \frac{x^5}{5}\right]_{x_0}^{r_0}$$
$$I_{slosh} = \pi\rho\left[\frac{2r_0^5}{15} + \frac{x_0^5}{5} - \frac{r_0^2 x_0^3}{3}\right] \qquad (2.6.8)$$

The slosh moment of inertia is maximized when

$$\frac{d}{dx} I_{slosh} = x_0^2\left(x_0^2 - r_0^2\right) = 0 \qquad (2.6.9)$$

This happens again when x_0=0, and the fuel height $h = r_0$, that is, the worst disturbance condition is when the tank is half full. In this case the pendulum length, or the slosh mass distance from the center from equation (2.6.7) is

$$r = \frac{3}{8} r_0 \qquad (2.6.10)$$

In our subsequent slosh analysis, therefore, we shall assume that the fuel tank is half full, and the sloshing pendulum mass is equal to half of the full tank mass, and the soft pendulum length (r) is only 3/8 of the actual tank radius (r_0).

2.6.2 Zero-g Non-Linear Slosh Model

The acceleration ($\underline{a}_t$) of the spacecraft at the center of the tank is

$$\underline{a}_t = -\underline{d}_s \times \dot{\underline{\omega}}_b + \underline{a}_s \quad where: \ \underline{d}_s = \underline{l}_{tnk} + \underline{x}_s - \underline{l}_{CG}$$

Where:

$\underline{d}_s$ is the distance of the slosh mass from the spacecraft CG,

$\dot{\underline{\omega}}_b$ is the spacecraft angular acceleration,

$\underline{a}_s$ is the spacecraft translational acceleration at the CG.

When the pendulum string is stretched, the tension at the string F_{st} can be expressed by the following equation

$$\begin{aligned} F_{st} \quad &= m_s(r+\delta)\dot{\theta}^2 && centripetal\ force\ due\ to\ angular\ rate\ of\ mass \\ &+k_s(\delta)\delta && non-linear\ spring\ force\ due\ to\ deflection\ \delta \\ &+k_{d1}(\dot{\underline{x}}_s \bullet \underline{u}_1) && axial\ viscous\ friction \end{aligned}$$

Where:

$\dot{\theta}$ is the angular rate of the pendulum relative to tank,

$k_s(\delta)$ is the non-linear spring constant of the sluggish mass which is a function of spring displacement (δ), $k_s = c_1\delta^2 + c_2$

k_{d1} is the axial damping coefficient

$\dot{\underline{x}}_s$ is the slosh mass velocity relative to the tank, dotted with

$\underline{u}_1$ which is the unit vector from the tank center to the slosh mass

$$\underline{u}_1 = \frac{\underline{x}_s}{|\underline{x}_s|}$$

The inertial acceleration of the slosh mass consists of two components, the acceleration of the mass relative to the tank $\ddot{\underline{x}}_s$, plus the inertial acceleration of the tank $\underline{a}_t$, and it is the result of axial forces from the string along $\underline{u}_1$, plus viscous forces as it rotates around rubbing against the inside of the tank along $\underline{u}_2$.

$$m_s(\ddot{\underline{x}}_s + \underline{a}_t) = -F_{st}\underline{u}_1 - (k_{d2}\dot{\theta})\underline{u}_2$$

Where:

k_{dv} is the tangential damping coefficient of the mass as it slides along the surface creating a force parallel to the surface resisting the mass motion along $\underline{u}_2$

$\underline{u}_2$ which is the unit vector parallel to the surface in the velocity direction

$$\underline{u}_v = \frac{\dot{\underline{x}}_s}{|\dot{\underline{x}}_s|} \qquad \underline{u}_2 = [(\underline{u}_1 \times \underline{u}_v) \times \underline{u}_1]$$

The disturbance force on the spacecraft $F_{s/c}$ is equal and opposite to the force on the slosh mass and the torque on the spacecraft is obtained by cross-multiplying $F_{s/c}$ with the distance d_s of the slosh mass from the spacecraft CG.

$$\underline{F}_{s/c} = F_{st}\underline{u}_1 + \left(k_{d2}\dot{\theta}\right)\underline{u}_2$$
$$\underline{T}_{s/c} = \left(\underline{d}_s \times \underline{F}_{s/c}\right) \; where: \underline{d}_s = \underline{l}_{tnk} + \underline{x}_s - \underline{l}_{CG}$$

The velocity and position of the slosh mass m_s with respect to the tank are obtained by integrating the slosh mass acceleration starting from velocity and position initial conditions v_0 and p_0

$$\underline{\dot{x}}_s = \underline{v}_0 + \int_0^t \underline{\ddot{x}}_s(t)\,dt \qquad \underline{x}_s = \underline{p}_0 + \int_0^t \underline{\dot{x}}_s(t)\,dt$$

The slosh mass angular rate $\dot{\theta}$ is obtained by resolving the slosh mass velocity $\underline{\dot{x}}_s$ along the $\underline{u}_2$ direction. The pendulum angle θ is a function of the x-direction component of vector u_1.

$$\dot{\theta} = \left(\frac{\dot{x}_s}{r+\delta}\right) \bullet \underline{u}_2 \qquad \theta = \cos^{-1}\left[u_1(1)\right]$$

2.6.3 Implementing and Testing the Zero-g Slosh Model alone

Before coupling the slosh equations with the spacecraft dynamic model we are going to create a separate zero-g pendulum slosh model "*Slosh.mdl*", shown in Figure (2.6.2). This system will be used to test the slosh dynamics alone under the influence of external forces, starting from an initial condition of m_s. The slosh equations are implemented in Matlab function "*Slosh_0g.m*". The inputs to the Simulink model are: spacecraft rotational acceleration vector, and translational acceleration vector at the CG, coming from the vehicle dynamic model. The spacecraft accelerations induce forces on the mass and opposite reaction forces on the vehicle. The slosh mass acceleration relative to tank $\underline{\ddot{x}}_s$ is integrated twice to calculate the mass velocity and position relative to the tank. The model output is the reaction force vector which is applied to the vehicle model at the center of the tank. The forces are applied only when the spring is stretched, that is, $\delta > 0$. The file "*start.m*" loads the tank mass properties and initializes the mass position and velocity relative to the tank. The file "*plsl.m*" plots the simulation parameters.

2.7 Minimum Fuel RCS 6-dof Simulation with Fuel Sloshing and Flexibility at Zero-g

Now that we have developed our rigid-body and flex spacecraft models, and we have successfully analyzed the fuel minimization algorithm in both rotational and translational directions, and also tested the zero-g slosh model, the next step is to integrate all these models together in a 6-dof simulation that will be used to analyze the simultaneous, attitude and translation control while sloshing. Actually, we are going to develop two integrated models, a linear, and a non-linear model for comparison. The simulation files in this analysis are located in folder "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis***(h) NonLin RB+Slosh+Flex 6-dof RCS Attitude Control*". The file "*start.m*" initializes the spacecraft parameters. The slosh mass is initialized with an initial position x_{s0}, and velocity x_{sd0} relative to the center of the tank.

2.7.1 Linear 6-dof Model with Slosh and Flex

The linear Simulink model is in file "*Sim_Lin_Flex_Slo_6dof.mdl*", and shown in Figure (2.7.1). For spacecraft dynamics it uses the discrete state-space model file "*flex-spacecraft-fem_z.m*", title: "*RB+Flex Spacecraft with RCS and CMG (Z-Transf)*", which has 6 rigid-body modes and 40 flex modes, and was created using the flex spacecraft modeling program.

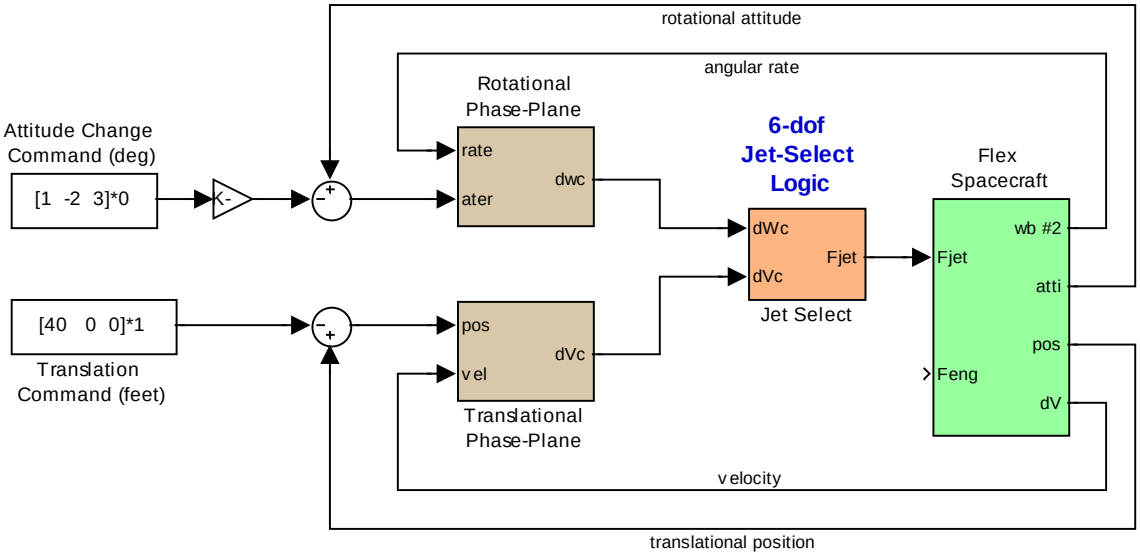


Figure 2.7.1 Min Fuel 6-dof Simulation Model "Sim_Lin_Flex_Slo_6dof.mdl" that includes Slosh and Flexibility

The model uses the two separate rotational and translational phase-planes and the 6-dof fuel minimizing jet selection logic that was described in Section (2.5). The simulation is running at 5 msec, and the phase-planes are running at 100 msec. Figure (2.7.2) shows the spacecraft model

2.7.2 Non-Linear 6-dof Model with Slosh and Flex

The non-linear Simulink model is in file "*Sim_NonLin_Flex_Slo_6dof.mdl*", shown in Figure (2.7.3). This model uses quaternion for attitude control and calculates a quaternion error for attitude feedback. The quaternion command (left) is calculated by combining the commanded angle of rotation and the rotation axis, see Figure (2.7.4). The attitude feedback signal (q_e) consists of the 3-dimensional vector part of the quaternion error. For small angles q_e= half the attitude error vector.

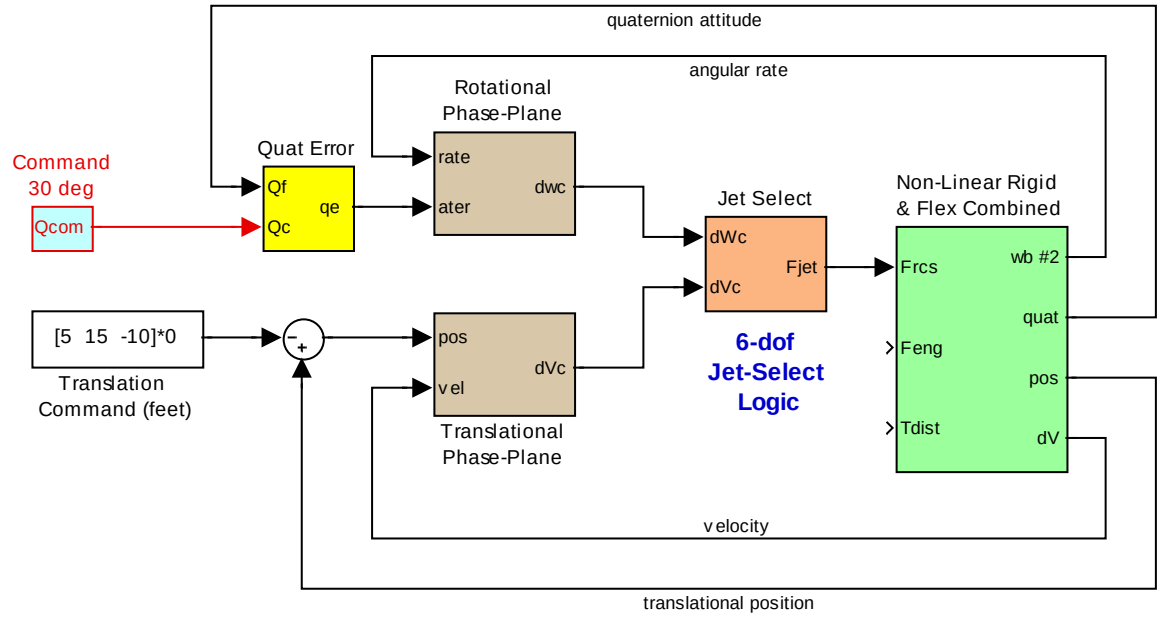


Figure 2.7.3 Min Fuel 6-dof Simulation Model "Sim_NonLin_Flex_Slo_6dof.mdl" that includes Slosh and Flexibility

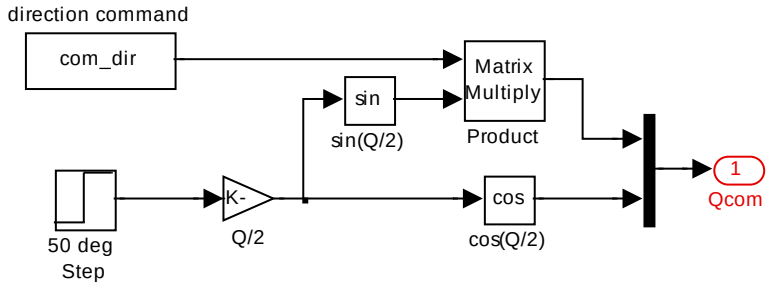
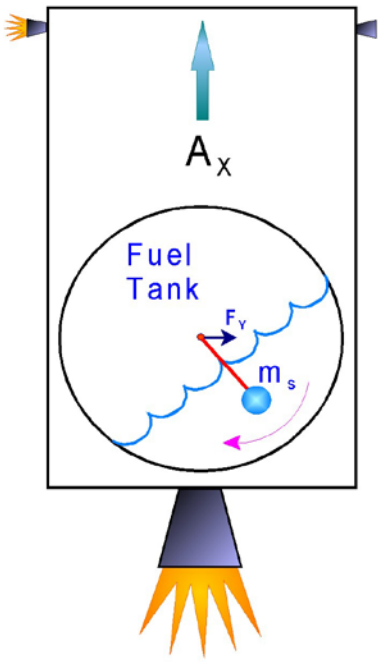


Figure 2.7.4 Quaternion Command Generator

The spacecraft dynamics is shown in Figure (2.7.5). It consists of two dynamic systems in parallel with the slosh model in the feedback path. It uses the 6-dof non-linear rigid-body dynamics which is implemented in Matlab function "*RB_Dynamics.m*". From the forces and moments this function calculates the velocities, angular rates, and integrates the quaternion from

2.8 Reboost Analysis with the Main Engine Firing



During reboost the spacecraft fires the main engine to modify its altitude or change orbit. The ACS points the spacecraft in a proper orientation, the main engine ignites, and the RCS attempts to keep it at constant attitude or a slowly varying attitude during the orbital maneuver. We assume of course that the RCS can provide enough torque to correct any attitude errors that may occur due to misalignment of the thrust vector from the spacecraft CG. The translation control system is obviously turned off during this phase since we are constantly accelerating. The acceleration causes the fuel to accumulate towards the engine at the bottom of the tank, and any lateral disturbance causes sloshing which generate oscillatory disturbances on the spacecraft. The fuel dynamic behavior resembles that of a simple pendulum. When the desired orbit is reached the main engine is turned off and the ACS switches to a different mode of operation. In this section we will analyze a couple of accelerating reboost models: a linear model that includes a linear pendulum slosh model, and a non-linear model coupled with slosh and structural flexibility. The simulation files for this analysis are in folder: "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\Reaction Control System Analysis**(i) Orbital Maneuvering-Reboost Phase". The file "*start.m*" initializes both simulation models.

2.8.1 Linear Reboost Simulation Model

The linear Simulink model for the reboost phase uses the state-space system that was generated using the Flixan "Flight Vehicle Modeling Program" in Section 1.2. Its title is "*Flexible Agile Spacecraft, Reboost Model (Z-Transf)*" and it was saved in file "*reboost_fvp_z.m*". In this case we are not using the complex zero-g slosh model wrapped around the spacecraft dynamics as in the previous zero-g case. This system is slightly different from the zero-g version used earlier because in this accelerating case the slosh dynamics simplifies to a linear pendulum resonance that can be included inside the vehicle state-space system. The pendulum frequency and other parameters are defined in the vehicle input data file. Also, in this case, the vehicle acceleration was set to 0.5 (ft/sec^2) in the x direction to capture the constant firing of the reboost engine. Otherwise, the program will not accept a slosh resonance when the vehicle acceleration is zero. Note, that in the vehicle data the slosh frequency is specified at 1g, (32.2 ft/sec^2) because it is usually known at 1g. The program adjusts the slosh frequency according to the total linear acceleration.

The simulation model for this linear case is in file "*Sim-Lin-Reboost-fvp.mdl*", shown in Figure (2.8.1). It uses the fuel optimal attitude control logic described earlier. There is no translation control during reboost. The flex spacecraft system with slosh is in Figure (2.8.2). The simulation results are shown in Figure (2.8.3). The spacecraft is commanded to maintain a constant zero

attitude during reboost. The engine thrust is constant throughout the simulation at 110 (lb). There is a repetitive RCS jet firing for counteracting the torque generated due to engine and CG misalignment. The RCS is holding the spacecraft attitude error below 0.5 (deg) defined by the dead-band.

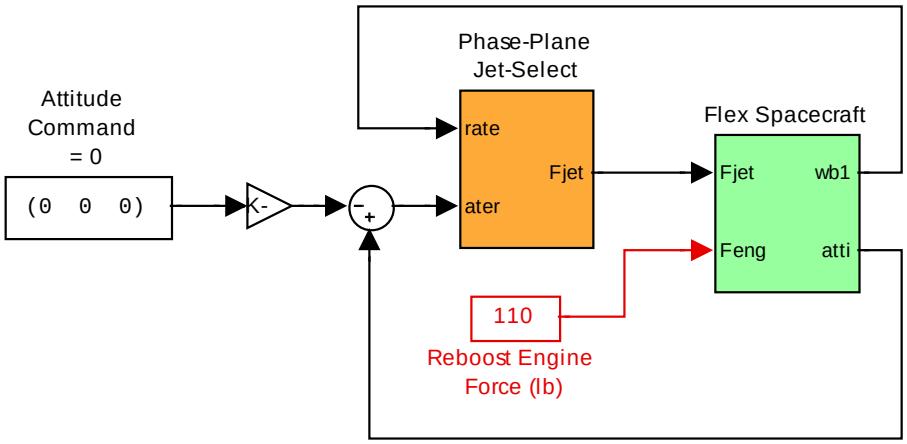


Figure 2.8.1 Linear Reboost Simulation Model "Sim_Lin_Reboost_fvp.mdl"

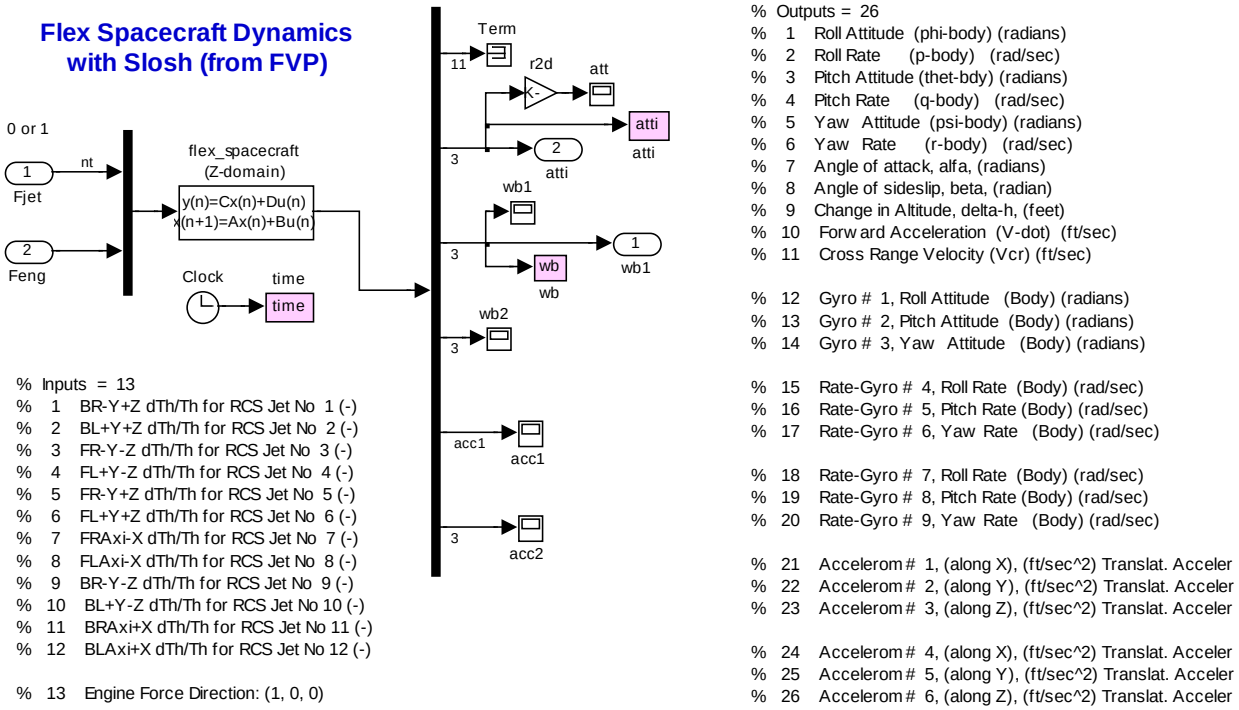


Figure 2.8.2 Spacecraft dynamic model with slosh created using the FVP

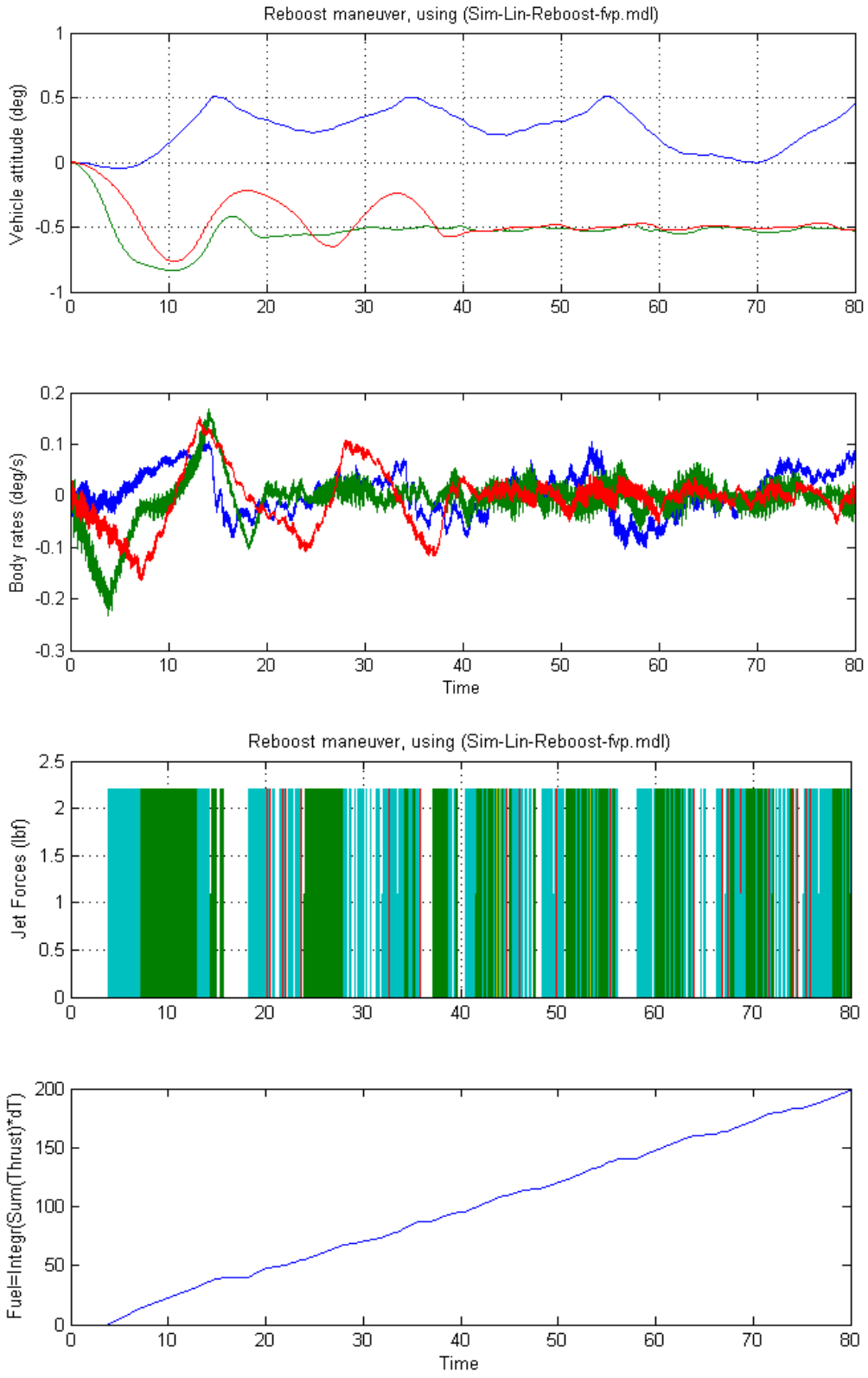


Figure 2.8.3 Simulation Results from the linear reboost system with slosh

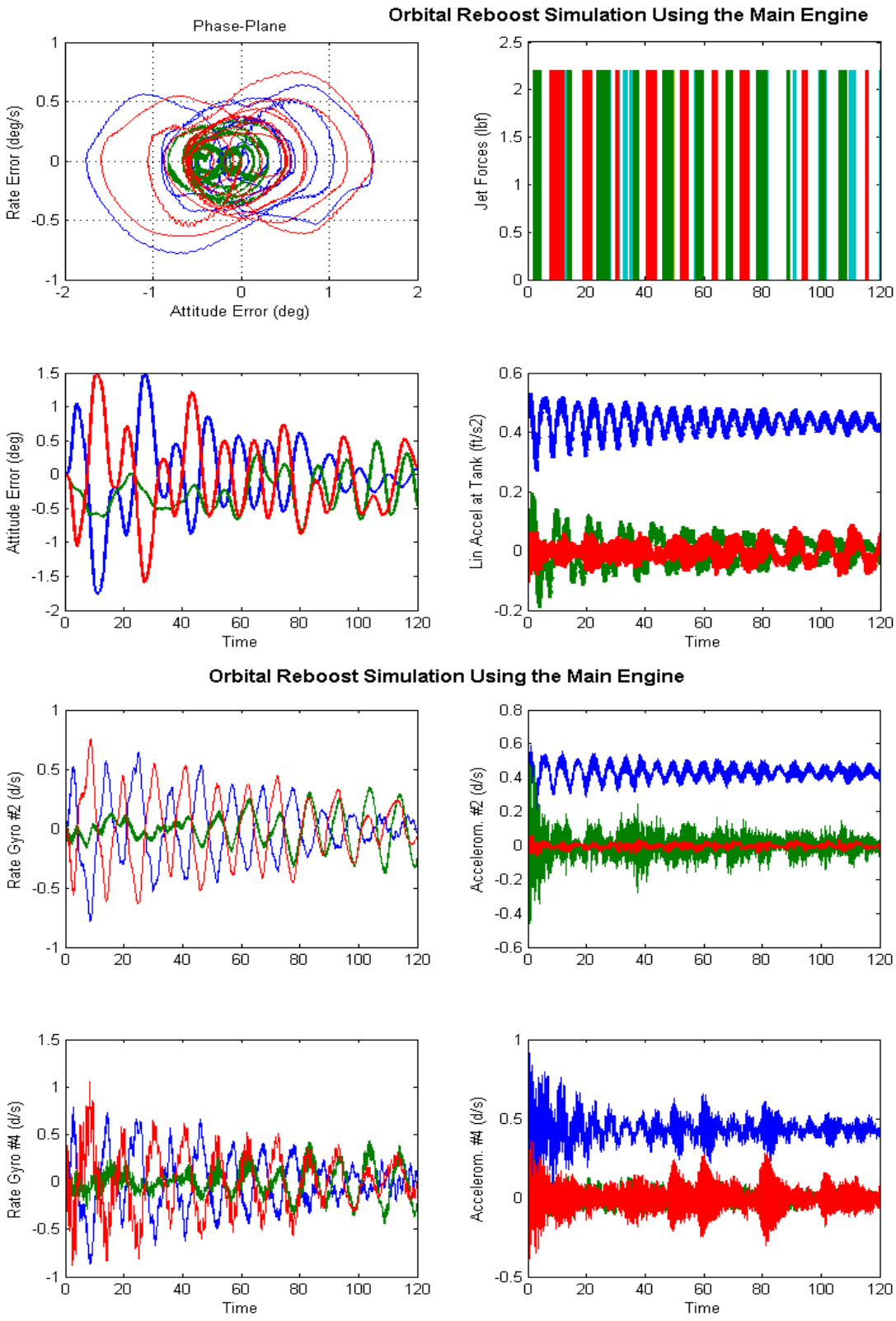


Figure 2.8.5 Non-Linear Reboost Simulation, Vehicle Response in Various Locations

2.8.3 Describing Function Analysis of the RCS during Reboost

The Describing Function (DF) is a very powerful frequency domain method for evaluating the stability margin of a non-linear control system, determining the existence of limit cycles, and also for designing filters to attenuate resonances and to shape the control system's frequency response in order to improve stability margin and to prevent limit-cycles. Limit cycles are sustained oscillations caused by non-linearities. We are not concerned with rigid-body limit-cycles because there is always going to be some degree of rigid-body limit-cycling. We are more concerned with limit-cycles which are caused due to structure flexibility excitation. It is very important, therefore, to achieve sufficient gain and phase margin in flex mode resonances in order to prevent structural limit-cycling because they are very undesirable. They use up a lot of fuel, corrupt measurements, spacecraft performance, and they cause structural damage due to metal fatigue. In the classical Describing Function methodology we separate a single-input-single-output (SISO) control systems in two parts: (a) the linear G(s) part which is a function of frequency (ω), and (b) the non-linear N(e) part which is a function of the error signal amplitude (e). Then we solve the feedback equation $G(j\omega)\,N(e)=1$ by using Nichols or Nyquist diagrams. Intersections of the G(jω) locus with the -1/N(e) locus indicate the possibility of limit-cycles, but not every intersection defines a sustained oscillation. There are convergent limit-cycles and divergent limit-cycles, and they usually alternate. We are not concerned with divergent limit-cycles because they do not sustain an oscillation. We are only concerned with the convergent limit-cycles. The convergent limit-cycle amplitude and frequency can be obtained approximately from the loci intersections. The amplitude is obtained from the 1/N(e) locus, and the frequency is obtained from the G(jω) locus which are both co-plotted on a Nichols or a Nyquist plot.

It is not straightforward, however, how to apply the DF method in a typical RCS phase-plane system and requires some simplifications and assumptions. The classical DF method is applicable only to SISO systems, and it assumes that the DF of the non-linearity N(e) is a SISO, and a function of only amplitude but not frequency. In our situation, however, the control system structure is a little different and requires some assumptions to be made and block diagram manipulations in order to shape it in a form where the DF method can be applied. To start with, our phase-plane controller is unconventional because it has two inputs and one output. If we assume, however, that during limit-cycling the inputs are approximately sinusoidal, the two inputs: rate and attitude errors are related because the attitude error is the integral of the rate, hence, we can reduce the phase-plane inputs to only one input, the body rate and integrate the rate to get attitude. Any sustained limit-cycle generates a pattern in the phase-plane symmetric about the origin, producing, therefore, a periodic output torque with zero average, as shown in Figure (2.8.6).

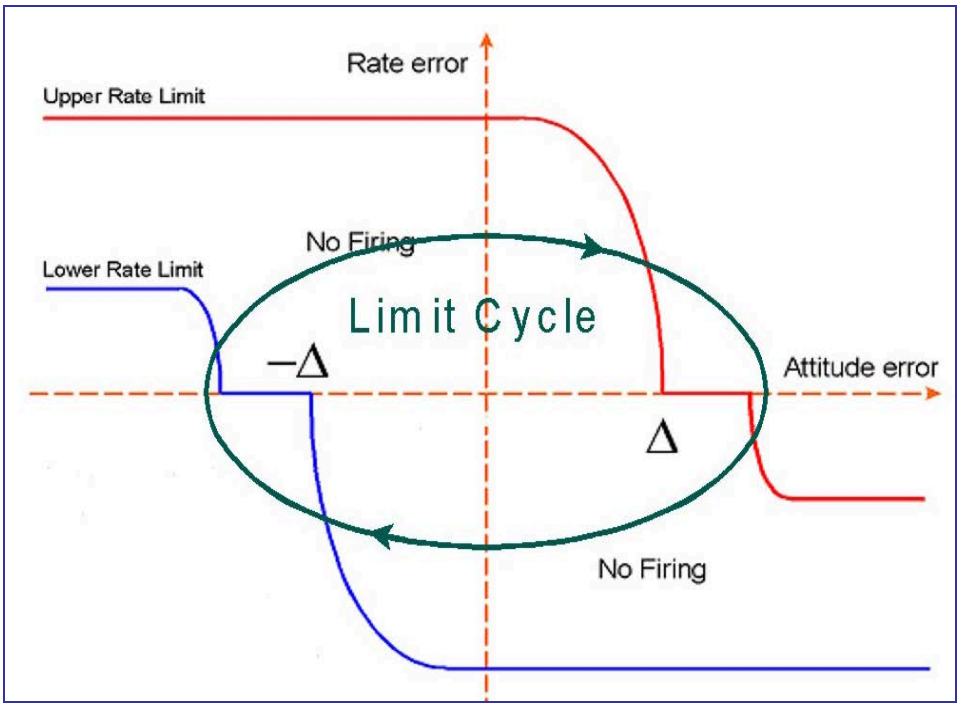


Figure 2.8.6 Sustained Limit-Cycle Trajectory in the Phase-Plane

The next problem to overcome is the fact that the output from the jet selection logic consists of multiple jet forces (F_{jet}). Remember, we need to separate the G(s) part from the N(e) part and, therefore, we must break the control loop at two points. One obvious point to separate the systems is at the spacecraft rate output which is the input to the non-linearity. We must also break the loop at the output of the non-linearity which is 12 jet forces. But it is not convenient to break the control loop at the jet force for stability analysis because the DF method requires a breaking point at a scalar, not a vector. One step closer to our goal is to transform the jet forces into torques (T) in the non-linear control system output and break the loop at the torque output. This transformation creates two separate (3x3) systems in a feedback loop, a non-linear part N(e) and a linear part G(s) that connect to each other by means of roll, pitch, and yaw rates and roll, pitch, and yaw torques, as shown in Figure (2.8.7). By cutting the loop at the torques it reduces the number of outputs to 3 instead of 12. The spacecraft plant inputs in this case must be torques (T). Now, each axis can be analyzed separately using the DF method, and that is not just roll, pitch, and yaw, but other skewed axes in between, because each direction uses a unique set of thrusters and excites the structure differently. The spacecraft torque is related to the jet forces by the following equation

$$T = V_t F_{jet}$$

Where:

$$V_t = \left(\underline{v}_1 \quad \underline{v}_2 \quad \ldots \quad \underline{v}_j\right) \; where: \underline{v}_j = \left(l_j \times u_j\right)$$

$$V_p = pseudo\_inverse\left(V_t\right)$$

where:

l_j is the thruster (j) location relative to the spacecraft CG

u_j is the thrust direction for a thruster (j)

v_j is the torque on the spacecraft created by thruster (j)

But if the new plant input is torque, this torque it must be converted back into jet forces because the original plant model requires forces and, therefore, we use the pseudo-inverse of (V_t), V_{pinv}, to create pseudo forces to the plant input from torques, $F'_{jet} = V_{pinv}\, T$.

These forces will not be the same as the original forces F_{jet}, but they will produce the same amount of torque on the spacecraft. The biggest difference is that F'_{jet} does not excite the structure the same way as the original F_{jet}, but this is acceptable if we assume that the structure is sufficiently stiff between the jets, and that flexibility is mainly due to the appendages, such as, solar array, antennas, etc. It is, therefore, be acceptable to drive the plant input with F'_{jet} instead of the actual jet forces F_{jet}, as long as both force excitations create the same torque. Figure (2.8.7) shows the two (3x3) feedback interconnected systems: a non-linear (orange) block containing the controls and a linear spacecraft dynamics (green) block, interacting by means of torques and body rates. The attitude command is not shown because it does not affect stability.

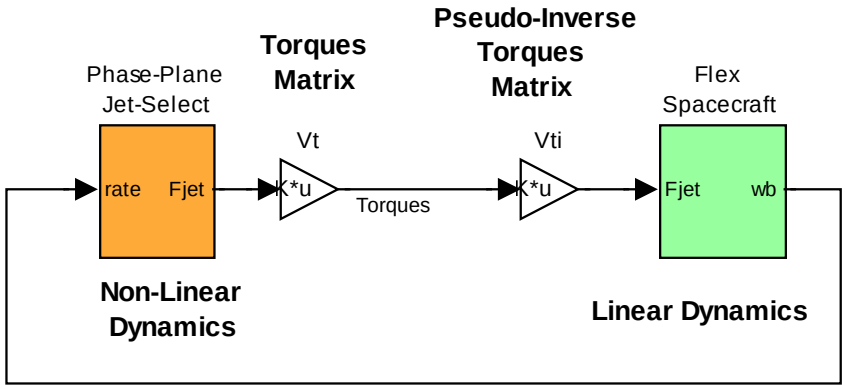


Figure 2.8.7 Modified Non-Linear System used for Describing-Function Analysis

The next step is to separate the above (3x3) systems to individual SISOs and analyze the dynamic motion and stability as if the spacecraft is excited and can move only in one direction at a time. The assumption is that since the system is strongly diagonally dominant by the selection of the jets, if it is stable in all individual SISO directions, it will also be globally stable when it is fully coupled. Let us assume, for example, that we are analyzing the pitch axis. We must first create a pitch SISO plant model by applying a scalar torque in pitch and measure the vehicle response only in pitch, as shown in Figure (2.8.8). Similarly, we can create a roll plant by changing the rotation input vector to (1 0 0), or a yaw plant (0 0 1), or a plan in any skewed direction, ex. (0.2 -0.4 0.5), defined by the rotational vector. We are ignoring, of course, the cross-coupling between axes in order to be able to use the DF method, but if the jets are properly selected the cross-axial coupling is negligible. So we can use the model in figure (2.8.8) to calculate the frequency responses of the plants G(s) in many different directions. The input is a scalar torque which is converted into vector torque in a specific direction and the output is a scalar body rate which is obtained from the spacecraft rate vector resolved in the same direction.

This plant model connects with the phase-plane non-linearity of which we shall calculate its DF using Simulink.

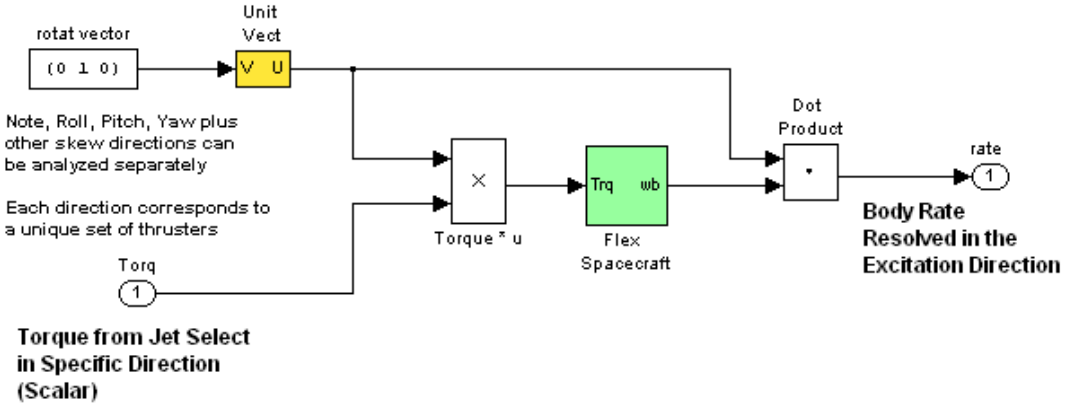


Figure 2.8.8 Plant Model Resolved in a Single Direction (pitch shown above)

Calculating the Describing Function of the Phase-Plane Non-Linearity

We can also apply the same approach to decouple the non-linear control system to behave like a SISO system in a specific rotational direction. Figure (2.8.9) shows the phase-plane non-linearity converted into a SISO Simulink block. The input is a scalar body rate feedback coming from the SISO vehicle model. It is converted to a vector (pitch rate in this example), and it is integrated to obtain attitude. We are assuming that the signal is sinusoidal since we are examining the possibility of limit-cycling. Attitude and rate errors drive the phase-plane (which is running slower, at 100 msec) and it generates the rate commands to the jet selection logic. The jet selection logic generates the jet forces, which turn-off sequentially during the 100 msec control cycle, and generate the control torque (T) after multiplying the acceleration matrix with the jet forces, $T = V_t F_{jet}$. The torque vector (T) is then resolved in a specific direction, pitch in this example, since the input rate was also in pitch. The model in Figure (2.8.9) is used to calculate the DF of the phase-plane and jet-selection logic together using Simulink. The direction of motion is defined by the rotation vector. Notice, that because of the integrator the DF in this case N(e,ω), is a function of both, error amplitude and frequency.

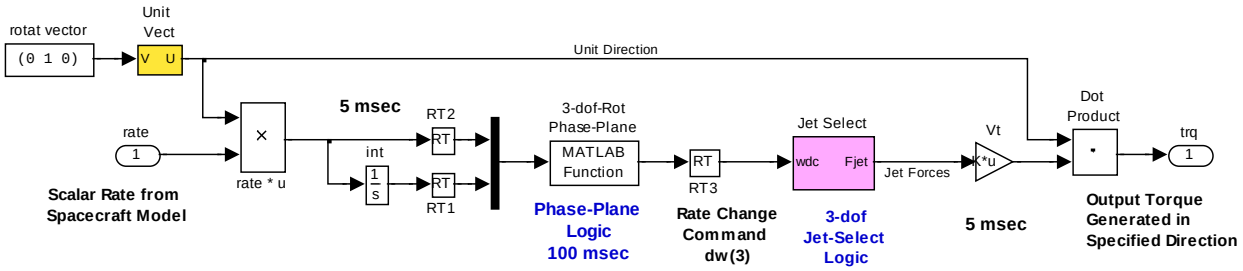
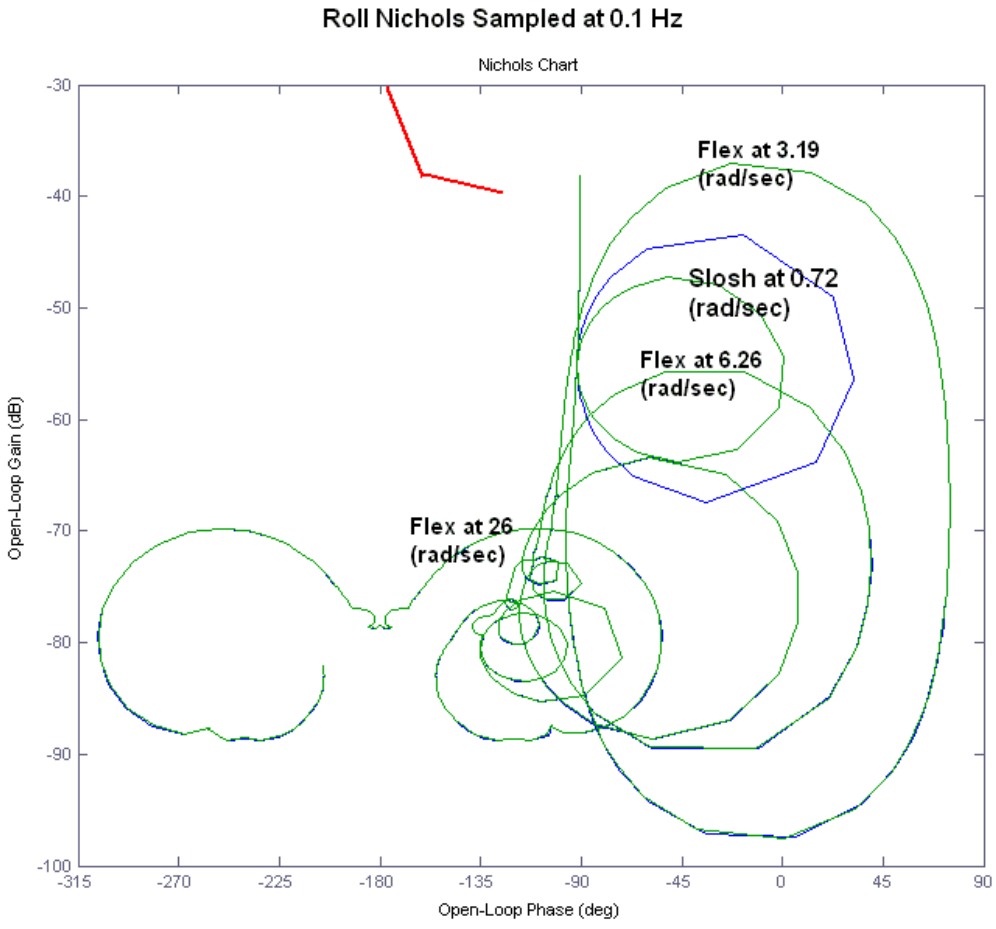
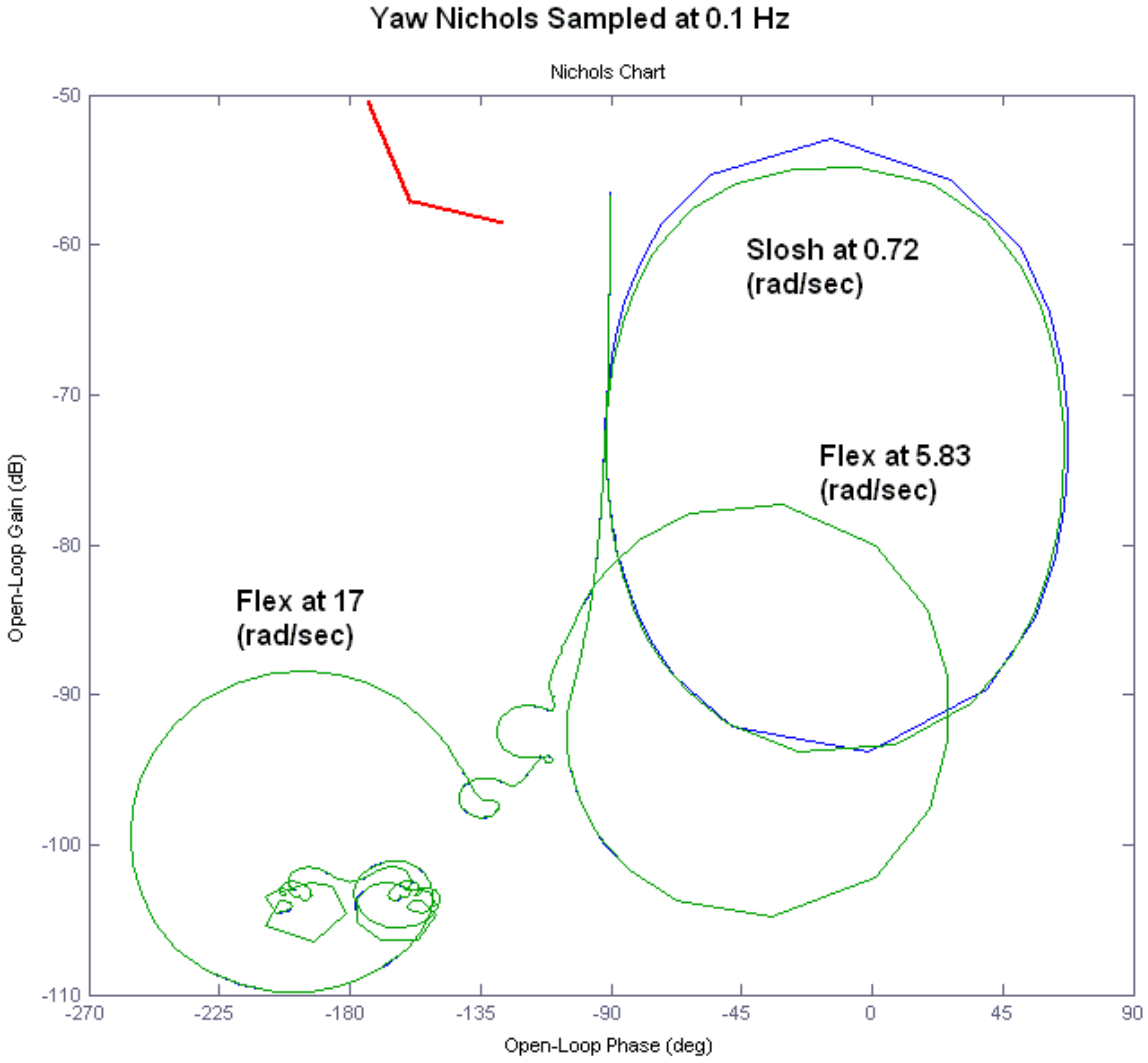


Figure 2.8.9 Non-Linear Control System Resolved in a Single Direction (pitch)

Stability Analysis Using the DF

The sampling frequency of the control loop is at 10 Hz and, therefore, in the DF analysis the plant should be sampled at 10 Hz. In the DF analysis we co-plot the Nichols of the G(s) with the -1/N(e) locus and hope that there are no intersections. The DF in this case is frequency dependent and it would require a 3-d illustration but we can take advantage that the -1/N(e) does not vary much with frequency. We can plot only a few of the loci, $-1/N(e,\omega_1)$, $-1/N(e,\omega_2)$, $-1/N(e,\omega_3)$, where ω_i correspond to some of the big resonances, that stick out, such as slosh and a couple of appendage modes below 5 Hz and overlay them with the G(s) of the linear plant. This analysis must be repeated for at least 3 directions separately, roll, pitch, yaw, plus a few additional skewed directions, but we are only showing a couple cases. The file "run_freq" discretizes the two Simulink models "Open_Loop_FEM.mdl" and "Open_Loop_FVP.mdl" and calculates the Nichols plots shown in Figure (2.8.15) for roll and yaw. The Describing Function overlay was done manually from the plots in Figure (2.8.12). There is a slight difference in the size of the slosh resonance between the models, appearing in the roll direction. The results clearly indicate that this system is not threatened by limit cycles in neither direction. In fact, all resonances are well behaved in phase, exhibiting min-phase type behavior.




Yaw Nichols Sampled at 0.1 Hz
Nichols Chart
Slosh at 0.72
(rad/sec)
Flex at 5.83
(rad/sec)
Flex at 17
(rad/sec)
Open-Loop Gain (dB)
Open-Loop Phase (deg)
-50
-60
-70
-80
-90
-100
-110
-270
-225
-180
-135
-90
-45
0
45
90

3. Spacecraft Controlled by an Array of Single Gimbal Control Moment Gyros

This section provides a detailed tutorial on designing spacecraft Attitude Control Systems using Single-Gimbal Control Moment Gyros devices. We will use the same spacecraft model that was described in the previous section using RCS and design an alternate ACS that uses SG-CMGs. The ACS consists of a cluster of four SG-CMGs mounted together on a solid structure near the center of the spacecraft. They are controlled by a non-linear Max Energy control logic that attempts to use the maximum control torque and momentum capability of the CMG devices to improve (in comparison with linear controls) the spacecraft ability to maneuver. The control law also uses a steering logic that prevents CMG singularities. Singularities or "gimbal locks" occur when the SGCMG cluster cannot provide torque in a required direction. In the following sections we will discuss the SGCMG dynamics, the equations of motion of a spacecraft with SG-CMGs, design the control and steering laws, and describe the simulation models. We start with simple rigid-body models that exclude the CMG gimbal dynamics and the flexibility of the structure, and we gradually upgrade the models with more details. We finally analyze a low cost configuration that uses a combination of one reaction wheel and two CMGs. The flex spacecraft models are created using two separate Flixan modeling programs for comparison.

3.1 CMG Array Geometry

A single gimbal control moment gyroscope (SGCMG) is shown in Figure 3.1. It consists of a spinning rotor that is mounted on gimbal perpendicular to the rotor axis. The rotor spin rate is maintained at a constant speed by a small motor that produces a constant angular momentum (h_{cmg}). The momentum direction can be rotated by a stronger motor which is mounted at the gimbal. The gimbal motor controls the gimbaling rate, and hence the output torque. By commanding the gimbal to rotate (by means of a servo system that controls the gimbal motor), high precession torques are generated by changing the orientation of the angular momentum vector. The reaction torque on the spacecraft (T) is equal and opposite to the rate of change in momentum vector $\dot{h}$, which is orthogonal to the momentum vector and the gimbaling vector according to the right hand rule. At any instant it is a function of the gimbal position. In fact the torque magnitude is equal to the CMG momentum multiplied by the gimbal rate. A CMG generates much greater torques than a reaction wheel and for this reason it is very attractive in high torque and fast maneuvering spacecraft applications. Notice also, that the SG-CMGs generate the high precession torques without requiring high power.

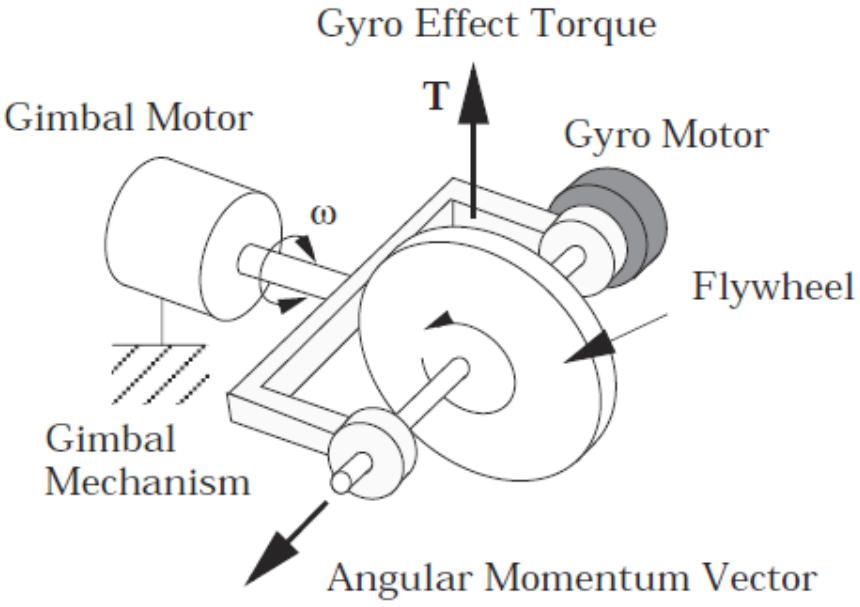
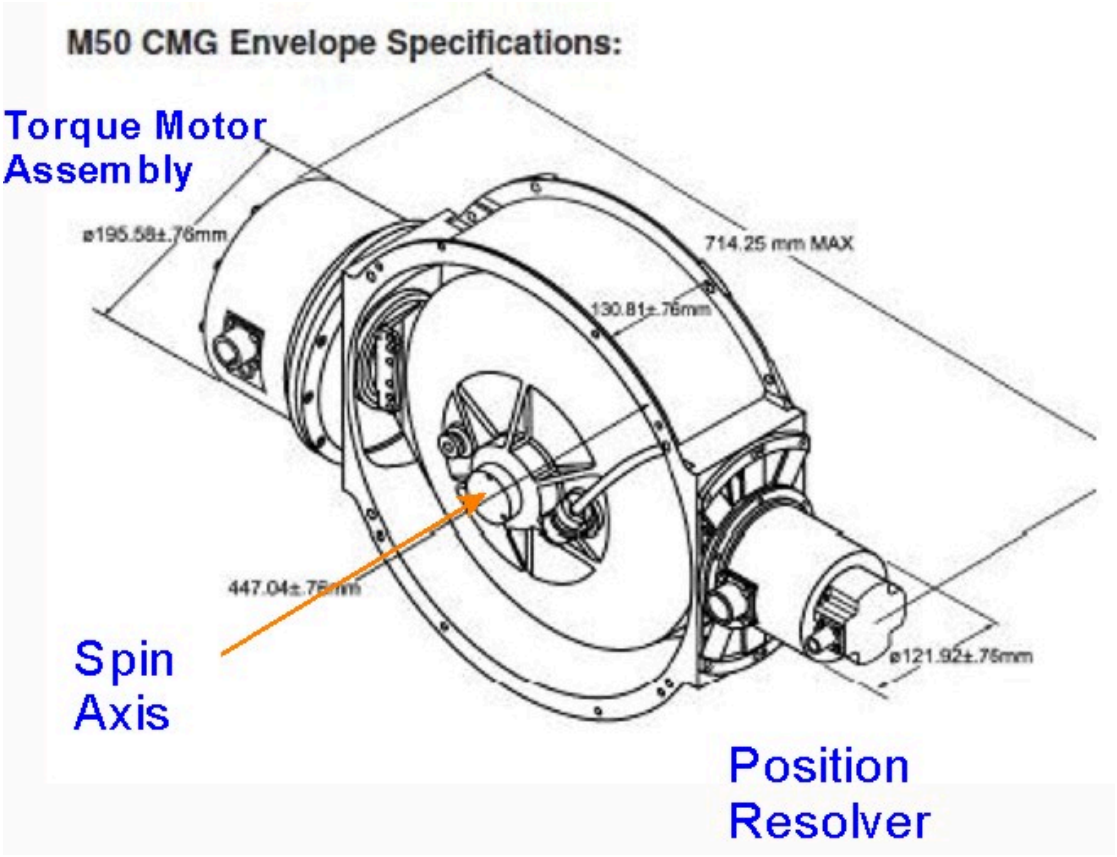


Figure 3.1 Single-Gimbal Control Moment Gyro

Another attractive feature of CMGs compared with reaction wheels is that the rotor in a CMG spins at a constant rate which places the vibrations at known frequencies while in a RW, the rotor speed changes, thus, exciting the spacecraft structure in multiple frequencies which may not be desirable in precision applications. CMGs, however, are complex systems, expensive, and require complex controls with singularity avoidance algorithms. Figure 3.2 shows a picture of a SG-CMG. It consists of a rotor that spins at a constant high rate about an axis that can be rotated. There is also a torque motor assembly that rotates the rotor about a gimbal axis that is fixed relative to the spacecraft, and a position sensor that measures the gimbal rotation relative to the spacecraft.



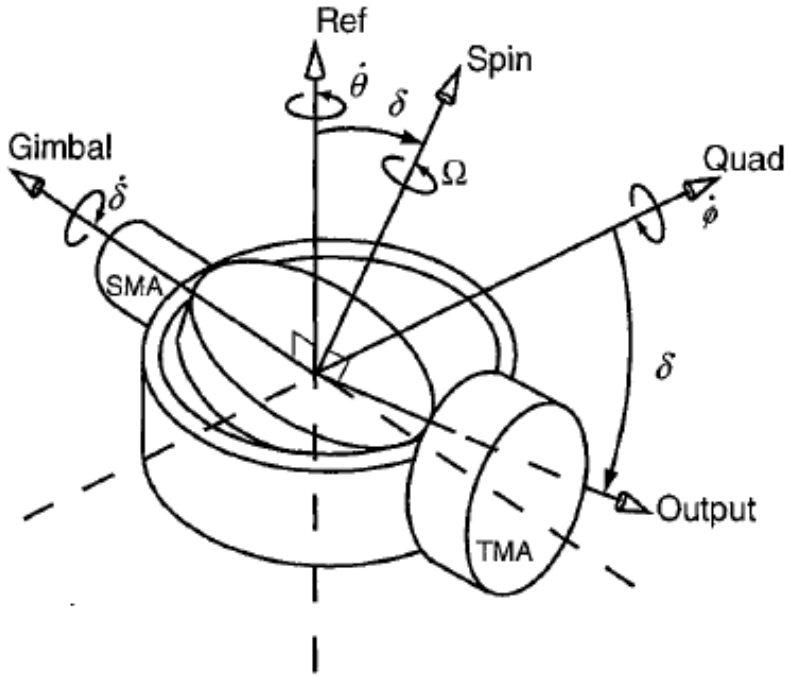


Figure 3.2 Single Gimbal Control Moment Gyro

The momentum of the CMG due to the spinning rotor and due to the rotation of the other axes in the Gimbal, Output, and Spin axes is

$$\begin{pmatrix} h_g \\ h_o \\ h_s \end{pmatrix} = \begin{pmatrix} J_g \dot{\delta}_i \\ J_o\left(\dot{\phi}\cos\delta - \dot{\theta}\sin\delta\right) \\ h_0 + J_s\left(\dot{\theta}\cos\delta + \dot{\phi}\sin\delta\right) \end{pmatrix} \tag{3.1}$$

Figure 2.3 defines the orientation of a Single Gimbal CMG with respect to the spacecraft reference axes. The rate of change of momentum which is the moment generated by a SGCMG in the Gimbal, Output, and Spin axes respectively as a result of gimbaling and base motion are:

$$\begin{bmatrix} M_G \\ M_O \\ M_S \end{bmatrix} = \begin{bmatrix} T_{gi} \\ J_o\left(\ddot{\phi}\cos\delta - \dot{\phi}\dot{\delta}_i\sin\delta - \dot{\theta}\dot{\delta}_i\cos\delta - \ddot{\theta}\sin\delta\right) + h_0\dot{\delta}_i + \dot{\delta}_i\left(J_s - J_g\right)\left(\dot{\theta}\cos\delta + \dot{\phi}\sin\delta\right) \\ J_s\left(\dot{\Omega} + \ddot{\theta}\cos\delta + \ddot{\phi}\sin\delta + \dot{\phi}\dot{\delta}_i\cos\delta - \dot{\theta}\dot{\delta}_i\sin\delta\right) + \dot{\delta}_i\left(J_g - J_o\right)\left(\dot{\phi}\cos\delta - \dot{\theta}\sin\delta\right) \end{bmatrix} \tag{3.2}$$

The reaction torque on the spacecraft is minus [M_G, M_O, M_S]

Where:

J_g is the CMG inertia about its gimbal axis
J_o is the CMG inertia about its output axis
J_s is the CMG inertia about its spin axis
T_{gi} is the torque applied by the torque motor at the gimbal
$\ddot{\delta}_i$ is the gimbal inertial angular acceleration including spacecraft
δ is the CMG gimbal rotation about the m axis with respect to spacecraft r axis
h_0 is the constant CMG momentum about its spin axis ($I_s\Omega$)
$\dot{\Omega}$ is the rotor spin acceleration
$\dot{\theta}$ is the vehicle rate in the CMG r axis
$\dot{\phi}$ is the vehicle rate in the CMG q axis

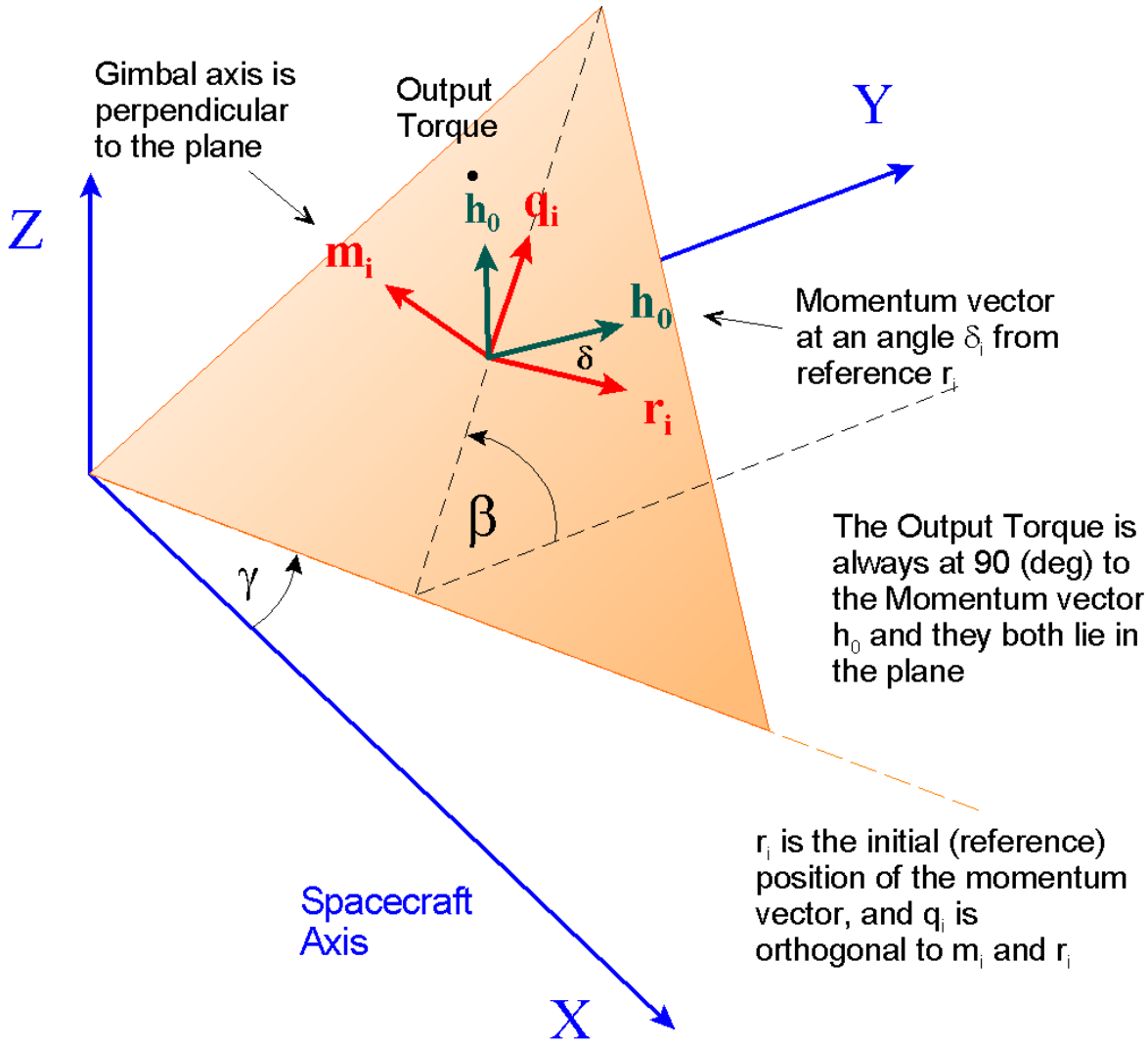


Figure 3.3 Orientation of a CMG in Spacecraft Coordinates

The rotation rates of the CMG axes can be related to the spacecraft body rate. If $\underline{\omega}$ is the spacecraft body rate vector (ω_X, ω_Y, ω_Z), the following relationships resolve the spacecraft rates about the CMG axes: (r, q, m). $\dot{\theta}$ *and* $\dot{\phi}$ are the spacecraft rates resolved about the CMG reference and quad axes. The gimbal rate $\dot{\delta}_i$ in addition to the gimbal rate relative to spacecraft $\dot{\delta}$ it includes also the spacecraft rate about the CMG gimbal.

$$\begin{aligned}
\dot{\theta} &= \omega_X \cos\gamma + \omega_Y \sin\gamma \\
\dot{\phi} &= -\omega_X \cos\beta \sin\gamma + \omega_Y \cos\beta \cos\gamma + \omega_Z \sin\beta \\
\dot{\delta}_i &= \dot{\delta} + \omega_X \sin\beta \sin\gamma - \omega_Y \sin\beta \cos\gamma + \omega_Z \cos\beta
\end{aligned} \qquad (3.3)$$

The following projection matrix (P) transforms the CMG torques from CMG axis to spacecraft axis.

$$\begin{pmatrix} M_X \\ M_Y \\ M_Z \end{pmatrix} = \begin{bmatrix} \sin\beta\sin\gamma & -\sin\delta\cos\gamma - \cos\delta\cos\beta\sin\gamma & \cos\delta\cos\gamma - \sin\delta\cos\beta\sin\gamma \\ -\sin\beta\cos\gamma & -\sin\delta\sin\gamma + \cos\delta\cos\beta\cos\gamma & \cos\delta\sin\gamma + \sin\delta\cos\beta\cos\gamma \\ \cos\beta & \cos\delta\sin\beta & \sin\delta\sin\beta \end{bmatrix} \begin{pmatrix} M_G \\ M_O \\ M_S \end{pmatrix}$$

When CMGs are used to steer spacecraft, at least three CMGs are needed to provide 3 axes control. If we consider an array of SGCMG mounted on the surfaces of a pyramid with their gimbal axes directions ($\underline{m}_i$) perpendicular to the corresponding surface and the momentum direction (h_i) always aligned with the surface of the pyramid as the gimbal σ_i rotates. The output torque from each CMG is equal to the rate of change of angular momentum which is in the ($\underline{m}_i$ x $\underline{h}_i$) direction and proportional to the gimbal rate $\dot{\delta}_i$. From the pyramid surfaces orientations we can calculate some important matrices that will be used in the equations of motion.

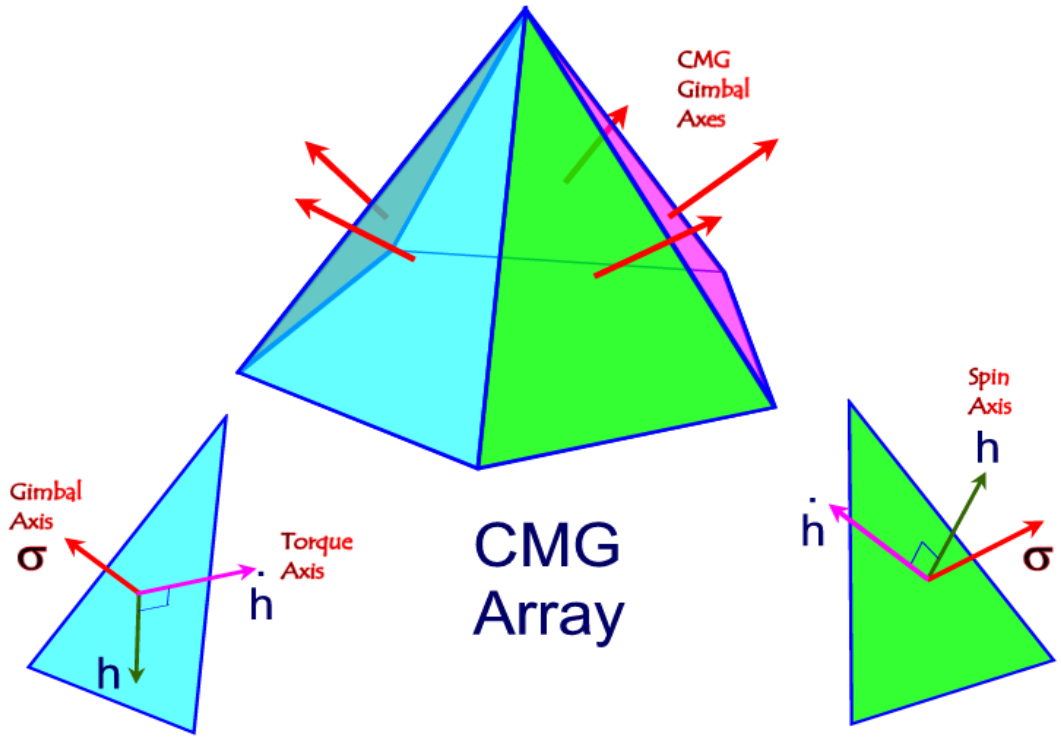


Figure 3.4 Array of five CMGs in a Pyramid Configuration

Let us consider the CMG pyramid arrangement presented in this example shown in Figure 3.5. The spacecraft has four SGCMGs which are mounted to the four faces of a four sided pyramid. All CMGs have the same angular momentum, h_{cmg}=1200 (ft-lb-sec) about their spin axis. Their CMG momentum vectors ($\underline{h}_i$) can be rotated about the gimbal vectors ($\underline{\delta}_i$), which are perpendicular to each surface, and they are constrained to lay parallel to the surface of the pyramid. The pyramid angle β is 68 (deg), and the γ_i angles of the four surfaces, according to figure (2.3), are: (90º, 180º, 270º, and 0º). The columns of the following (3x4) matrix M_g^b contains the four gimbal direction unit vectors $\underline{m}_i$. It is a gimbal to body transformation matrix.

$$M = \begin{bmatrix} \underline{m}_1 & \underline{m}_2 & \underline{m}_3 & \underline{m}_4 \end{bmatrix} \; where: \; \underline{m}_i = \begin{bmatrix} \sin\beta_i \sin\gamma_i \\ -\sin\beta_i \cos\gamma_i \\ \cos\beta_i \end{bmatrix}$$

$$M = \begin{bmatrix} \sin\beta & 0 & -\sin\beta & 0 \\ 0 & \sin\beta & 0 & -\sin\beta \\ \cos\beta & \cos\beta & \cos\beta & \cos\beta \end{bmatrix} = \begin{bmatrix} 0.927 & 0 & -0.927 & 0 \\ 0 & 0.927 & 0 & -0.927 \\ 0.375 & 0.375 & 0.375 & 0.375 \end{bmatrix} \tag{3.5}$$

The (3x4) matrix [R] represents the momentum reference directions. That is, the initial directions $\underline{r}_i$ of the momentum vectors $\underline{h}_{cmg(i)}$. More precisely, are the momentum directions when the gimbal angles ($\underline{\delta}_i$) are at zero. The initial gimbal angles δ_{0i}=0 to provide zero momentum bias.

$$R = \begin{bmatrix} \underline{r}_1 & r_2 & \underline{r}_3 & \underline{r}_4 \end{bmatrix} \; where: \; \underline{r}_i = \begin{bmatrix} \cos\gamma_i \\ \sin\gamma_i \\ 0 \end{bmatrix}; \; R = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

We must also define a (3x4) matrix Q, containing column vectors of the cross product direction unit vectors ($\underline{q}_i$).

$$Q = \begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \underline{q}_3 & \underline{q}_4 \end{bmatrix}; \; \underline{q}_i = (\underline{m}_i \times \underline{r}_i);$$

$$Q = \begin{bmatrix} -0.375 & 0 & 0.375 & 0 \\ 0 & -0.375 & 0 & 0.375 \\ 0.927 & 0.927 & 0.927 & 0.927 \end{bmatrix} \tag{3.7}$$

Notice, that the pyramid structure is only used for visualization. The CMGs do not have to be physically mounted on the four surfaces of an actual pyramid, as in Figure 3.5, but they can be translated anywhere on the spacecraft as long as their gimbal axes ($\underline{m}_i$) and their reference momentum vectors ($\underline{r}_i$) are parallel to the directions shown in the pyramid. See, for example, the CMG cluster in Figure 3.6. The CMGs are typically mounted on a structure that it is mechanically isolated from the spacecraft by means of vibration isolation struts, as shown in Figure 3.6, that attenuate vibrations from the CMGs.

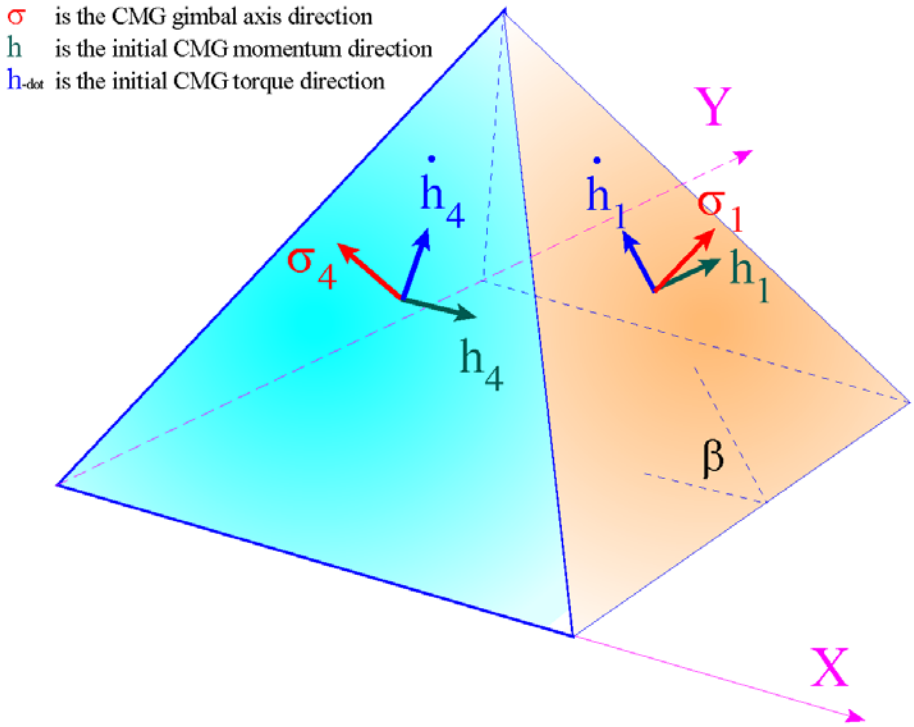


Figure 3.5 Array of four CMGs in a Pyramid Configuration

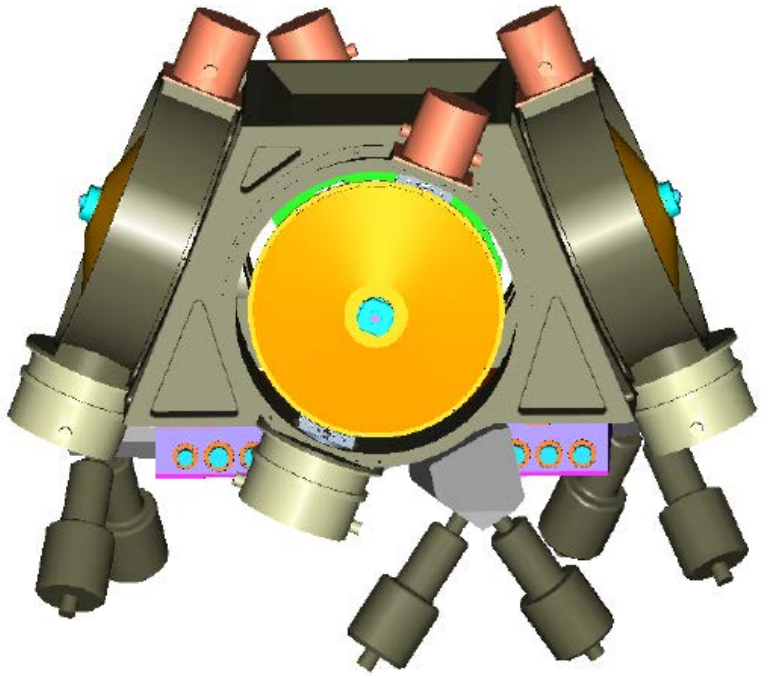


Figure 3.6 A Cluster of four Single Gimbal CMGs mounted on a pyramid structure which is isolated from the spacecraft by means of disturbance isolation struts

Attitude Control System Description

The maximum energy control system block diagram is shown in Figure 3.3.2. It consists of an inner rate control loop (D) and an outer (PI) attitude control loop. The CMG array steering logic is also included in the rate control loop. The inner rate control loop must be designed first. It receives a body rate command from the attitude controller and regulates the vehicle rate by issuing acceleration commands to the RW steering logic. If the bandwidth of the rate loop is sufficiently high to keep up with the body rate command then we can assume that ($\omega \approx \omega_c$) and continue with the outer loop.

Inner Rate Loop

The rate control loop regulates the spacecraft rate. It receives (roll, pitch, and yaw) rate error commands which become acceleration commands that drive the CMG steering law. The steering law generates the CMG gimbal rate commands ($\dot{\delta}_{com}$). An acceleration limiter (AL) is included in the rate loop that prevents the CMG torque commands from exceeding their maximum torque capability. The steering law is given in equation (3.3.4). It uses a pseudo-inverse of matrix A which is defined in equation 3.15 and it is a function of the gimbal angles δ_i

$$\underline{\dot{\omega}}_{com} = k_r \, \underline{\omega}_{err}$$

$$if \; |\underline{\dot{\omega}}_{com}| > AL, \quad \underline{\dot{\omega}}_{com} = \underline{\dot{\omega}}_{com} \frac{AL}{|\underline{\dot{\omega}}_{com}|} \tag{3.3.4}$$

$$\underline{\dot{\delta}}_{com} = -\left[A^T \left(AA^T\right)^{-1} + \lambda E\right] J \, \underline{\dot{\omega}}_{com}$$

Depending on the orientation of the gimbal angles δ_i the pseudo-inverse matrix may become singular. To avoid the singularity, the pseudo-inverse matrix is modified using a singularity avoidance logic (λE), where:

$$E = \begin{bmatrix} 1 & 0.01\sin(wt) & 0.01\cos(wt + \frac{\pi}{2}) \\ 0.01\sin(wt) & 1 & 0.01\sin(wt - \frac{\pi}{2}) \\ 0.01\cos(wt + \frac{\pi}{2}) & 0.01\sin(wt - \frac{\pi}{2}) & 1 \end{bmatrix} \; and \; \lambda = \frac{k}{\det(AA^T)}$$

Where: w and *k* are properly selected. The singularity avoidance logic checks the condition of the A matrix and introduces a small perturbation to prevent a gimbal lock. When the matrix A is moving towards a singularity the perturbation rotates around it. The closer to a singularity the bigger the perturbation gets.

Attitude Control Loop

The input to the attitude control loop is the attitude error, or more precisely, quaternion error. The attitude error signal goes to the PI controller that operates in two modes based on the switch k_y which is either set to zero or one. During maneuvering (when k_y=1) the controller becomes a simple proportional gain K_p. Otherwise, during PID mode, (that is when k_y=0) the block becomes a PI transfer function $K_P \frac{s+b}{s}$ that provides better tracking and disturbance attenuation at low frequencies. The switch setting is controlled by a signal that is a combination of rate plus attitude error. In the beginning of the maneuver when this error signal is large k_y is set to 1. When the error signal drops below a certain value k_y is set to zero and the integrator is turned on.

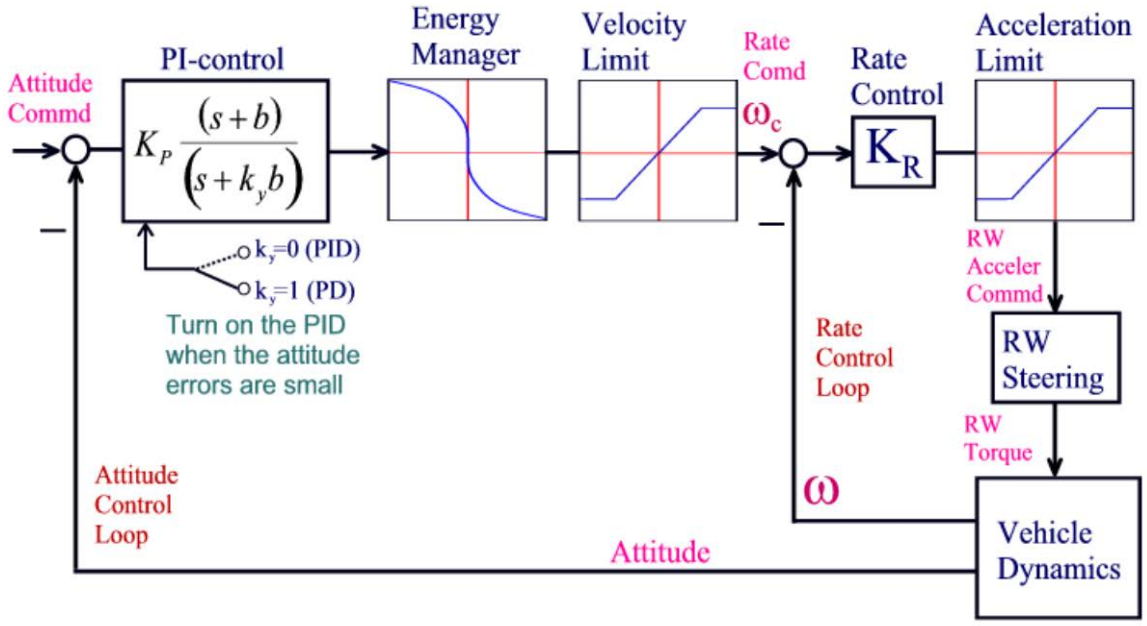


Figure 3.3.2 Block Diagram of Maximum Energy Control System

The energy manager block shapes the velocity command as a function of the attitude error in order to bring the phase-plane trajectory directly to zero without chattering, according to the parabolic switching line in equation (3.3.3). This maintains a proportional attitude error reduction in all directions throughout the maneuver. The velocity limiter in series with the energy limiter bounds the vehicle rate during the acceleration phase of the maneuver. It also bounds the CMG momentum during maneuvering preventing it from reaching saturation levels.

3.4 Simulation Models

In this section we are presenting simulations and analysis of the agile spacecraft controlled by an array of four SGCMG. We start with a simple rigid-body simulation model that uses equations (2.13) and is assuming that the gimbal rates are equal to the gimbal rate commands coming from the steering logic. The CMG momentum is calculated from equation (3.14) as a function of the gimbal angles. The gimbal rate commands are calculated by the steering logic which was described in equation 3.3.4. The control torque on the spacecraft is $\underline{T}_{con} = -[A(\delta)]\dot{\underline{\delta}}$. This model is used for initial evaluation of the control law before advancing into more complex models. The second simulation model is still rigid-body uses equations (3.11) and (3.12). The CMG torque in addition to the control torque T_{con} it contains also an estimate of the gyroscopic torques. The CMG momentum is calculated by integrating the second part of equation (3.11). The CMG torques are calculated from equations (3.2), (3.4) and (3.10). The gimbal rates are calculated by integrating equation (3.16). The gimbal torque Tgi is provided by a gimbal torque motor servo system. The quaternion output is obtained by integrating equation (3.17). The third simulation model is similar to the second one but it was adjusted to include structural bending.

3.4.1 Max-Energy/ 4 CMG Simple Rigid Body Simulation

Figure 3.4.1.1 shows the Max-Energy Attitude Control System simple simulation model that uses four Single-Gimbal CMGs. It is implemented in a Simulink file "*MaxEn-NonLin_4SGCMG.mdl*" which is located in folder "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(a) Simple RigBody 4SGCMG ACS*". It consists of three major blocks: the Attitude Control System, the inner rate loop and Steering logic block, and the spacecraft/ CMG dynamics. The ACS receives a quaternion command which is compared with the spacecraft quaternion attitude to generate the error signal which drives the Steering logic and rotates the spacecraft.

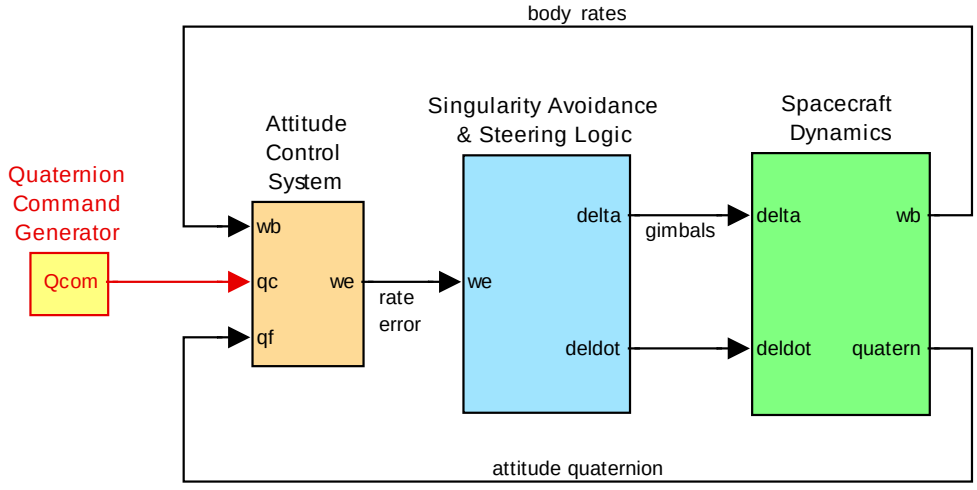


Figure 3.4.1.1 Max-Energy ACS Simulation Model "MaxEn_NonLin_4SGCMG.mdl"

Simulation Results

The initialization file “Start.m” initializes the Simulink models before running the simulations. This m-file loads the spacecraft mass properties, CMG parameters, and a transformation matrix Tc2b from CMG to body coordinates. It calls function M4.m to calculate the matrices M, R, and Q. It defines the CMG gimbal system bandwidth, damping, plus gimbal rate and acceleration limits. It defines also spacecraft ACS and steering max rate limits and max accelerations, ACS gains, the singularity avoidance parameters, and the rotation command eigenaxis. The rotation angle command is defined inside the quaternion command yellow block in the simulation model. The simulation results in Figure (3.4.1.3) show the ACS response to a 90° maneuver.

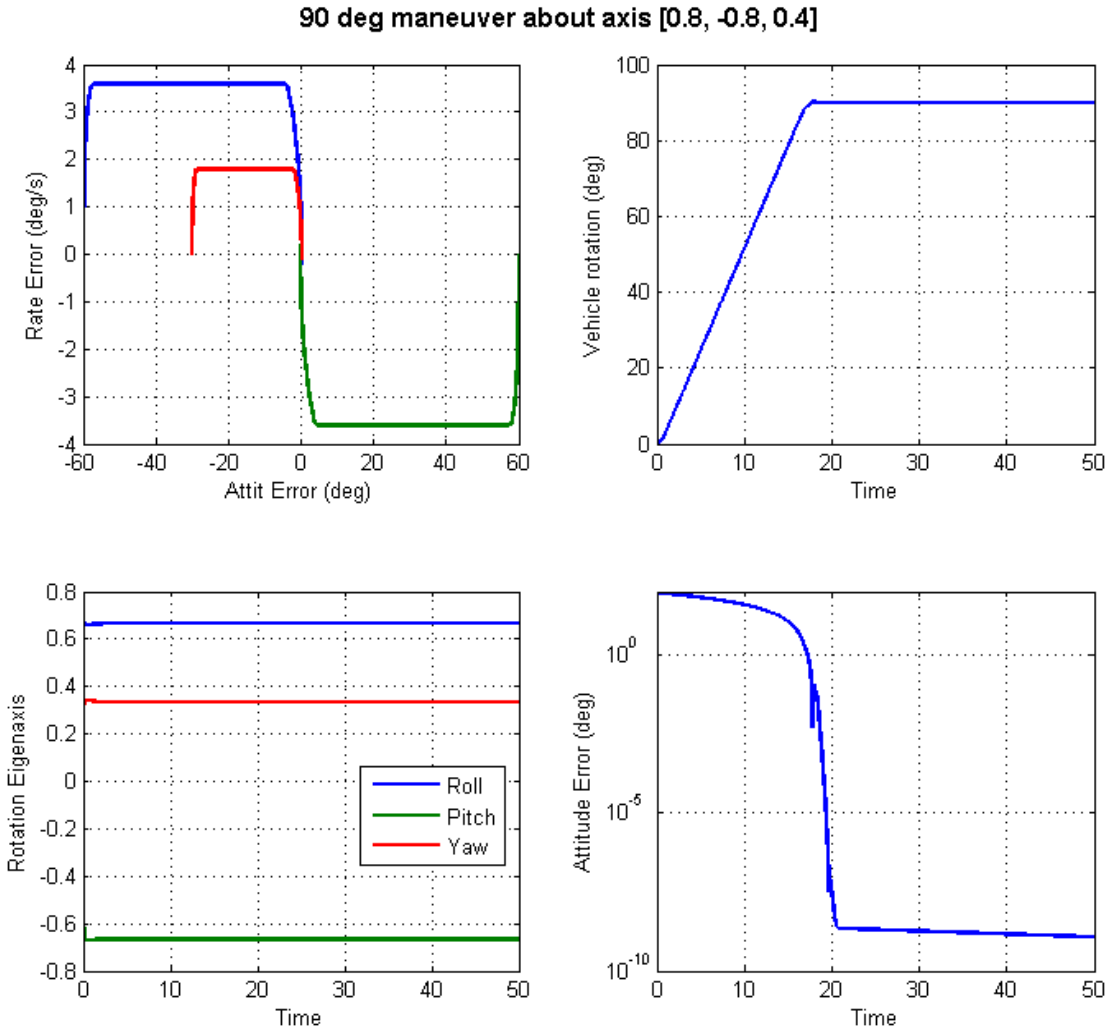


Figure 3.4.1.3(a) System performs a perfect rotation about the commanded eigenaxis achieving very small attitude errors in 20 seconds.

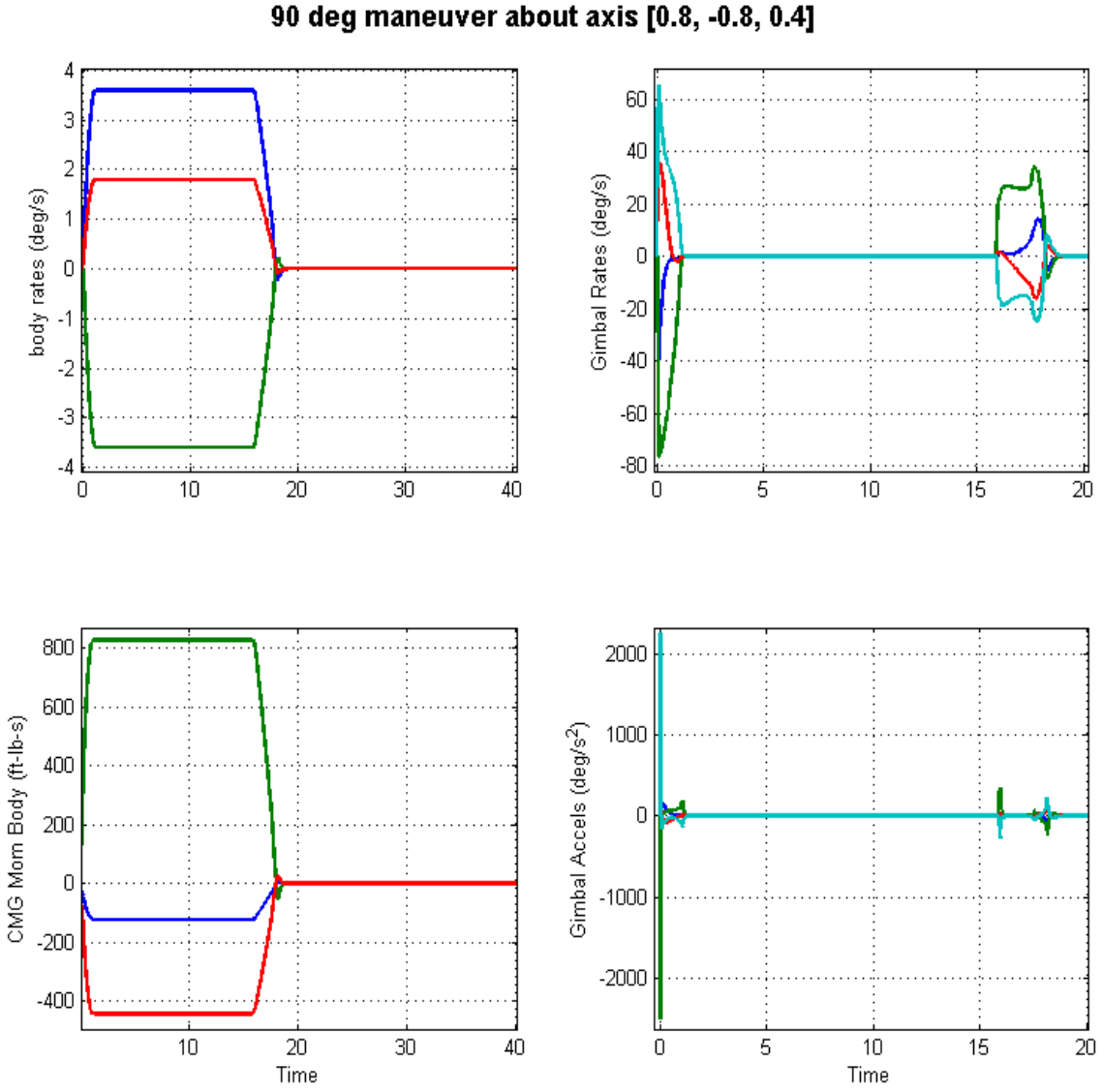


Figure 3.4.1.3(b) Max acceleration is used to reach max momentum capability where the vehicle maintains constant rate until acceleration is reversed to slow it down to its target position.

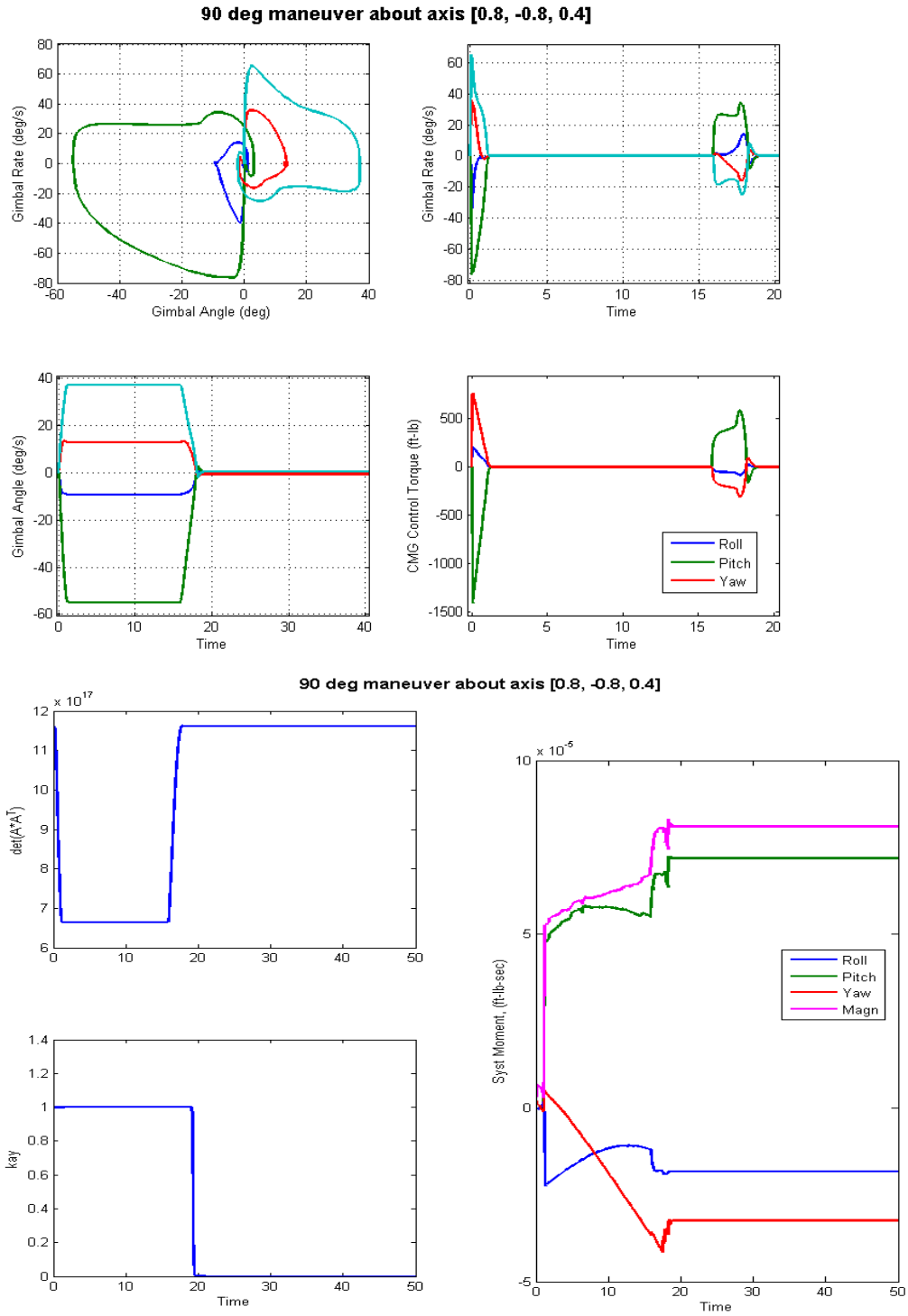


Figure 3.4.1.3(c) System momentum remains constant (zero) throughout the maneuver. At approx. 20 seconds the PID integrators are turned on to further attenuate attitude errors. The gimbal angles are at constant positions during the middle section of the maneuver maximizing the spacecraft momentum along the commanded eigenaxis. The gimbal rates and torques are zero when the momentum is constant.

Spacecraft Dynamics Function SC_CMG_Gimbal

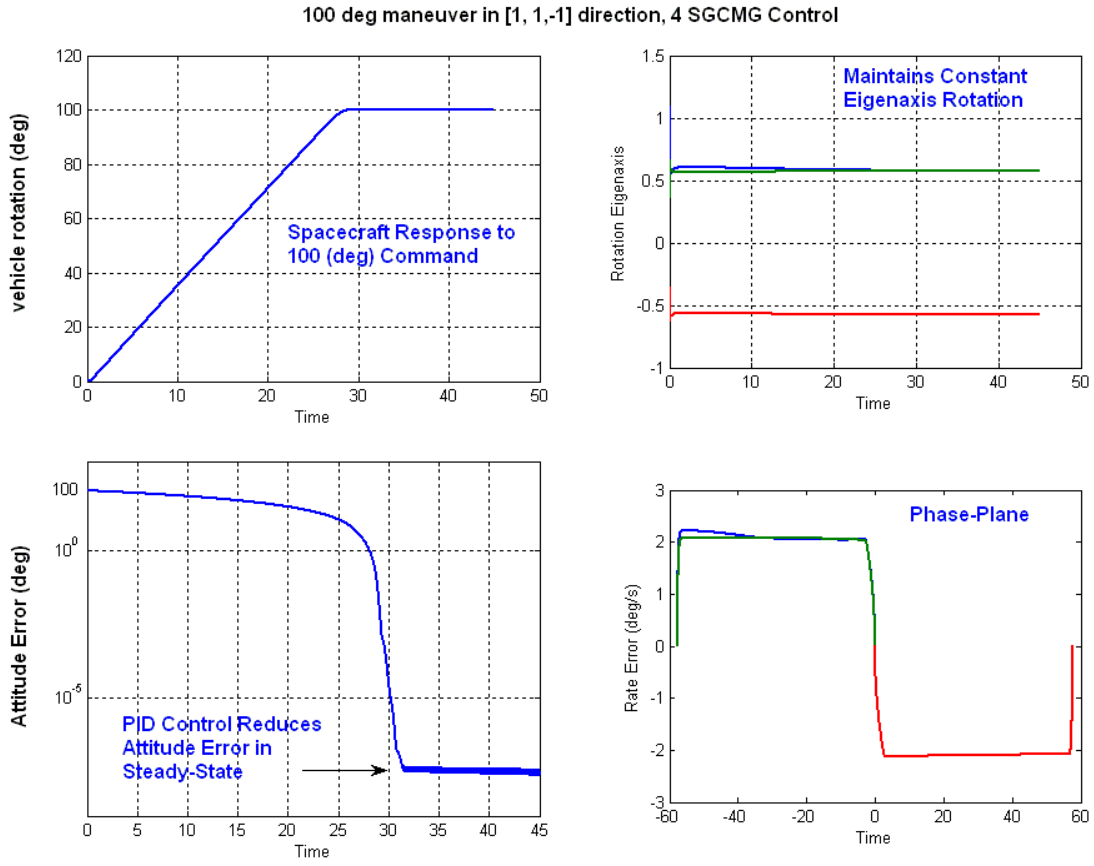
```
function xdot= SC_CMG_Gimbal(x,Tci,Td)                % s/c Dynamics with CMG
global J Jinv Jgi m ref quad hcmg d2r r2d
global bet gam Tc2b

% State Variables (x)
% x(1:3)   = Body rates   (w)        (rad/sec)
% x(4:6)   = CMG Momentum body (h)   (ft-lb-sec)
% x(7:10)  = Gimbal rates (delt-dot) (rad/sec)
% x(11:14) = Gimbal angles (delta)   (rad)
% x(15:18) = Quaternion
% Inputs:
% Tci(4)   = Gimbal Torques (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)

xdot= zeros(28,1);
w =     x(1:3);                                       % Body rates
h =     x(4:6);                                       % CMG Momentum
deldot= x(7:10);                                      % Gimbal rates
delta = x(11:14);                                     % Gimbal Angles
qt=     x(15:18);                                     % Quaternion
[Pj,thd,phd,ddd]= Transforms(bet,gam,delta,w);

Tcmg=zeros(3,1); h2=Tcmg;
for i=1:4
    Mj=[Tci(i); ...                                   % CMG Torque in CMG axis
        hcmg(i)*deldot(i); ...
        0];
    Tcmg= Tcmg -Tc2b*Pj(:,:,i)*Mj;                    % Torque on Vehicle
%   h2= h2 + (cos(delta(i))*ref(:,i) + sin(delta(i))*quad(:,i))*hcmg(i);
end

Hs = J*w + h;                                         % System Momentum

xdot(1:3)= Jinv*(Tcmg -cross(w,J*w) +Td);             % Vehicle acceleration
wdot=xdot(1:3);                                       % Vehicle Acceleration
xdot(4:6)=-Tcmg -cross(w,h);                          % H-dot
for i=1:4
    sd=sin(delta(i)); cd=cos(delta(i));
    xdot(6+i)= (Tci(i)-hcmg(i)*(thd(i)*sd-phd(i)*cd))/Jgi;  % Gimbal accelr delt-ddot
    xdot(10+i)= x(6+i);                               % Gimbal rates  delta-dot
end

xdot(15:18)= 0.5*[0    w(3)  -w(2)   w(1);            % Quaternion Update
              -w(3)    0      w(1)   w(2);
               w(2)   -w(1)   0      w(3);
              -w(1)   -w(2)  -w(3)   0]*qt;
xdot(19:21)= Hs;                                      % System Momentum Hs
xdot(22:24)= Tcmg;                                    % CMG Torques on Vehi
xdot(25:28)= ddd;                                     % delta_dot (Inert-Relat)
```

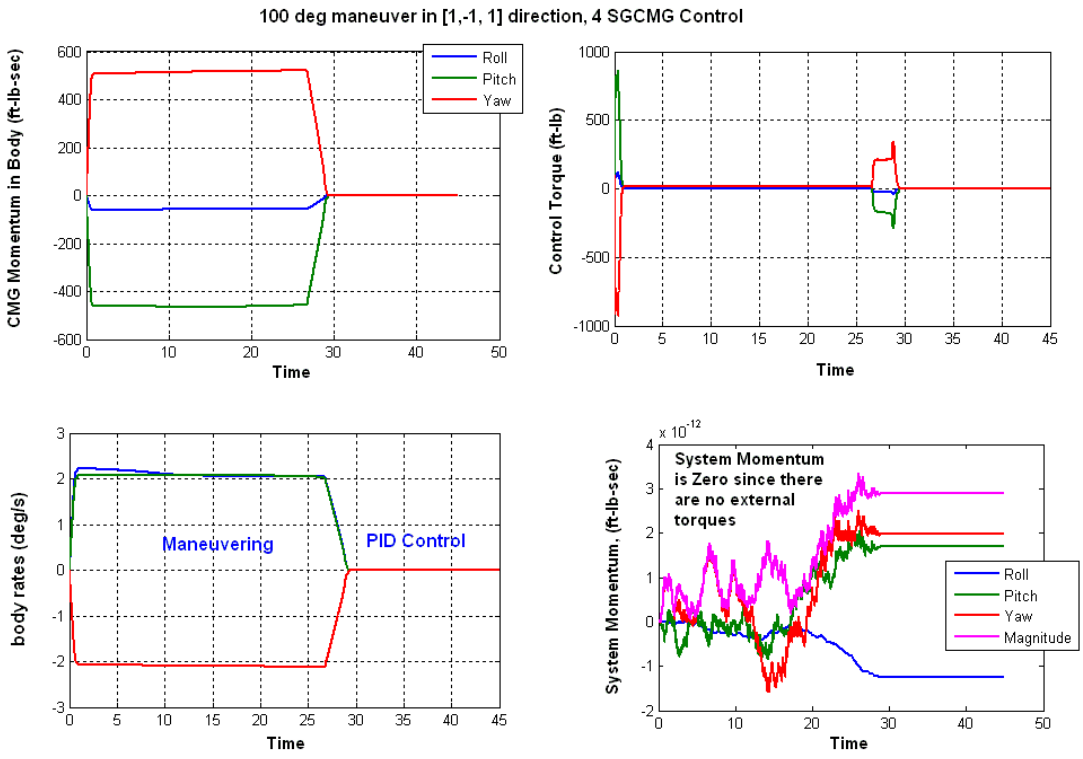
Simulation Results

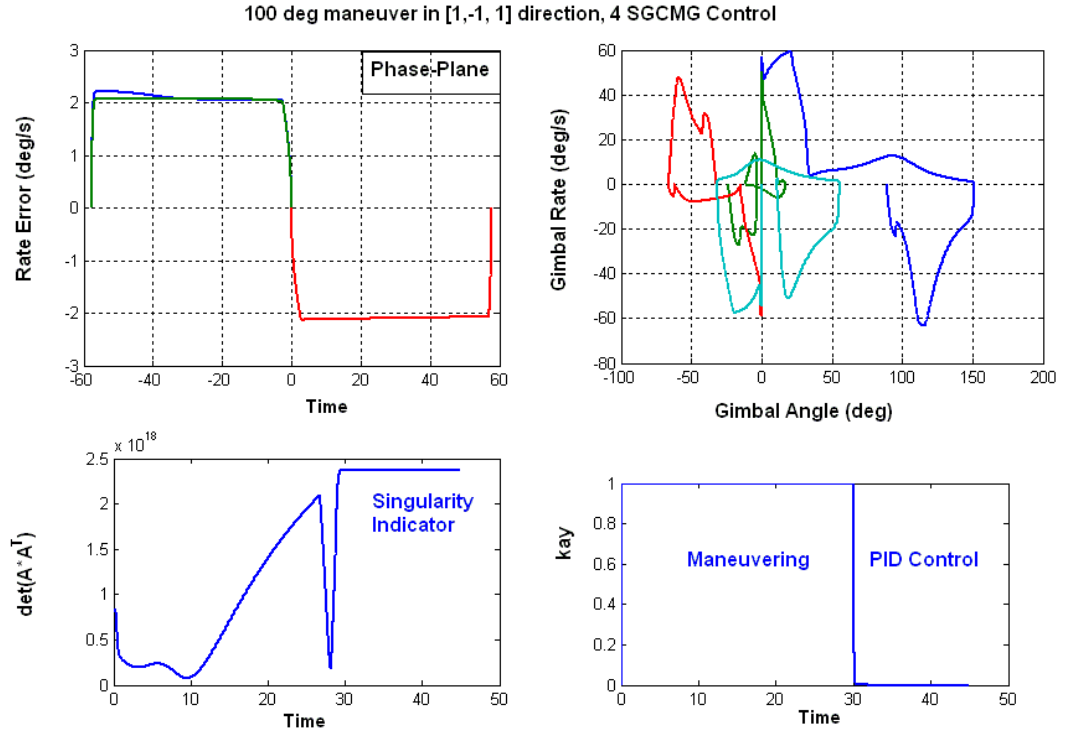
The Simulink model described will be used here to demonstrate a couple of maneuvering simulations. The simulation and maneuver parameters are loaded into Matlab by the initialization file "start.m". In the first case we have a typical maneuver without singularity problems. In the second case a singularity occurs while maneuvering. The file "pl.m" plots the simulation results.

Simulation 1 A typical 100 (degrees) maneuver (without singularity)

In this case we are commanding the spacecraft to rotate 100º in an arbitrary direction (1, 1, -1). The algorithm uses a singularity proximity measure, which is the determinant of the AA^T matrix. When this measure becomes very small it is an indication of singularity occurrence. In this case we hit a low point at 10 seconds but it is not a real singularity. The spacecraft maneuvers smoothly towards the commanded direction maintaining an almost constant eigenaxis, body rate, and CMG momentum. The CMG torque and phase-plane show an acceleration pulse in the beginning and at the end of the maneuver. The deceleration is not as high and lasts longer than the acceleration. Shortly before the 30 seconds when the error becomes small the "kay" factor drops to zero which turns on the PID controller that keeps the steady-state error in the 10^{-7} level. The four gimbal rates versus gimbal angles are also shown in phase-plane. Notice that the system momentum is maintained at zero throughout the maneuver.




100 deg maneuver in [1,-1, 1] direction, 4 SGCMG Control
CMG Momentum in Body (ft-lb-sec)
Roll
Pitch
Yaw
Time
Control Torque (ft-lb)
Time
body rates (deg/s)
Maneuvering
PID Control
Time
x 10^-12
System Momentum is Zero since there are no external torques
System Momentum, (ft-lb-sec)
Roll
Pitch
Yaw
Magnitude
Time


100 deg maneuver in [1,-1, 1] direction, 4 SGCMG Control
Phase-Plane
Rate Error (deg/s)
Time
Gimbal Rate (deg/s)
Gimbal Angle (deg)
x 10^18
Singularity Indicator
det(A*A^T)
Time
kay
Maneuvering
PID Control
Time

3.4.3 Adding Flexibility to the Four CMG Simulation Model

This time we will go one step further and include structural flexibility. The Simulink model in this example is "*MaxEn_4SGCMG_Gimbals_Flex.mdl*", shown in Figure (3.4.3.1). The Matlab and other data files used are in directory "*…\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(c) Flex 4SGCMG ACS w Gimbal*". It is very similar to the previous model, described in section 3.4.2, but it includes structural bending. The attitude control law and steering logic are the same, but some of the control parameters were adjusted to accommodate flexibility. The simulation data are loaded into Matlab by executing file "Start.m", and the file "pl.m" plots the simulation results, as in previous examples.

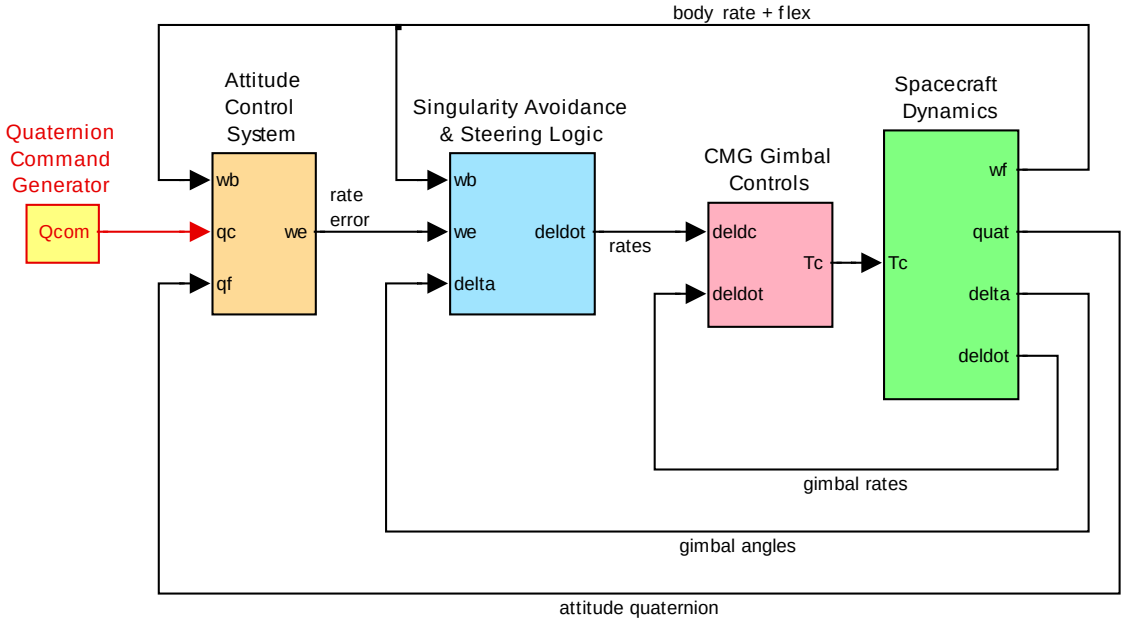


Figure 3.4.3.1 Simulation model "MaxEn_4SGCMG_Gimbals.mdl", consisting of rigid plus flexible spacecraft dynamics and 4 SG-CMGs

As you can see in Figure 3.4.3.2, the main difference of this simulation model from the previous one shown in figure 3.4.2.3 is that the spacecraft dynamics (green) block includes flexibility. The structural flexibility subsystem is a state-space model shown in detail in Figure 3.4.3.3. It is loaded from file "*flex_only_fem_s.m*" which was created by Flixan "flex spacecraft modeling program" in section 1.1 and its title is "*Flex Only Spacecraft with RCS and CMG*". It has the rigid-body modes removed because the rigid-body dynamics is implemented in Matlab function "*SC_CMG_Gimbal.m*", which contains also the CMG dynamics. The coupling between the rigid and flex models is not straightforward. The 4 CMG gimbal torques drive the rigid-body dynamics as they are controlled by the 4 gimbal servos attempting to maintain the commanded gimbal rates. The torque motors have to be strong enough because in addition to the CMG load inertia they also have to fight the gyroscopic torque $h_{cmg}\left(\dot{\theta}\sin\delta - \dot{\phi}\cos\delta\right)$ which is not small.

The inertial gimbal rates $\dot{\delta}_i$ rotate the CMG momentum vectors generating torques which produce body rates, and attitudes. The combined CMG torque in vehicle axes (T_{cmg}) is also

Initialization File "Start.m"

The initialization file "start.m" initializes the simulation. It loads the vehicle mass properties in body coordinates which are transformed to CMG pyramid coordinates, see Figure(). It also loads the flexibility state-space model, the transformation matrices, CMG parameters, orientation angles, gimbal rate limits, and gains for the gimbal control system. It initializes also the spacecraft state-vector, sets torque limits, max acceleration and rate limits, and parameters for the singularity avoidance logic.

```
global J Jinv m ref quad hcmg d2r r2d wcmg zeta
global Acc_Lim Rat_Lim kr ki kp Es Ps Fs wlim mx
global bet gam Jgi Jsi Joi
d2r= pi/180; r2d=180/pi;

[Af, Bf, Cf, Df] = flex_only_fem_s;              % Load Flex Only Spacecr Dynamics
J= [0.17E+94, -0.16e+93,  0.11E+92;              % Vehicle MOI matrix in (slug-ft^2).
   -0.16e+93,  1.32E+94,  0.31E+92;
    0.11E+92,  0.31E+92,  1.41E+94];
Tc2b= [0, 0, 1; ...                              % Transformation Matrix
       0, 1, 0; ...                              % from CMG to Body Axis
      -1, 0, 0]; Tb2c= inv(Tc2b);
J= Tb2c*J*Tc2b;  Jinv= inv(J);                   % Convert Inertias to CMG axes
Ctr= Tb2c*[0 1 0; 1 0 0; 0 0 -1];                % Transform Matrix (append to body to CMG
axis)

Jsi= 1.2; Jgi= 0.6; Joi= 0.8;                    % CMG Inertia about Spin, Gimbal, Outp axes
wcmg=500; zeta= 0.95; lamb=0.5;                  % CMG servo bandwidth (rad/s), damping coeff
Kii=lamb*Jgi*wcmg^3;                             % CMG Servo Gains
Ka=(1+2*zeta*lamb)*Jgi*wcmg^2;
Kb=(2*zeta+lamb)*wcmg/Ka;
hcmg=[1, 1, 1, 1]*1200;                          % CMG Momentum capability

% CMG Geometry
bet=68;                                          % Pyramid Beta angle
gam=[90, 180, 270, 0];                           % Pyramid Gamma angles
[m, ref, quad]= M4(bet);                         % CMG Gimbal, Spin, & Torq direction matrices

sigma0=[0; 0; 0; 0]*d2r;                         % Initial CMG Gimbal positions (rad)
wb0= [0; 0; 0];                                  % Initial body rates
h0= [0; 0; 0];                                   % Initial CMG Momentum (body)
deldot= [0 0 0 0]';                              % Initial Gimb Rates
delta = [0 0 0 0]';                              % Initial Gimb Angles
Qt0=[0; 0; 0; 1];                                % Initial Quaternion
ini= [wb0',h0',deldot',delta',Qt0'];             % State Integrator Initialization
wlim= 3.6*d2r;                                   % MaxEn ACS Rate Limit 3.9 (deg/s)
Tmax= 650;                                       % Max CMG Torque 650
mx= Tmax*[1,1,1]./[J(3,3),J(2,2),J(3,3)];        % Max accelerations x,y,z
Acc_Lim=30*d2r; Rat_Lim=12*d2r;                  % Steer Law Accel & Rate limit (deg/sec)
Tglim=120; Wclim=100*d2r;                        % CMG Model Gimbal Accelerat and Rate Limits
ki=0.07; kp=1.4; kr=10;                          % PID Gains: ki=0.3; kp=7.3; kr=12
Es=0.05; Ps=1.0e18; Fs=0.2;                      % Singul Avoid Param Es=0.01 Ps=1.0e11 Fs=0.2
xlim= inf(1,18); xlim(7:10)=Wclim*[1 1 1 1];     % State Integrator Limits

com_dir=Tb2c*[-1; 1; 1];                         % Command Direct Unit Vector (body)
com_dir=com_dir/sqrt(com_dir'*com_dir);          % Unit Vector [0.7; -0.6; 0.4]
```

Frequency Response Stability Analysis

The following model "*Open_Loop.mdl*" is similar to the non-linear simulation model but it has the control laws linearized and it is used to perform linear stability analysis. It uses the linearized functions "*Lin_ACS.m*" for attitude control, and "*Lin_Steering.m*" for steering. The control loop is broken for frequency response analysis at the rate error command to the steering logic. Only two of the control loops, pitch, or yaw, are analyzed. In the yaw analysis example shown in Figure 3.4.3.8, the yaw loop is opened, but the roll and pitch loops are closed. The Matlab file "freq.m" calculates the frequency response between the input and the output and plots the Bode and the Nichols plots. The stability analysis plots are shown in Figure 3.4.3.9. Stability is measured by the phase and gain margins from the red cross. This model is also used for tuning the PID gains to maximize stability. The ACS bandwidth was reduced from the previous rigid-body analysis to avoid exciting low frequency flex modes to instability. Compensator filters were also included in the ACS. A low-pass filter was also included to filter out the error signal which turns on the PID to provide a smoother transition between maneuvering and PID control.

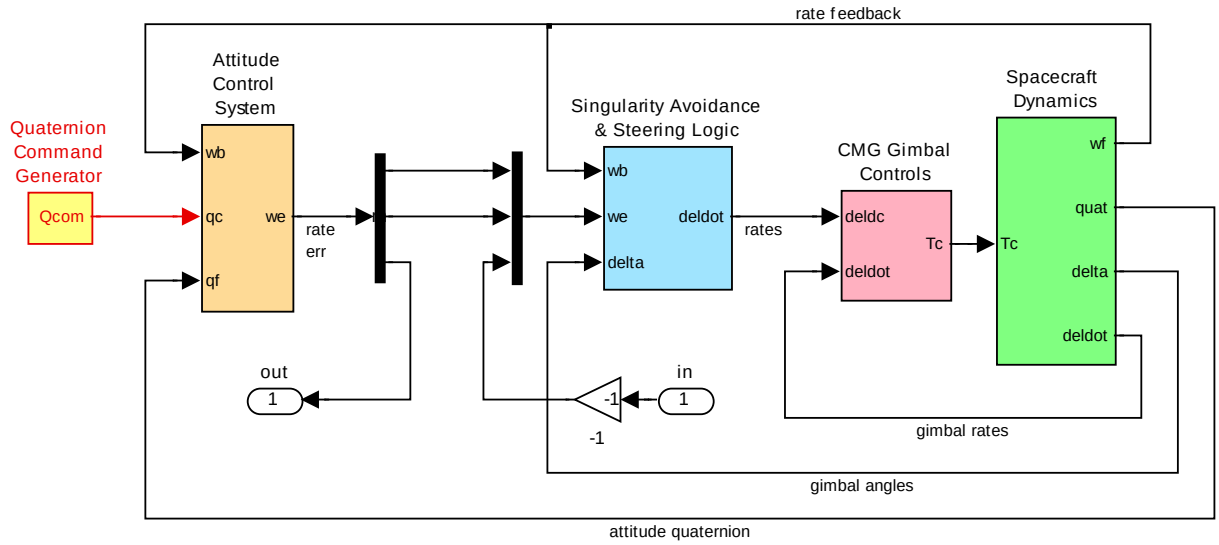


Figure 3.4.3.8 Simulink Model "Open_Loop.mdl" used for frequency response stability analysis, shown in this case for yaw axis analysis

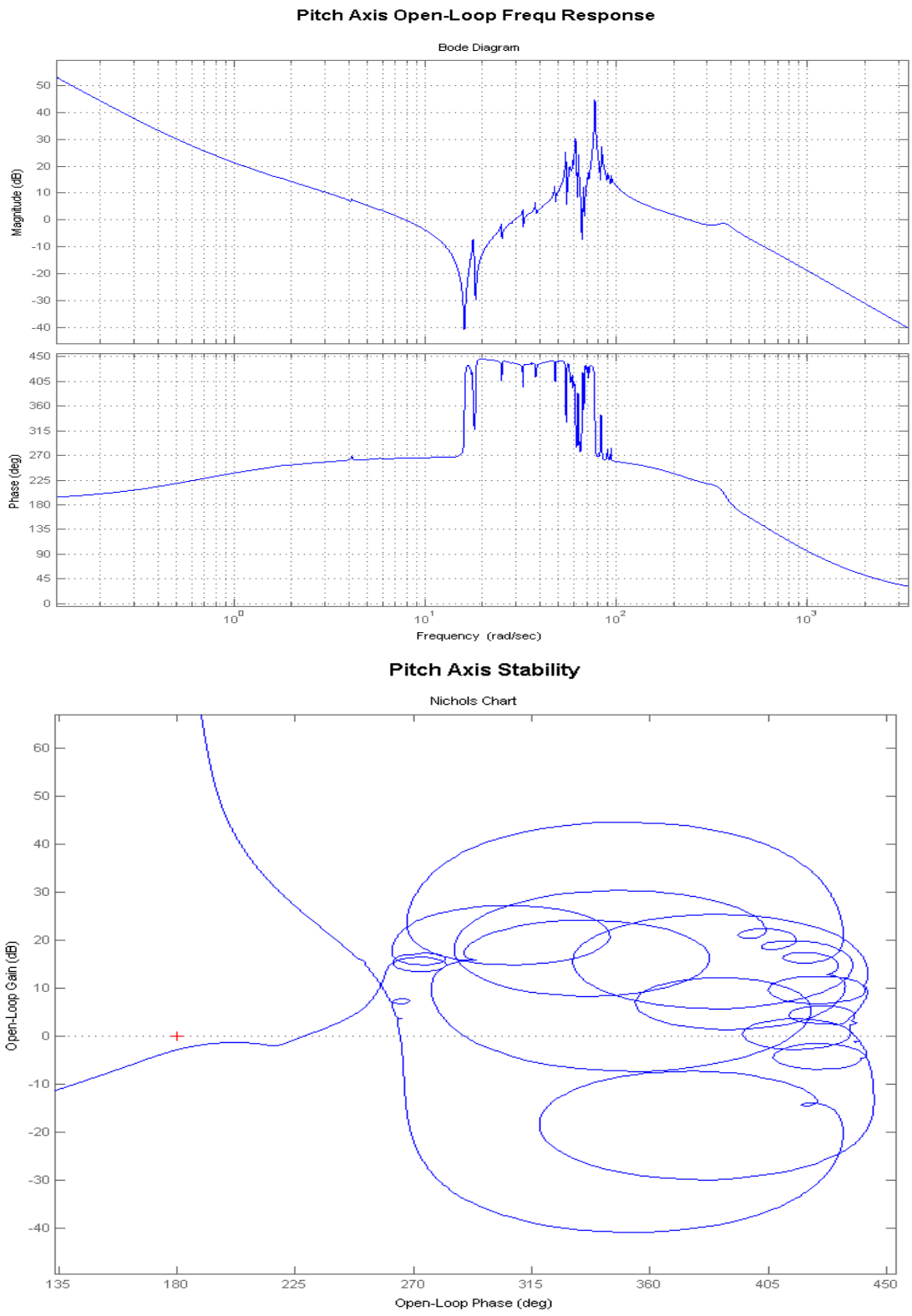


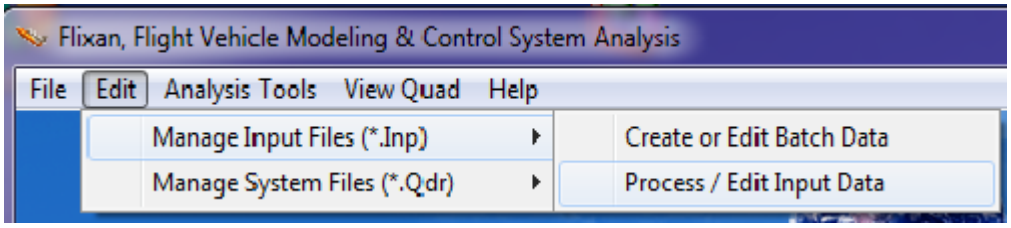
Figure 3.4.3.9a Bode and Nichols Plots showing stability margins of the Pitch axis PID system

3.5 Using the Flight Vehicle Modeling Program to Analyze the Four SGCMG System

In the previous section we used spacecraft and reaction wheel non-linear models developed in Matlab and then we coupled them with state-space flexibility models that were developed using the Flixan Flex Spacecraft program. Now we will use the Flixan Flight Vehicle Modeling Program (FVP) to create similar models of the spacecraft coupled with four single gimbal CMGs, including structural flexibility, and hopefully the results will match with the results obtained from the previous section in folder "*(c) Flex 4SGCMG ACS w Gimbal*". We will first show how to obtain the spacecraft state-space models by using the existing data files and running the FVP in batch mode, and then go into details to show how we create the spacecraft and flex data from scratch. The data for this analysis is in directory "*C:\Flixan\Examples\Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP* ". The input data file is "*FlexSc_CMG_FVP.Inp*" the systems file is "*FlexSc_CMG_FVP.Qdr*". The modal data files are the same as previously in the RCS analysis "*FlexSc_FEM.Mod*" and "*FlexSc_FEM.Nod*". The input data file contains data for two spacecraft with 4 SGCMGs, a rigid model and a flex model. A set of modal data consisting of 40 selected flex modes is included at the bottom of the input data file. Some unused states and outputs are eliminated using the Flixan truncation utility. The Matlab analysis is performed in the sub-directory "Matan".

Creating the Systems in Batch Mode

On the top of the input data file there is a batch data-set "*Batch for Spacecraft with 4 SG CMG*". It is a short script of commands that speeds up the generation of the spacecraft systems. To run the batch, first start the Flixan program, go to folder "*…Flex Agile Spacecraft with SGCMG & RCS\CMG Control\(e) 4SGCMG using FVP*". Go to "*Edit*", "*Manage Input Files*", and then "*Process/ Edit Input Data*".



When the following dialog appears, click "Continue".

Robustness Analysis of the Four SG-CMG Spacecraft

Robustness is the ability of the system to tolerate uncertainties and variations, either internal or external. Our next step is to analyze the spacecraft control system's robustness to internal parameter variations. The question is, how much parameter variations can a system tolerate before it becomes unstable, or stops performing properly? Parameter uncertainties are imprecise knowledge of the plant model parameters, such as in this case, the mass properties, moments of inertia, CMG momentum, momentum direction, gimbal angle and direction, etc. The structured uncertainties in a model are specified in terms of variations in the actual plant parameters above and below their nominal values. We will use the IFL method to "pull" the uncertainties out of the plant M(s). The uncertainties are represented by a diagonal block Δ that is connected to the plant by means of some additional inputs and outputs, as shown in Figure (1). An input/ output pair for each parameter variation (δ_i).

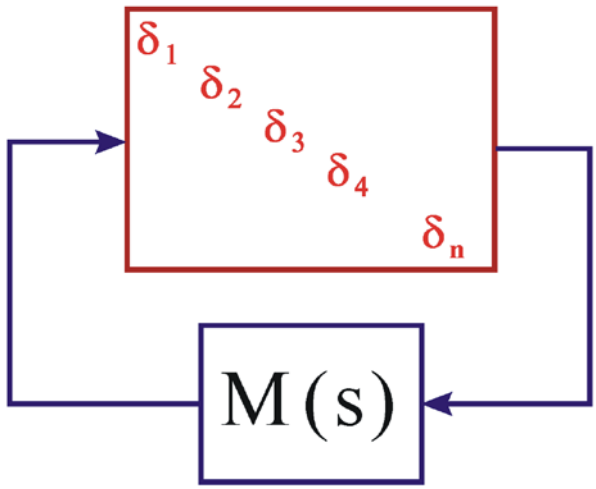


Figure 1 Closed-Loop Plant M(s) with the uncertainties block "pulled out"

The IFL methodology is implemented in Flixan. The program reads the nominal plant parameters for the spacecraft and the CMGs from an input file. It also reads the uncertainties for some of the parameters from the same file. For every non-zero parameter variation the program creates an additional input/ output pair that is supposed to hook up to a (δ_i) element in the block. The magnitude of each element represents the maximum possible variation of the parameter above or below its nominal value. In essence we create (n) additional inputs and outputs to the plant that connect to the uncertainties block Δ, which is a block diagonal matrix $\Delta = \text{diag}(\delta_1,\delta_2,\delta_3,...\delta_n)$. For convenience the diagonal block Δ is normalized so that its individual elements now vary between +1 and -1. We also normalize the plant M(s) by scaling its input/ output elements as needed to connect with the normalized Δ whose elements are now bounded to ± 1. The individual elements of Δ may be scalars or matrices and each represents a real

uncertainty in the plant. Some parameter variations which are higher than rank-1 dependency may generate two or three (δ_i). An Ixz cross-product of inertia, for example, will couple in both, roll and yaw axis, creating two separate δ_i's, one in roll and another in yaw axis. In this case we treat them as two separate uncertainties. M(s) represents the known dynamics consisting of the plant model with the control system in closed-loop form. The augmented state-space system is used to perform robustness analysis using μ. The system in this configuration is defined to be robust if it remains stable despite all possible variations in the Δ block as long as the magnitude of each individual variation is bounded below $\pm\delta_{(i),}$ or ±1 in the normalized system. The structured singular value (μ) is the perfect tool for analyzing this type of robustness problems in the frequency domain. The value of 1/μ(M) represents the magnitude of the smallest perturbation that will destabilize the normalized system M(s).

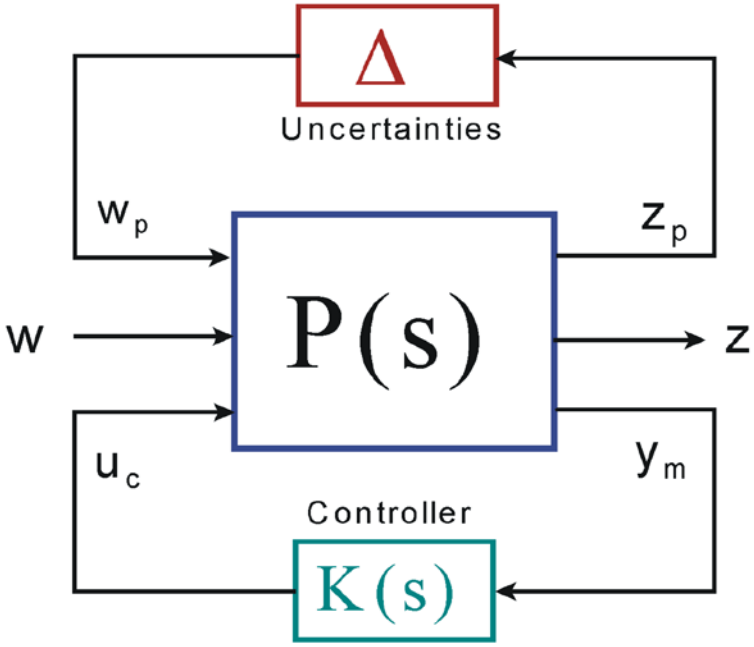


Figure1a Robustness Analysis Model

The block diagram formulation, shown in Figure (1a), is used for μ-synthesis or robustness/ performance analysis using (μ) methods. The plant has inputs and outputs that connect to the uncertainty block. It also has inputs and outputs that connect to the control system K(s). It also has external disturbances (w) and output performance criteria (z) which are also normalized to unity. We say that the plant meets sensitivity requirements when the mu frequency response between (w) and (z) is less than one at all frequencies. Similarly, and according to the small gain theorem, the closed-loop system is robust to the specified uncertainties as long as μ(M) across the normalized block Δ, (that is, between w_p and z_p) is less than one at all frequencies. Robust performance is when it

The batch performs the following operations. It creates a nominal spacecraft model with 40 flex modes, title: "*Flexible Agile Spacecraft with 4 SG-CMG*" in file "*FlexSc_4CMG.Qdr*". It also creates the perturbation model with the 61 uncertainties, title: "*Flexible Agile Spacecraft with 4 SG-CMG (Uncertainties)"*. The two state-space models are then converted to Matlab system functions, "*sc_4cmg_flex.m*" and "*sc_4cmg_flex_unc.m*", respectively that can be loaded into Matlab. Both systems contain roll, pitch, and yaw coupled vehicle dynamics. The second system, in addition to the standard inputs and outputs of the first system, it includes also the 61 pairs of inputs and outputs that connect to the uncertainties Δ block. This is the diagonal block that contains the normalized uncertainties which vary between -1 and +1. Some of the uncertainties couple only in one axis, but some uncertainties couple in more than one axis.

Simulation Model

Now, let us take a look at a linear simulation model that uses the nominal spacecraft dynamics without parameter variations. This simulation model is in file "*Lin_Flex_Sim.mdl*", shown in Figure (2). The model is initialized using file "start.m", which also loads the two spacecraft systems into Matlab workspace.

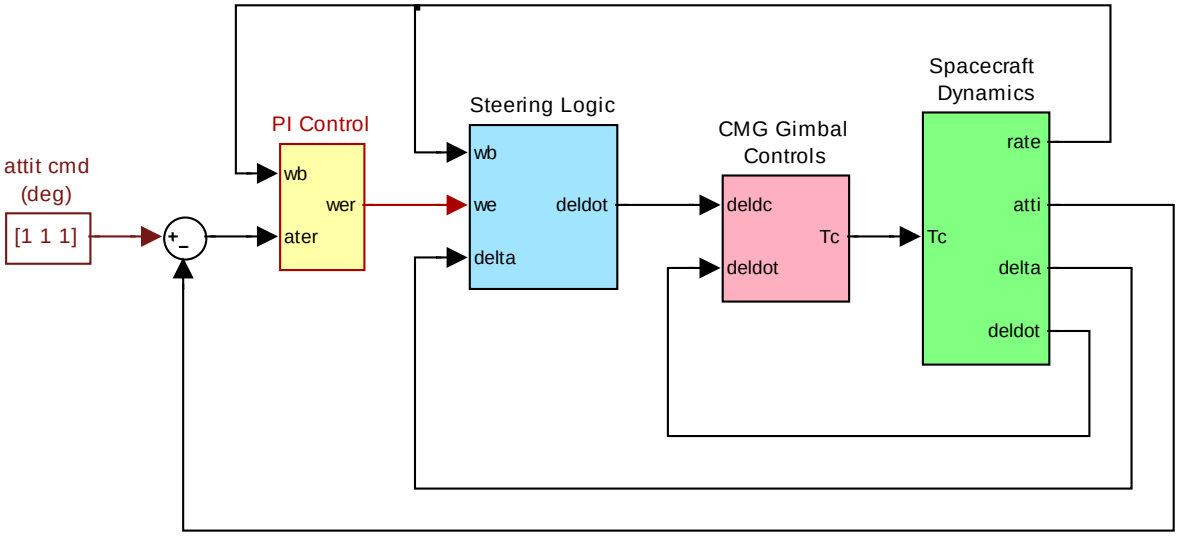


Figure 2 Simulink model for the Flex Agile Spacecraft with 4 SG-CMGs, in file "*Lin_Flex_Sim.mdl*"

The spacecraft dynamics (green) block consists of the nominal state-space system "*sc_4cmg-flex.m*" (no uncertainties). The input is a vector of four CMG gimbal torques which control the gimbal rates. The outputs are: spacecraft attitude, rates, CMG gimbal angles, and gimbal rates. The gimbal rate commands come from the CMG steering logic, and the gimbal rate control system provides the gimbal torques required to control the rates. The purpose of the steering logic is to control the spacecraft rate by creating gimbal rate commands at the 4 CMG gimbals. The inputs to the steering logic are: spacecraft rates, gimbal angles, and spacecraft rate error. The attitude control system is a simple PI. The (D) part of the PID is included in the steering. In the simulation the spacecraft is commanded to perform a one degree rotation in all 3 directions. The purpose of the simulation is to demonstrate nominal system stability.

Figure 7 Spacecraft System from "*sc_4cmg_flex_unc.m*" that includes the uncertainty inputs and outputs.

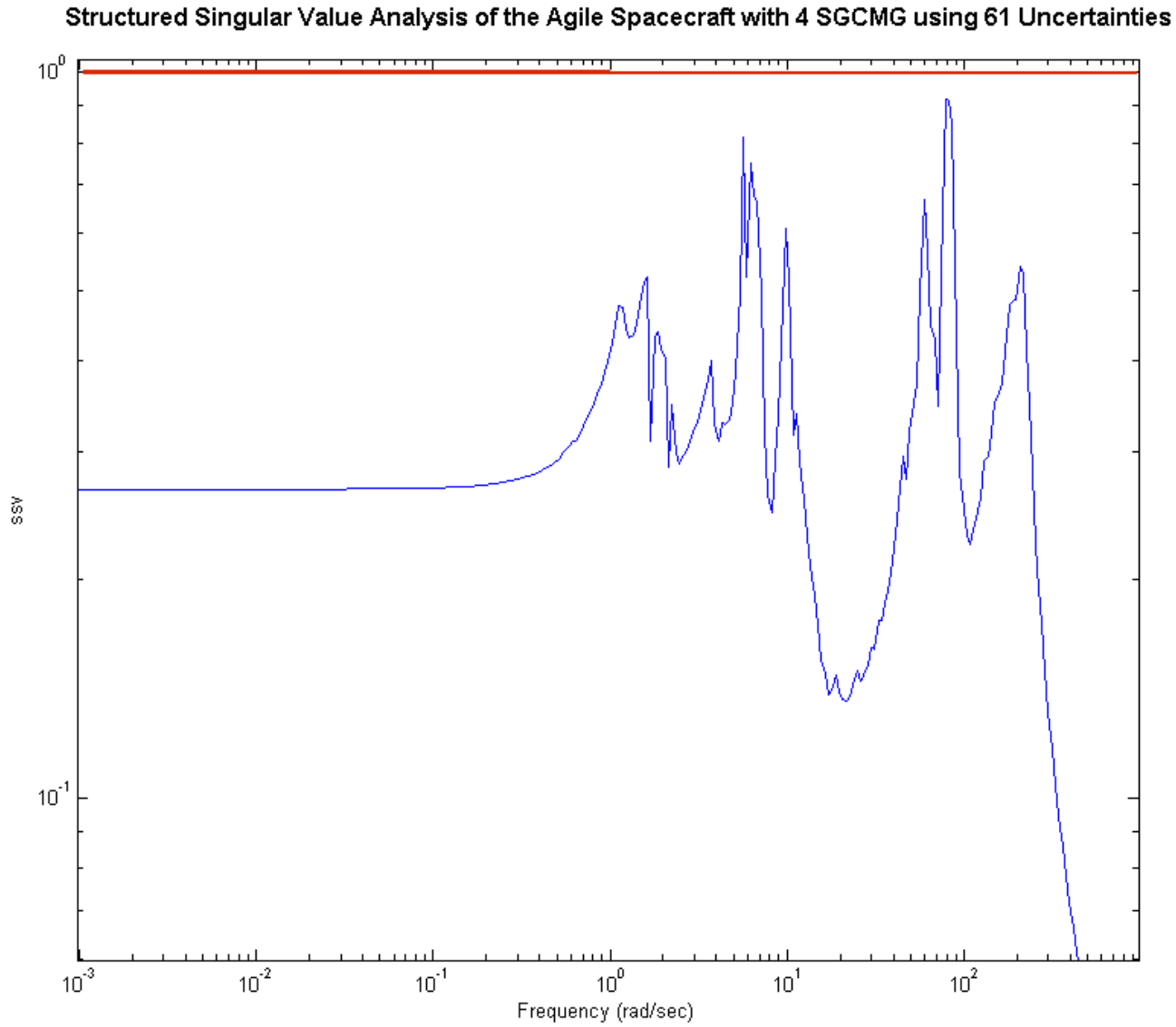


Figure 8 The Structured Singular Value frequency response across the perturbations block Δ is less than one at all frequencies, therefore, system is robust to parameter variations.

3.6 Spacecraft Configuration Using 2 SG-CMG and a RW

In the previous section we analyzed spacecraft configurations using four SGCMG that perform 3-axis attitude control of the spacecraft for quick maneuvering between targets. The control law is efficient, symmetrical and fast because it uses the max control torque and momentum capability of the CMG devices as it performs eigenaxis maneuvers, assuming that the pointing requirement is the same in all directions. But what if the maneuvering requirements on the spacecraft are not the same in all directions? For example, if we want to point an antenna or a beam of light (which is along the spacecraft x axis) in a certain direction we are more interested in the pitch and yaw efficiency of response and much less in roll, since roll errors do not affect antenna operation. In addition, the spacecraft which is normally pointing nadir (towards the earth center) is required to maneuver not more than 40 degrees from nadir. Furthermore, the spacecraft moment of inertia in roll is much less than in pitch and yaw axes, so the torque and momentum requirement in roll would be significantly less than pitch and yaw. It is conceivable, therefore, and since the SGCMG are very costly devices that we may be able to get by with fewer momentum control devices.

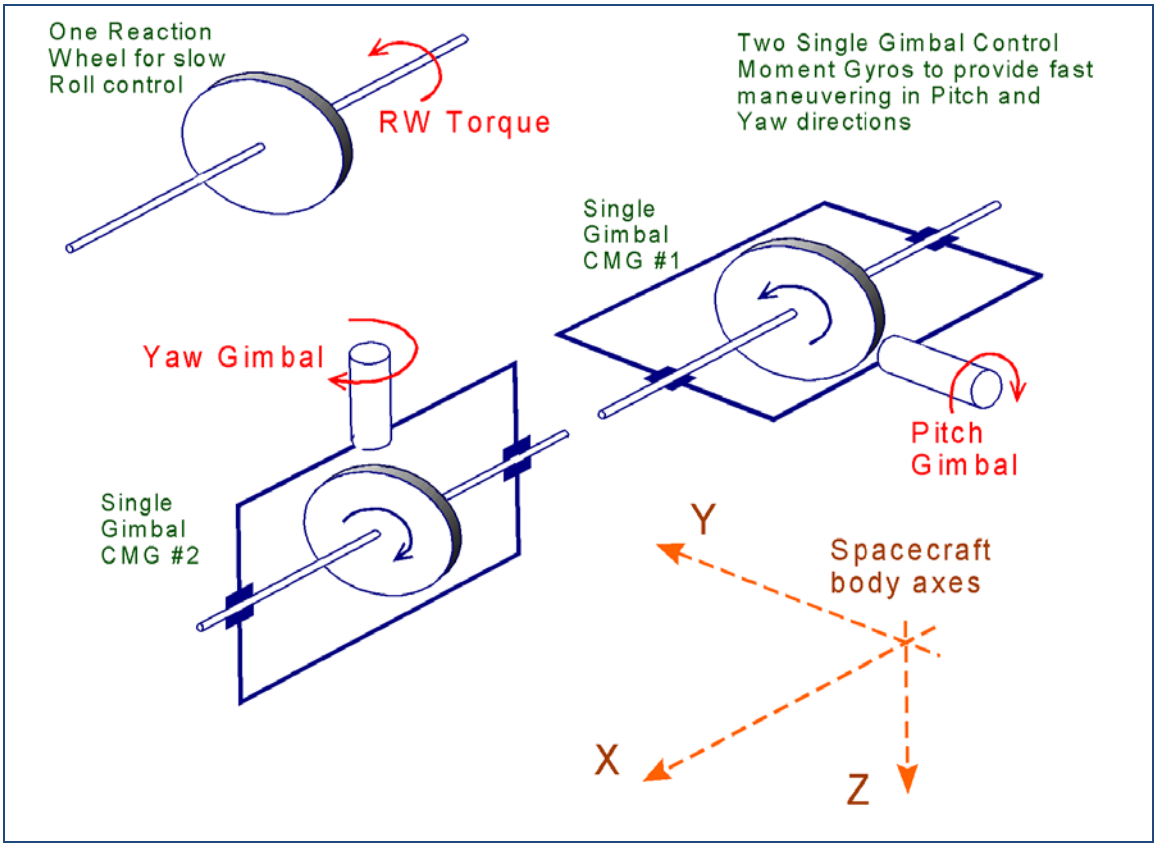


Figure 3.6.1 The Momentum Control System in this configuration consists of one Reaction Wheel combined with two Single-Gimbal CMGs

The Momentum Control System (MCS) described in this configuration consists of two single-gimbal CMGs for pitch and yaw and only one reaction wheel for roll control, see Figure (3.6.1). The RW momentum (spin) direction is parallel to the spacecraft roll axis and controls roll attitude. The two SG-CMGs are spinning at constant rate and their initial (nominal) momentum directions are along the +x and –x axis respectively, cancelling each other's momentum. Between operations there are occasional momentum dumps using the RCS jets to prevent the MCS momentum from reaching saturation. The momentum desaturation system is attempting to keep the CMG gimbals in the nominal position and the RW speed at zero. Each CMG gimbals only in one direction relative to the spacecraft and their momentum directions vary as they gimbal. CMG #1 is gimbaling in pitch, and initially (when the gimbal is at zero and the momentum is along x) it generates a yaw torque. The second CMG is gimbaling in yaw and it generates a pitch torque when the gimbal is in its nominal zero position. When the CMGs are gimbaling at bigger angles, rolling torques are also generated which are counteracted by the reaction wheel control system. The torque output direction of each CMG varies. It is orthogonal to the momentum vector and the gimbal direction, and since the momentum direction is constantly changing the steering algorithm must keep track of the gimbal angles (δ_i) and calculate the gimbal rate commands for the two SG-CMGs. It calculates also the roll torque command for the RW.

The Line-of-Sight (LOS) direction of the antenna is along the spacecraft x-axis, and the main priority of the ACS is to point the LOS at the target as fast as possible, which requires pitch and yaw maneuvering. Positioning the spacecraft in roll is a secondary priority and it is acceptable if it takes longer to converge in roll in comparison to pitch and yaw. Attitude maneuvering in pitch and yaw should, therefore, be performed faster using the CMG’s, while the roll axis is controlled by the reaction wheel system which is slower. The CMG torque capability in pitch and yaw and the control system bandwidth is much greater than the RW torque and bandwidth controlling the roll axis and the MCS, therefore, cannot perform ideal eigenaxis maneuvers as it was demonstrated by the 4 SG-CMG control system in previous sections. For comparison purposes, the LOS pointing error will be shown separately from the roll error.

Spacecraft Dynamics

The spacecraft rotational acceleration $\dot{\omega}$ is a function of the internal reaction wheel torque (T_{RW}), the CMG internal torque ($\underline{T}_{CMG}$), and also the external torques (T_{ext})

$$J_{sc}\underline{\dot{\omega}} = \underline{T}_{CMG} + \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T T_{RW} - \underline{\omega}\times\left(J_{sc}\underline{\omega}\right) + \underline{T}_{ext}$$

Where: $\underline{T}_{CMG}$ is the CMG torque and T_{RW} is the reaction wheel torque in spacecraft body. The CMG torque is a function of the gimbal rates and angles as defined in equations (2.2 and 2.4). It consists of torques transmitted through the gimbals and gyroscopic torques transmitted through the bearings. The RW motor speed control dynamics is ignored and we assume that the RW torque is equal to the commanded torque from RW steering, but it is limited to less than ±10 (ft-lb).

The reaction wheel rate of change of momentum is a function of the reaction wheel torque which is in the x direction. I_W is the wheel moment of inertia about the spin axis.

$$\underline{\dot{H}}_{RW} + \omega\times\underline{H}_{RW} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T T_{RW}$$

The reaction wheel spin rate in (rad/sec) is

$$w_{RW} = \frac{H_{RW}(1)}{I_w}$$

The CMG rate of change of momentum is a function of the internal CMG torque ($\underline{T}_{CMG}$). The control torque experienced by the spacecraft in body axes due to gimbaling (δ) is ($-\dot{H}_{cmg}$), which is the rate of change in the CMG momentum.

$$\underline{\dot{H}}_{CMG} + \underline{\omega}\times\underline{H}_{CMG} = -\underline{T}_{CMG}$$

$$-\underline{\dot{H}}_{CMG} = \underline{T}_{con} = -\left[A(\delta)\right]\underline{\dot{\delta}}$$

The (2x3) matrix [A] consists of two column vectors ($\underline{a}_i$) which are functions of the gimbal angles (δ_i). They are also functions of the CMG quad and reference directions (q_i and r_i).

$$A(\delta) = \left(\underline{a}_1 \quad \underline{a}_2\right) \; where: \; \underline{a}_i = \left(\underline{q}_i\cos\delta_i - \underline{r}_i\sin\delta_i\right)h_{CMG(i)}$$

The two CMG gimbal directions are in pitch and yaw:

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

The two CMG momentum reference directions (initially facing in opposite x directions at zero gimbal angles) are:

$$r = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The quad directions from the cross-product ($\underline{m}_i$ x $\underline{r}_i$) are:

$$q = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}$$

The CMG gimbal rates are controlled by a servo system that generates gimbal torques. The servo torques are attempting to counteract the gyroscopic disturbance torques created by the spacecraft rate. $\dot{\theta}$ *and* $\dot{\phi}$ are the spacecraft rates resolved about the CMG reference and quad axes, as defined in equations (2.3). The CMG gimbal inertial acceleration is obtained by integrating the gimbal moment equation below, ignoring friction. T_{gi} is the motor torque applied at the gimbal. Even though the CMG moment of inertia about the gimbal J_g is relatively small, the gyroscopic moment caused by the CMG momentum h_{cmg} coupling with spacecraft rate is a big torque, requiring a powerful gimbal servo-motor in order to control the gimbal rate.

$$J_g \ddot{\delta}_i + h_{cmg(i)} \left(\dot{\theta} \sin\delta - \dot{\phi} \cos\delta \right) = T_{gi}$$

We also check the simulation by calculating the system momentum which is always constant. In this case it's zero because it is initialized at zero.

$$\underline{H}_{sys} = J_{sc}\,\underline{\omega} + \underline{H}_{CMG} + \underline{H}_{RW} = const.$$

This dynamic model is implemented in Matlab function "*SC_RW_2CMG_Dyn2.m*". There is also a simple dynamic model implemented in Matlab function "*SC_RW_2CMG-Dyn1.m*" that has simplified gimbal dynamics. The gimbal torques are not calculated in this model, but the gimbal rates are assumed to follow the commanded rates. The servo system is replaced by a 12 Hz second order CMG gimbal rate control model. These dynamic models are used in separate simulations. The attitude quaternion is updated using body rate with flexibility included (w_t= w + w_f). Flex rate at the gyro (w_f) is provided by the flex state-state model that gets excited by the MCS torque.

```
function xdot= SC_RW_2CMG_Dyn2(x,Tci,Tw,Td,wf)
global nc d2r r2d
global J Jinv Iw Jsi Jgi Joi hcmg
%-------------------------------------------------------------
% State Variables (x)
% x(1-3)   = Body rates   (w)   (rad/sec)
% x(4-7)   = Quaternion
% x(8:10)  = h (CMG momentum)
% x(11:13) = RW Momentum
% x(14:15) = deldot
% x(16:17) = delta
% Inputs:
% Tci(2)   = Gimbal Torques (ft-lb)
% Tw(1)    = Reaction Wheel Torque in spacrft x-axis, (ft-lb)
% Td(3)    = Disturbance Torque (ft-lb)
% wf(3)    = Flex rate only from FEM system
%-------------------------------------------------------------
xdot= zeros(30,1);
w=  x(1:3);                                              % S/C Body rates (rad/sec)
qt= x(4:7);                                              % S/C Quaternion Attitude
h = x(8:10);                                             % CMG Momentum
hrw= x(11:13);                                           % React Wheel Momentum (body)
deldot= x(14:15); delidot=deldot;                        % Gimbal Rates relatv to s/c
delta = x(16:17);                                        % Gimbal Angles (rad)

wt= w+wf;                                                % Total body rate + flex
wr = hrw(1)/Iw;                                          % React Wheel Rate (rad/sec)
Twi= [Tw, 0, 0]';                                        % Wheel Torque Vector (ft-lb)
hs = J*w + h + hrw;                                      % System Momentum (hs)
[Pj,thd,phd,ddd]= Transforms(delta*r2d,w);               % SC Rates al ref, quad, gmb

Tcmg=zeros(3,1);                                         % Calc CMG Torque in Body
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    Mj= [Tci(i); ...                                     % CMG Torque in CMG axis
    deldot(i)*((Jsi-Jgi)*(thd(i)*cd+phd(i)*sd) +hcmg(i)); ...
    deldot(i)* (Jgi-Joi)*(phd(i)*cd-thd(i)*sd)];
    Tcmg= Tcmg - Pj(:,:,i)*Mj;                           % Torque on Vehicle
    delidot(i)= delidot(i) + ddd(i);                     % Delta_Inertial_dot (rad)
end

xdot(1:3)= Jinv*(Tcmg +Twi - cross(w,J*w) );             % Vehicle acceleration
xdot(4:7)= 0.5*[0     wt(3)  -wt(2)  wt(1);              % Quaternion Update
             -wt(3)   0       wt(1)  wt(2);              % includes flex)
              wt(2)  -wt(1)   0      wt(3);
             -wt(1)  -wt(2)  -wt(3)  0]* qt;
xdot(8:10)= -Tcmg - cross(w,h);                          % Hcmg_dot
xdot(11:13)= -Twi - cross(w,hrw);                        % R-Wheel Momentum dot
for i=1:nc
    sd=sin(delta(i)); cd=cos(delta(i));
    xdot(13+i)= (Tci(i)-hcmg(i)* ...                     % Gimbal acceler delta-ddot
        (thd(i)*sd-phd(i)*cd))/Jgi;
    xdot(15+i)= x(13+i);                                 % Gimbal rates  delta-dot
end

% Additional Outputs
xdot(18:20)= hs;                                         % System Momentum
xdot(21:23)= wt;                                         % Total Vehi rate
xdot(24:26)= h + hrw;                                    % CMG + RW Momentum
xdot(27:29)= Tcmg + Twi;                                 % CMG + RW Torques on s/c
xdot(30)   = wr;                                         % Wheel Rate (rad/sec)
```