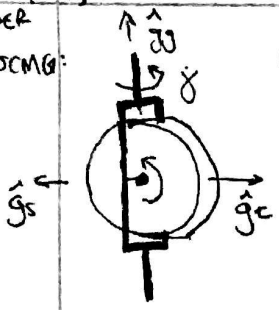
1. #1a) PADRAIG J. LYSANDROU

CONSIDER the VSCMG:



we shall define the G frame

$G: \{\hat{g}_s, \hat{g}_t, \hat{g}_g\}$ Note $\hat{g}_g$ is fixed. (IN BODY)

$\therefore \vec{\omega}_{G/B} = \dot{\gamma}\hat{g}_g$ for gimbal rate $\dot{\gamma}$

we shall define the W wheel frame

$W: \{\hat{g}_s, \hat{w}_t, \hat{w}_g\}$ which is NOT AS useful.

$\therefore \omega_{W/G} = \Omega \hat{g}_s$ for rate Ω about SPIN axis

& we shall define the inertia characteristics:

${}^G I_G = {}^G\begin{bmatrix} I_{gs} & & \\ & I_{gt} & \\ & & I_{gg} \end{bmatrix}$; ${}^W I_W = {}^W\begin{bmatrix} I_{ws} & & \\ & I_{wt} & \\ & & I_{wt} \end{bmatrix}$ → Assume disc symmetry.

we can ALSO THEREFORE DEFINE THE ROTATION ${}^B_G A = [\hat{g}_s \ \hat{g}_t \ \hat{g}_g]$

& $\therefore {}^B I_G = [{}^B_G A]\,{}^G[I_G]\,[{}^B_G A]^T$

WE CAN SUM UP THE ANGULAR MOMENTUM OF EACH BODY TO GET THE FOLLOWING:

$\vec{H}_{total} = H_B + H_G + H_W$ → $H_W = [I_W]\,\omega_{W/N}$

$H_G = [I]_G\,\omega_{G/N}$

↳ Body: $H_B = [I_S]\,\omega_{B/N}$ → I_S INCLUDES VSCMG inertia as POINT MASSES.

Let us focus on $H_G = {}^B[I_G](\omega_{G/N}) = {}^B[I_G](\omega_{G/B} + \omega_{B/N})$

Let us use our DCM to get an expression for ${}^B I_G\,\omega_{B/N}$

${}^B I_G = [{}^B_G A]\,{}^G I_G\,[{}^B_G A]^T = [{}^B_G A]\begin{bmatrix} I_{Gs} & & \\ & I_{gt} & \\ & & I_{gg} \end{bmatrix}\begin{bmatrix} \hat{g}_s^T \\ \hat{g}_t^T \\ \hat{g}_g^T \end{bmatrix} = \left(I_{gs}\hat{g}_s\hat{g}_s^T + I_{gt}\hat{g}_t\hat{g}_t^T + I_{gg}\hat{g}_g\hat{g}_g^T\right)$

for too many g's...

SUBSTITUTE & recall that $\vec{\omega}_{G/B} = \dot{\gamma}\hat{g}_g$

$\therefore H_G = \left(I_{Gs}\hat{g}_s\hat{g}_s^T + I_{gt}\hat{g}_t\hat{g}_t^T + I_{gg}\hat{g}_g\hat{g}_g^T\right)\vec{\omega}_{B/N} + \dot{\gamma}\hat{g}_g$

for SIMPLICITY WE SHALL DECOMPOSE $\omega_{B/N}$ AS SUCH: (CONSTANTS!) IN G FRAME

$\omega_s = \hat{g}_s^T\omega_{B/N}$; $\omega_t = \hat{g}_t^T\omega_{B/N}$; $\omega_g = \hat{g}_g^T\omega_{B/N}$ ⇒ SO $\boxed{\omega = \omega_s\hat{g}_s + \omega_t\hat{g}_t + \omega_g\hat{j}_t}$

$\therefore H_G$ becomes $\boxed{H_G = I_{Gs}\omega_s\hat{g}_s + I_{Gt}\omega_t\hat{g}_t + I_{Gg}(\omega_g + \dot{\gamma})\hat{g}_g}$ represented IN gimbal frame.

NOW LET US FOCUS ON THE WHEEL: w/ decomposition of $\vec{\omega}_{W/N} = \vec{\omega}_{W/G} + \vec{\omega}_{G/B} + \vec{\omega}_{B/N}$

$\vec{H}_W = [I_W](\vec{\omega}_{W/G} + \vec{\omega}_{G/B} + \vec{\omega}_{B/N})$

DCM on inertia works out the same:

${}^B I_W = I_{ws}\hat{g}_s\hat{g}_s^T + I_{wt}\hat{g}_t\hat{g}_t^T + I_{wg}\hat{g}_g\hat{g}_g^T$ & reuse $\vec{\omega}_{B/N}$ decomp for:

${}^B I_W\,\vec{\omega}_{B/N} = I_{ws}\omega_s\hat{g}_s + I_{wt}\omega_t\hat{g}_t + I_{wg}\omega_g\hat{g}_g$

& ${}^G I_W\,\vec{\omega}_{G/B} = {}^G\begin{bmatrix} I_{ws} & & \\ & I_{wt} & \\ & & I_{wt} \end{bmatrix}\begin{pmatrix} 0 \\ 0 \\ \dot{\gamma} \end{pmatrix} = I_{wt}\dot{\gamma}\hat{j}$

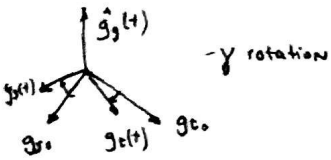
${}^W I_W\,\vec{\omega}_{W/G} = {}^W\begin{bmatrix} \ddots \end{bmatrix}\begin{pmatrix} \Omega \\ 0 \\ 0 \end{pmatrix} = I_{ws}\Omega\hat{g}_s$

$\therefore \boxed{H_W = I_{ws}(\omega_s + \Omega)\hat{g}_s + I_{wt}\left(\omega_t\hat{g}_t + (\omega_g + \dot{\gamma})\hat{g}_g\right)}$

1 CONT.

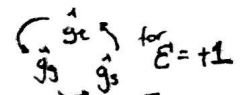
The gimbal axes can be tracked using these equations which rely upon the initial conditions:

$$\hat{g}_s(t) = \cos(\gamma(t)-\gamma_0)\hat{g}_s(t_0) + \sin(\gamma(t)-\gamma_0)\hat{g}_t(t_0)$$
$$\hat{g}_t(t) = -\sin(\gamma(t)-\gamma_0)\hat{g}_s(t_0) + \cos(\gamma(t)-\gamma_0)\hat{g}_t(t_0)$$
$$\hat{g}_g(t) = \hat{g}_g(t_0) \longrightarrow \text{recall axis is fixed.}$$



Let us perform vectorial derivatives

$$\frac{{}^B d}{dt}(\hat{g}_s) = \frac{{}^G d}{dt}(\hat{g}_s)^{\nearrow 0} + \vec{\omega}_{G/B} \times \hat{g}_s = \dot{\gamma}\hat{g}_g \times \hat{g}_s = \dot{\gamma}\hat{g}_t$$
$$\frac{{}^B d}{dt}(\hat{g}_t) = \frac{{}^G d}{dt}(\hat{g}_t)^{\nearrow 0} + \vec{\omega}_{G/B} \times \hat{g}_t = \dot{\gamma}\hat{g}_g \times \hat{g}_t = -\dot{\gamma}\hat{g}_s$$
$$\frac{{}^B d}{dt}(\hat{g}_g) = \frac{{}^G d}{dt}(\hat{g}_g)^{\nearrow 0} + \dot{\gamma}\hat{g}_g \times \hat{g}_g = \vec{0}$$



And further to inertial...

$$\frac{{}^N d}{dt}(\hat{g}_s) = \frac{{}^B d(\hat{g}_s)}{dt} + \vec{\omega}_{B/N} \times \hat{g}_s = \dot{\gamma}\hat{g}_t + \vec{\omega}_{B/N} \times \hat{g}_s = \dot{\gamma}\hat{g}_t + (\omega_s\hat{g}_s + \omega_t\hat{g}_t + \omega_g\hat{g}_g) \times \hat{g}_s$$
$$= \dot{\gamma}\hat{g}_t + \varnothing + -\omega_t\hat{g}_g + \omega_g\hat{g}_t = (\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g = \dot{\hat{g}}_s$$

$$\frac{{}^N d}{dt}(\hat{g}_t) = \frac{{}^B d}{dt}(\hat{g}_t) + \vec{\omega}_{B/N} \times \hat{g}_t = -\dot{\gamma}\hat{g}_s + (\omega_s\hat{g}_s + \omega_t\hat{g}_t + \omega_g\hat{g}_g) \times \hat{g}_t$$
$$= -\dot{\gamma}\hat{g}_s + \omega_s\hat{g}_g - \omega_g\hat{g}_s = -(\dot{\gamma}+\omega_g)\hat{g}_s + \omega_s\hat{g}_g = \dot{\hat{g}}_t$$

$$\frac{{}^N d}{dt}(\hat{g}_g) = \frac{{}^B d}{dt}(\hat{g}_g)^{\nearrow 0} + \vec{\omega}_{B/N} \times \hat{g}_g = (\omega_s\hat{g}_s + \omega_t\hat{g}_t + \omega_g\hat{g}_g) \times \hat{g}_g = -\omega_s\hat{g}_t + \omega_t\hat{g}_s = \dot{\hat{g}}_g$$

Next let us find the derivatives of the gimbal framed spads (constants!)

$$\text{so } \dot{\omega}_s = \frac{{}^N d}{dt}(\hat{g}_s^T\vec{\omega}_{B/N}) = \dot{\hat{g}}_s^T\vec{\omega} + \hat{g}_s^T\dot{\vec{\omega}} = \dot{\gamma}\underbrace{\hat{g}_t\vec{\omega}}_{\omega_t} + \hat{g}_s^T\dot{\vec{\omega}} = \dot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}} = \dot{\omega}_s$$

$$\dot{\omega}_t = \frac{{}^N d}{dt}(\hat{g}_t^T\vec{\omega}) = \dot{\hat{g}}_t^T\vec{\omega} + \hat{g}_t^T\dot{\vec{\omega}} = -\dot{\gamma}\hat{g}_s^T\vec{\omega} + \hat{g}_t^T\dot{\vec{\omega}} = -\dot{\gamma}\omega_s + \dot{\hat{g}}_t^T\dot{\vec{\omega}} = \dot{\omega}_t$$

$$\dot{\omega}_g = \frac{{}^N d}{dt}(\hat{g}_g^T\vec{\omega}) = \dot{\hat{g}}_g^T\vec{\omega} + \hat{g}_g^T\dot{\vec{\omega}} = \dot{\hat{g}}_g^T\dot{\vec{\omega}} = \dot{\omega}_g$$

Now we shall tackle the beast $\dot{H} = \frac{{}^N d}{dt}(H) = \frac{{}^N d}{dt}(H_B + H_G + H_W)$ INDIVIDUALLY FIRST.

$$\dot{H}_W = \frac{{}^N d}{dt}(H_W) = \frac{{}^N d}{dt}(\hat{g}_s I_{Ws}\omega_g + \hat{g}_s I_{Ws}\Omega + \hat{g}_t I_{Wt}\omega_t + \hat{g}_g I_{Wt}\omega_g + \hat{g}_g I_{Wt}\dot{\gamma})$$

$$= \dot{\hat{g}}_s I_{Ws}\omega_g + \hat{g}_s I_{Ws}\dot{\omega}_g + \dot{\hat{g}}_s I_{Ws}\Omega + \hat{g}_s I_{Ws}\dot{\Omega} + \dot{\hat{g}}_t I_{Wt}\omega_t + \hat{g}_t I_{Wt}\dot{\omega}_t + \dot{\hat{g}}_g I_{Wt}\omega_g + \hat{g}_g I_{Wt}\dot{\omega}_g + \dot{\hat{g}}_g I_{Wt}\dot{\gamma} + \hat{g}_g I_{Wt}\ddot{\gamma}$$

$$= \cancel{I_{Ws}\omega_s(\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g)} + \hat{g}_s I_{Ws}(\dot{\gamma}\omega_t + \dot{\hat{g}}_s^T\dot{\vec{\omega}}) + \cancel{\hat{g}_s I_{Ws}\Omega((\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g)} + \hat{g}_s I_{Ws}\dot{\Omega}$$
$$+ \cancel{I_{Wt}\omega_t((\dot{\gamma}+\omega_g)\hat{g}_s + \omega_s\hat{g}_g)} + \hat{g}_t I_{Wt}(-\dot{\gamma}\omega_s + \dot{\hat{g}}_t^T\dot{\vec{\omega}}) + I_{Wt}\omega_g(\quad) + \hat{g}_g I_{Wt}($$

$\Longrightarrow$

3.

$\dot{H}_W$

$= \left[(\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g\right] I_{ws}\omega_s + \hat{g}_s I_{ws}(\dot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}})$

$+ \left[(\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g\right] I_{ws}\Omega + \hat{g}_s I_{ws}\dot{\Omega} + \left[-(\dot{\gamma}+\omega_g)\hat{g}_s + \omega_s\hat{g}_g\right] I_{wt}\omega_t$

$+ \hat{g}_t I_{wt}(-\dot{\gamma}\omega_s + \hat{g}_t^T\dot{\vec{\omega}}) + \left[-\omega_s\hat{g}_t + \omega_t\hat{g}_s\right] I_{wt}\omega_g + \hat{g}_g I_{wt}(\hat{g}_g^T\dot{\vec{\omega}})$

$+ (\omega_t\hat{g}_s - \omega_s\hat{g}_t) I_{wt}\dot{\gamma} + \hat{g}_g I_{wt}\ddot{\gamma}$

(COMBINE!)

$= \hat{g}_s\left[I_{ws}(\dot{\gamma}\omega_t + \hat{g}_g^T\dot{\vec{\omega}} + \dot{\Omega}) + I_{wt}(-\omega_t(\dot{\gamma}+\omega_g) + \omega_t\dot{\gamma} + \omega_t\omega_g)\right]$

$+ \hat{g}_t\left[I_{ws}\omega_s(\dot{\gamma}+\omega_g) + I_{ws}\Omega(\dot{\gamma}+\omega_g) + I_{wt}(-\dot{\gamma}\omega_s + \hat{g}_t^T\dot{\vec{\omega}}) - I_{wt}\omega_g\omega_s - I_{wt}\dot{\gamma}\omega_s\right]$

$+ \hat{g}_g\left[-I_{ws}\omega_s\omega_t - I_{ws}\Omega\omega_t + I_{wt}\omega_t\omega_s + I_{wt}(\hat{g}_g^T\dot{\vec{\omega}}) + I_{wt}\ddot{\gamma}\right]$

(SIMPLIFY!)

$\dot{H}_W = \hat{g}_s\left[I_{ws}(\dot{\Omega} + \dot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}})\right] + \hat{g}_t\left[I_{ws}(\omega_s(\dot{\gamma}+\omega_g) + \Omega(\dot{\gamma}+\omega_g)) + I_{wt}(-2\dot{\gamma}\omega_s - \omega_s\omega_g + \hat{g}_t^T\dot{\vec{\omega}})\right]$

$+ \hat{g}_g\left[I_{ws}[-\omega_s\omega_t - \Omega\omega_t) + I_{wt}(\omega_t\omega_s + \ddot{\gamma} + \hat{g}_g^T\dot{\vec{\omega}})\right]$ ✓

Now for $\dot{H}_G$

$-\frac{md}{dt}(I_{Gs}\omega_s\hat{g}_s + I_{Gt}\omega_t\hat{g}_t + I_{Gg}\omega_g\hat{g}_g + I_{Gg}\dot{\gamma}\hat{g}_g)$

$= I_{Gs}\dot{\omega}_s\hat{g}_s + I_{Gs}\omega_s\dot{\hat{g}}_s + I_{Gt}\dot{\omega}_t\hat{g}_t + I_{Gt}\omega_t\dot{\hat{g}}_t + I_{Gg}\dot{\omega}_g\hat{g}_g + I_{Gg}\omega_g\dot{\hat{g}}_g + I_{Gg}\ddot{\gamma}\hat{g}_g + I_{Gg}\dot{\gamma}\dot{\hat{g}}_g$

$= I_{Gs}(\dot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}})\hat{g}_s + I_{Gs}\omega_s((\dot{\gamma}+\omega_g)\hat{g}_t - \omega_t\hat{g}_g) + I_{Gt}(-\dot{\gamma}\omega_s + \hat{g}_t^T\dot{\vec{\omega}})\hat{g}_t$

$+ I_{Gt}\omega_t(-(\dot{\gamma}+\omega_g)\hat{g}_s + \omega_s\hat{g}_g) + I_{Gs}(\hat{g}_g^T\dot{\vec{\omega}})\hat{g}_g + I_{Gg}\omega_g(\omega_t\hat{g}_s - \omega_s\hat{g}_t)$

$+ I_{Gg}\ddot{\gamma}\hat{g}_g + I_{Gg}\dot{\gamma}(\omega_t\hat{g}_s - \omega_s\hat{g}_t)$

(COMBINE!)

$= \hat{g}_s\left[I_{Gs}(\dot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}}) - I_{Gt}\omega_t(\dot{\gamma}+\omega_g) + I_{Gg}\omega_g\omega_t + I_{Gg}\dot{\gamma}\omega_t\right]$

$+ \hat{g}_t\left[I_{Gs}\omega_s(\dot{\gamma}+\omega_g) + I_{Gt}(-\dot{\gamma}\omega_s + \hat{g}_t^T\dot{\vec{\omega}}) - I_{Gg}\omega_g\omega_s - I_{Gg}\dot{\gamma}\omega_s\right]$

$+ \hat{g}_g\left[-I_{Gs}\omega_s\omega_t + I_{Gt}\omega_t\omega_s + I_{Gs}(\hat{g}_g^T\dot{\vec{\omega}}) + I_{Gg}\ddot{\gamma}\right]$

(SIMPLIFY!)

$\dot{H}_G = \hat{g}_s\left[\dot{\gamma}\omega_t(I_{Gs} - I_{Gt} + I_{Gg}) + I_{Gs}\hat{g}_s^T\dot{\vec{\omega}} + \omega_t\omega_s(I_{Gg} - I_{Gt})\right]$

$+ \hat{g}_t\left[\dot{\gamma}\omega_s(I_{Gs} - I_{Gt} - I_{Gg}) + I_{Gt}\hat{g}_t^T\dot{\vec{\omega}} + \omega_g\omega_s(I_{Gs} - I_{Gg})\right]$

$+ \hat{g}_g\left[I_{Gg}(\ddot{\gamma} + \hat{g}_g^T\dot{\vec{\omega}}) + \omega_s\omega_t(I_{Gt} - I_{Gs})\right]$ ✓

FINALLY, $\dot{H}_B = I_s\dot{\vec{\omega}} + \vec{\omega}\times I_s\vec{\omega}$ w/ transport.

for $J = I_{Gt} + I_W = G\begin{bmatrix} J_s & & \\ & J_t & \\ & & J_g \end{bmatrix}$; define total S/C inertia matrix $[I] = [I_s] + [J]$

$J_s = I_{Gt} + I_{ws}$; $J_t = I_{Gt} + I_{wt}$; $J_g = I_{Gg} + I_{wt}$ $\Rightarrow$ $\begin{cases} I_{Gs} = J_s - I_{ws} \\ I_{Gt} = J_t - I_{wt} \\ I_{Gg} = J_g - I_{wt} \end{cases}$

$\dot{H} = \dot{H}_{/s} + \dot{H}_G + \dot{H}_W \Rightarrow$

$\hat{g}_s$ term: $= I_{ws}(\dot{\Omega} + \ddot{\gamma}\omega_t + \hat{g}_s^T\dot{\vec{\omega}}) + \dot{\gamma}\omega_t(I_{Gs} - I_{Gt} + I_{Gg}) + I_{Gs}\hat{g}_s^T\dot{\vec{\omega}} + \omega_t\omega_g(I_{Gg} - I_{gt})$

$= \dot{\gamma}\omega_t(\underbrace{I_{ws} + I_{Gs}}_{J_s} - I_{Gt} + I_{Gg}) + \hat{g}_s^T\dot{\vec{\omega}}\underbrace{(I_{ws} + I_{Gs})}_{J_s} + I_{ws}\dot{\Omega} + \omega_t\omega_g(I_{Gg} - I_{gt})$

$= J_s\ddot{\gamma}\omega_t + I_{ws}\dot{\Omega} - \dot{\gamma}\omega_t(\underbrace{I_{Gt} - I_{Gg}}) + \omega_t\omega_g(I_{Gg} - I_{Gt}) + \hat{g}_s^T\dot{\vec{\omega}}J_s$

$= J_t - \cancel{I_{wt}} - J_g + \cancel{I_{Gt}}$

$= J_t - J_g$

$\therefore = \boxed{J_s\ddot{\gamma}\omega_t + I_{ws}\dot{\Omega} - \dot{\gamma}\omega_t(J_t - J_g) + \omega_t\omega_g(\underbrace{I_{Gg} - I_{Gt}}_{J_g - J_t}) + \hat{g}_s^T\dot{\vec{\omega}}J_s}$

$\hat{g}_t$ term $= I_{ws}\left(\omega_s(\dot{\gamma} + \dot{\omega}_g) + \Omega(\dot{\gamma} + \omega_g)\right) + I_{wt}\left(-2\dot{\gamma}\omega_s - \omega_s\omega_g + \hat{g}_t^T\dot{\vec{\omega}}\right) + \dot{\gamma}\omega_s[I_{Gs} - I_{Gt} + I_{Gg}]$

$+ I_{Gt}\hat{g}_t^T\dot{\vec{\omega}} + \omega_g\omega_s(I_{Gs} - I_{Gg})$

$= \dot{\gamma}\omega_s\left(I_{ws} + I_{Gs} - I_{Gt} - \overset{-2I_{wt}}{\overset{\downarrow}{I_{Gg}}}\right) + \omega_g\omega_s\left(I_{Gs} - I_{Gg} - \overset{+I_{ws}}{\overset{\downarrow}{I_{wt}}}\right) + \hat{g}_t^T\dot{\vec{\omega}}(I_{wt} + I_{Gt})$

$+ I_{ws}\Omega(\dot{\gamma} + \omega_g)$

$= \dot{\gamma}\omega_s(J_s - J_t - J_g) + \omega_g\omega_s(J_s - J_g) + J_t\hat{g}_t^T\dot{\vec{\omega}} + I_{ws}\Omega(\dot{\gamma} + \omega_g)$

$= \boxed{\dot{\gamma}(J_s\omega_s + I_{ws}\Omega) - (J_t + J_g)\omega_s\dot{\gamma} + I_{ws}\Omega\omega_g + (J_s - J_g)\omega_g\omega_s + J_t\hat{g}_t^T\dot{\vec{\omega}}}$

$\hat{g}_g$ term $= I_{ws}(-\omega_s\omega_t - \Omega\omega_t) + I_{wt}(\omega_t\dot{\omega}_s + \ddot{\gamma} + \hat{g}_s^T\dot{\vec{\omega}}) + I_{Gg}(\ddot{\gamma} + \hat{g}_g^T\dot{\vec{\omega}}) + \omega_s\dot{\omega}_t(I_{Gt} - I_{Gs})$

$= \omega_s\omega_t(-I_{ws} + I_{wt} + I_{Gt} - I_{Gs}) + \hat{g}_s^T\dot{\vec{\omega}}(I_{wt} + I_{Gg}) + \ddot{\gamma}(I_{Gg} + I_{wt}) - I_{ws}\Omega\omega_t$

$= \omega_s\omega_t(J_t - J_s) + J_g\hat{g}_g^T\dot{\vec{\omega}} + J_g\ddot{\gamma} - I_{ws}\Omega\omega_t$

$= \boxed{J_g\ddot{\gamma} - I_{ws}\Omega\omega_t + (J_t - J_s)\omega_s\omega_t + J_g\hat{g}_g^T\dot{\vec{\omega}}}$

& $\dot{H} = L$ Let's write as in form using $[I]$ for EOM of whole S/C.

$\Rightarrow$

5.

$\dot{H}_T =$

$$[I_s]\dot{\vec{\omega}} + \vec{\omega}\times[I_s]\vec{\omega} + \hat{g}_s\left[J_s\ddot{\gamma}\omega_t + I_{ws}\dot{\Omega} + \dot{\gamma}\omega_t(J_t - J_g) + \omega_t\omega_g(J_g - J_t) + J_s\hat{g}_s^T\dot{\vec{\omega}}\right]$$
$$+ \hat{g}_t\left[\ddot{\gamma}(J_s\omega_s + I_{ws}\Omega) - (J_t+J_g)\omega_s\dot{\gamma} + I_{ws}\Omega\omega_g + (J_s - J_g)\omega_g\omega_s + J_t\hat{g}_t^T\dot{\vec{\omega}}\right]$$
$$+ \hat{g}_g\left[J_g\ddot{\gamma} - I_{ws}\Omega\omega_t + (J_t - J_s)\omega_s\omega_t + J_g\hat{g}_g^T\dot{\vec{\omega}}\right]$$
$$= \vec{L}$$

want $I\dot{\vec{\omega}} = [I_s + J]\dot{\vec{\omega}} = \underbrace{I_s\dot{\vec{\omega}}}_{\text{have.}} + \underbrace{J\dot{\vec{\omega}}}_{\text{Need.}} = I_s\dot{\vec{\omega}} + J_s\hat{g}_s^T\dot{\vec{\omega}} + J_t\hat{g}_t^T\dot{\vec{\omega}} + J_g\hat{g}_g^T\dot{\vec{\omega}}$ → decompose:

take from $\vec{g}$ components above.

$\therefore$

$$L = [I]\dot{\vec{\omega}} + \underbrace{\vec{\omega}\times[I_s]\vec{\omega}} + \dots$$

Lets make $\vec{\omega}\times[I]\vec{\omega} = \vec{\omega}\times[I_s + J]\vec{\omega} = \vec{\omega}\times I_s\vec{\omega} + \underbrace{\vec{\omega}\times J\vec{\omega}}$

$$\vec{\omega}\times J\vec{\omega} = \vec{\omega}^{\times}J\vec{\omega} = \begin{bmatrix}0 & -\omega_g & \omega_t\\ \omega_g & 0 & -\omega_s\\ -\omega_t & \omega_s & 0\end{bmatrix}\begin{bmatrix}J_s & \cdot & \cdot\\ \cdot & J_t & \cdot\\ \cdot & \cdot & J_g\end{bmatrix}\begin{bmatrix}\omega_s\\ \omega_t\\ \omega_g\end{bmatrix} = \vec{\omega}^{\times}\begin{bmatrix}J_s\omega_s\\ J_g\omega_t\\ J_g\omega_g\end{bmatrix} = \begin{matrix}J_g\omega_g\omega_t - J_t\omega_t\omega_g +\\ J_s\omega_s\omega_g - J_g\omega_g\omega_s +\\ J_t\omega_t\omega_s - J_s\omega_s\omega_t\end{matrix}$$

$$= (J_g - J_t)(\omega_t\omega_g) + (J_s - J_g)\omega_s\omega_g + (J_t - J_s)\omega_t\omega_s$$

Notice <u>again</u> these exist in components above! Steal them...

$\therefore$ Now we have

$$L = [I]\dot{\vec{\omega}} + \vec{\omega}\times[I]\vec{\omega} + \hat{g}_s\left[J_s\ddot{\gamma}\omega_t + I_{ws}\dot{\Omega} - \dot{\gamma}\omega_t(J_t - J_g)\right]$$
$$+ \hat{g}_t\left[\ddot{\gamma}(J_s\omega_s + I_{ws}\Omega) - \dot{\gamma}\omega_s(J_t + J_g) + I_{ws}\Omega\omega_g\right]$$
$$+ \hat{g}_g\left[J_g\ddot{\gamma} - I_{ws}\Omega\omega_t\right]$$

& Solving for $[I]\dot{\vec{\omega}} =$

$$[I]\dot{\vec{\omega}} = -\vec{\omega}\times[I]\vec{\omega} - \hat{g}_s\left[J_s\ddot{\gamma}\omega_t + I_{ws}\dot{\Omega} - \dot{\gamma}\omega_t(J_t - J_g)\right]$$
$$- \hat{g}_t\left[\ddot{\gamma}(J_s\omega_s + I_{ws}\Omega) - \dot{\gamma}\omega_s(J_t + J_g) + I_{ws}\Omega\omega_g\right]$$
$$- \hat{g}_g\left[J_g\ddot{\gamma} - I_{ws}\Omega\omega_t\right] + \vec{L}$$

E.O.M. SINGLE CMG.

LET US NOW EXTEND THIS SOLUTION TO N #VSCMG's

WE WILL THEN HAVE (decomposed)

$$[I] = [I_s] + \sum_{i=1}^{N} [J_i] = [I_s] + \sum_{i=1}^{N} J_{si}\,\hat{g}_{si}\hat{g}_{si}^T + J_{ti}\,\hat{g}_{ti}\,\hat{g}_{ti}^T + J_{gi}\,\hat{g}_{gi}\,\hat{g}_{gi}^T$$

∴ extend solution

$$\left\{ \begin{aligned} [I]\dot{\vec{\omega}} = -\vec{\omega}\times[I]\vec{\omega} &- \sum_{i=1}^{N} \hat{g}_{si}\left[J_{si}\dot{\gamma}_i\omega_{ti} + I_{ws_i}\dot{\Omega}_i - \dot{\gamma}_i\omega_{ti}(I_{ti}-J_{gi})\right] \\ &- \sum_{i=1}^{N} \hat{g}_{ti}\left[\dot{\gamma}_i(J_{si}\omega_{si}+I_{ws_i}\Omega_i) - \dot{\gamma}_i\omega_{si}(J_{ti}-J_{gi}) + I_{ws_i}\Omega_i\omega_{gi}\right] \\ &- \sum_{i=1}^{N} \hat{g}_{gi}\left[J_{gi}\ddot{\gamma}_i - I_{ws_i}\Omega_i\omega_{ti}\right] + L \end{aligned} \right\}$$

GIVEN that this is a SUMMATION, WE CAN separate it INTO A MATRIX MULTIPLICATION

Define $[G_s] = [\hat{g}_{s1}\ \hat{g}_{s2}\ \cdots\ \hat{g}_{sN}] \in \mathbb{R}^{3\times N}$ & it's respective $\tau_s = \begin{bmatrix} J_{s1}\dot{\gamma}_1\omega_{t1} + I_{ws_1}\dot{\Omega}_1 - \dot{\gamma}_1\omega_{t1}(I_{t1}-J_{g1}) \\ \vdots \\ J_{sN}\dot{\gamma}_N\omega_{tN} + I_{ws_N}\dot{\Omega}_N - \dot{\gamma}_N\omega_{tN}(J_{tN}-J_{gN}) \end{bmatrix} \in \mathbb{R}^{N\times 1}$

WE CAN REPRESENT THE LAST TWO SUMMATIONS SIMILARLY!

$$[G_t] = [\hat{g}_{t1}\ \hat{g}_{t2}\ \cdots\ \hat{g}_{tN}] \in \mathbb{R}^{3\times N} \quad \text{for} \quad \tau_t = \begin{bmatrix} \dot{\gamma}_1(J_{s1}\omega_{s1} + I_{ws_1}\Omega_1) - \dot{\gamma}_1\omega_{s1}(J_{t1}-J_{g1}) + I_{ws_1}\Omega_1\omega_{g1} \\ \vdots \\ \dot{\gamma}_N(J_{sN}\omega_{sN} + I_{ws_N}\Omega_N) - \dot{\gamma}_N\omega_{sN}(J_{tN}-J_{gN}) + I_{ws_N}\Omega_N\omega_{gN} \end{bmatrix} \in \mathbb{R}^{N\times 1}$$

and

$$[G_g] = [\hat{g}_{g1}\ \hat{g}_{g2}\ \cdots\ \hat{g}_{gN}] \in \mathbb{R}^{3\times N} \quad \text{for} \quad \tau_g = \begin{bmatrix} J_{g1}\ddot{\gamma}_1 - I_{ws_1}\Omega_1\omega_{t1} \\ \vdots \\ J_{gN}\ddot{\gamma}_N - I_{ws_N}\Omega_N\omega_{tN} \end{bmatrix} \in \mathbb{R}^{N\times 1}$$

& have the FINAL FORM

$$\boxed{[I]\dot{\vec{\omega}} = -\vec{\omega}\times[I]\vec{\omega} - [G_s]\tau_s - [G_t]\tau_t - [G_g]\tau_g + \vec{L}} \quad \in \mathbb{R}^{3\times 1}$$

7 1b.

Differentiate T fxn to determine power:

$$T = \underbrace{\tfrac{1}{2}\vec{\omega}^T[I_s]\vec{\omega}}_{1} + \tfrac{1}{2}\sum_{i=1}^{N} I_{w_{si}}(\Omega_i + \omega_{si})^2 + I_{G_{si}}\omega_{si}^2 + J_{ei}\omega_{ei}^2 + J_{gi}(\omega_{gi} + \dot{\gamma}_i)^2$$

FIRST TERM:

$$\frac{d}{dt}\left(\tfrac{1}{2}\vec{\omega}^T I_s \vec{\omega}\right) = \vec{\omega}^T[I_s]\dot{\vec{\omega}} = \longrightarrow \text{EOM!}$$

$\leftarrow$ for $u_{si} = I_{w_{si}}\left(\dot{\Omega}_i + \hat{g}_{s_i}^T\dot{\vec{\omega}}_i + \ddot{\gamma}_i\omega_{t_i}\right)$ spin torque. ; $u_{gi} = J_{g_i}(\hat{g}_{s_i}^T\dot{\vec{\omega}}_i + \ddot{\gamma}_i) - (J_{s_i} - J_{t_i})\omega_{s_i}\omega_{t_i} - I_{w_{s_i}}\Omega_{t_i}\omega_{t_i}$ gimbal torque

$$\vec{\omega}^T\left[\vec{L} - \vec{\omega}\times[I_s]\vec{\omega} = \sum_i^n \hat{g}_{s_i}\left[u_{s_i} + \dot{\gamma}_i\omega_{t_i}(I_{G_{s_i}} - J_{t_i} + J_{g_i}) + I_{G_{s_i}}\hat{g}_{s_i}^T\dot{\vec{\omega}}_i + \omega_{t_i}\omega_{g_i}(J_{g_i} - J_{t_i})\right]\right.$$

$$- + \sum_i^n \hat{g}_{t_i}\left[\ddot{\gamma}_i(J_{s_i}\omega_{s_i} + I_{w_{s_i}}\Omega_i) - (J_{t_i} + J_{g_i})\omega_{t_i}\dot{\gamma}_i + \cancel{I_{w_{s_i}}\dot{\Omega}_i\omega_{t_i}} + I_{w_{s_i}}\Omega_i\omega_{g_i} + (J_{s_i} - J_{g_i})\omega_{g_i}\omega_{s_i} + J_{t_i}\hat{g}_{t_i}^T\dot{\vec{\omega}}\right]$$

$$\left. - + \sum_i^n \hat{g}_{g_i}\left[u_{g_i}\right]\right]$$

$$= \omega^T\vec{L} - \sum_i^n \omega_{s_i}\left[\cancel{u_{s_i}} + \dot{\gamma}_i\omega_{t_i}(\cancel{I_{G_{s_i}}} - \cancel{J_{t_i}} + \cancel{J_{g_i}}) + I_{G_{s_i}}\hat{g}_{s_i}^T\dot{\vec{\omega}} + \omega_{t_i}\omega_{g_i}(\cancel{J_{g_i}} - J_{t_i})\right]$$

(below; below; third term)

$$- \sum_i^n \omega_{t_i}\left[\ddot{\gamma}(J_{s_i}\omega_{s_i} + I_{w_{s_i}}\Omega_i) - (\cancel{J_{t_i}} + \cancel{J_{g_i}})\omega_{t_i}\dot{\gamma}_i + I_{w_{s_i}}\dot{\Omega}_i\omega_{t_i} + (\cancel{J_{s_i}} - \cancel{J_{g_i}})\omega_{g_i}\omega_{s_i} + J_{t_i}\hat{g}_{t_i}^T\dot{\vec{\omega}}_i\right]$$

(below)

$$- \sum_i^n \omega_{g_i}\left[\cancel{J_{g_i}(\hat{g}_{s_i}^T\dot{\vec{\omega}} + \ddot{\gamma})} - (\cancel{J_{s_i}} - \cancel{J_{t_i}})\omega_{s_i}\omega_{t_i} - I_{w_{s_i}}\dot{\Omega}_i\omega_{t_i}\right]_i$$

(cancels w/ below; cancels w/ above; first term)

Littler to expand!

second term

$$\sum_{i=1}^{N}\frac{1}{2}\left(\frac{d}{dt}\left(I_{w_s}(\Omega_i + \omega_{s_i})^2 + I_{G_{s_i}}\omega_{s_i}^2 + J_{e_i}\omega_{e_i}^2 + J_{g_i}(\omega_{s_i}^0 + \dot{\gamma}_i)^2\right)\right)$$

$$= \cancel{\tfrac{1}{2}}\sum_{i=1}^{N} I_{w_s}\left(\cancel{2}\Omega_i\dot{\Omega}_i + \cancel{2}\dot{\Omega}_i\omega_{s_i} + \cancel{2}\dot{\Omega}_i\omega_{s_i} + 2\omega_{s_i}\dot{\omega}_{s_i}\right) + \cancel{2}I_{G_s}\omega_s\dot{\omega}_s + \cancel{2}J_{e_i}\omega_{e_i}\dot{\omega}_{e_i} + J_{g_i}\left(\omega_{s_i}\dot{\omega}_i + 2\omega_{s_i}\dot{\gamma} + 2\dot{\gamma}\omega_s + 2\dot{\gamma}\dot{\gamma}\right)_i$$

$$= \sum_{i=1}^{N} I_{w_s}\underbrace{\left((\Omega+\omega_s)(\dot{\Omega}+\dot{\omega}_s)\right)_i}_{(\dot{\Omega}+\dot{\gamma}\omega_t+\hat{g}_s^T\dot{\vec{\omega}})_i} + \underbrace{I_{G_{s_i}}\omega_{s_i}\dot{\omega}_{s_i}}_{I_{G_{s_i}}\omega_{s_i}(\dot{\gamma}\omega_{t_i}+\hat{g}_{s_i}^T\dot{\vec{\omega}})_i} + \underbrace{J_{e_i}\omega_{e_i}\dot{\omega}_{e_i}}_{J_{t_i}\omega_{e_i}(\hat{g}_{t_i}^T\dot{\vec{\omega}}-\dot{\gamma}\omega_{s_i})_i} + \underbrace{J_{g_i}\left((\dot{\gamma}+\omega_g)(\dot{\omega}_g+\ddot{\gamma})\right)_i}_{J_{g_i}(\dot{\gamma}+\omega_g)(\hat{g}_g^T\dot{\vec{\omega}}+\ddot{\gamma})_i}$$

$$= \sum_{i=1}^{n} u_{s_i}(\Omega+\omega_s) + I_{G_{s_i}}\hat{g}_{s_i}^T\dot{\vec{\omega}} + J_{t_i}\omega_{e_i}\hat{g}_{t_i}^T\dot{\vec{\omega}} + \dot{\gamma}\left(I_{G_{s_i}}\omega_{s_i}\omega_{e_i} - J_{t_i}\omega_{e_i}\omega_{s_i}\right) + J_{g_i}(\hat{g}_g^T\dot{\vec{\omega}}+\ddot{\gamma})(\dot{\gamma}+\omega_g)$$

(ω_{s_i})

$$= \sum u_{s_i}(\Omega_i + \cancel{\omega_{s_i}}) + \underbrace{I_{G_{s_i}}\omega_{s_i}\hat{g}_s^T\dot{\vec{\omega}}}_{\text{cancels above}} + \underbrace{J_{t_i}\omega_{e_i}\hat{g}_{t_i}^T\dot{\vec{\omega}}}_{\text{cancels w/ above}} + \omega_{s_i}\dot{\gamma}\cancel{(I_{G_{s_i}} - J_{t_i})} + J_{g_i}\left(\cancel{\omega_g\hat{g}_g^T\dot{\vec{\omega}}} + \cancel{\omega_g\ddot{\gamma}} + \dot{\gamma}\hat{g}_g^T\dot{\vec{\omega}} + \dot{\gamma}\ddot{\gamma}\right)_i$$

(ω_{s_i} cancels w/ above; cancels w/ above; cancels w/ above; cancels w/ above)

SUMMED & CANCELED:

$$\dot{T} = \omega^T\vec{L} + \sum_{i=1}^{n} u_{s_i}\Omega_i + \dot{\gamma}_i J_{g_i}(\hat{g}_g^T\dot{\vec{\omega}} + \ddot{\gamma})_i - \dot{\gamma}\omega_{t_i}\left(J_{s_i}\omega_{s_i} + I_{w_{s_i}}\Omega_i - J_{t_i}\omega_{e_i}\right)$$

$$+ \sum_{i=1}^{n} u_{s_i}\Omega_i + \dot{\gamma}\left(J_{g_i}(\hat{g}_g^T\dot{\vec{\omega}} + \ddot{\gamma})_i - (J_s - J_t)_i\omega_{s_i}\omega_{t_i} - I_{w_{s_i}}\Omega_i\omega_{t_i}\right)$$

→ notice u_{g_i} from above

$$\therefore \boxed{\dot{T} = \omega^T\vec{L} + \sum_{i=1}^{n} u_{s_i}\Omega_i + \dot{\gamma}_i u_{g_i}} \quad \checkmark$$

8. 1c.

OUR RESULT FROM Page 5 shows that w/ $\dot{\gamma}=\ddot{\gamma}=0$: (1 RW)

$$[I]\dot{\vec{\omega}} = -\vec{\omega}\times[I]\vec{\omega} - \hat{g}_s(I_{ws}\dot{\Omega}) - \hat{g}_t(I_{ws}\Omega\omega_g) - \hat{g}_g(-I_{ws}\Omega\omega_t) + \vec{L}$$

$$= -\vec{\omega}\times[I]\vec{\omega} - \hat{g}_s I_{ws}\dot{\Omega} - I_{ws}\Omega(\omega_g\hat{g}_t - \omega_t\hat{g}_g) + \vec{L}$$

Notice $= -\vec{\omega}\times I_{ws}\hat{g}_s\Omega$

& our SPIN control torque is defined as $u_{si} = I_{ws}(\dot{\Omega} + \hat{g}_s^T\dot{\vec{\omega}})$

Let's get into this form.

& know $[I_{RW}] = [I_s] + I_{wt}\hat{g}_t\hat{g}_t^T + I_{wg}\hat{g}_s\hat{g}_s^T$

$$\therefore [I_{RW}]\dot{\vec{\omega}} = -\omega\times[I]\vec{\omega} - \hat{g}_s[I_{wt}\dot{\Omega}] - I_{ws}\hat{g}_s\hat{g}_s^T\dot{\vec{\omega}} - \vec{\omega}\times I_{ws}\Omega\hat{g}_s + L$$

↓ pull out stuff; pulled from LHS

$$[I_{RW}]\dot{\vec{\omega}} = -\vec{\omega}\times[I_{RC}]\vec{\omega} - \hat{g}_s\underbrace{[I_{wt}(\dot{\Omega} + \hat{g}_s^T\dot{\vec{\omega}})]}_{u_{si}} - \vec{\omega}\times I_{ws}\Omega\hat{g}_s - \vec{\omega}\times I_{ws}\omega_s + L$$

$\therefore$ EOM

$$[I_{RW}]\dot{\vec{\omega}} = -\vec{\omega}\times[I_{RC}]\vec{\omega} - \hat{g}_s u_s - \vec{\omega}\times I_{ws}\hat{g}_s(\omega_s + \Omega) + L$$

EXTEND TO N RWs!

$$\boxed{[I_{RW}]\dot{\vec{\omega}} = -\vec{\omega}\times[I_{RW}]\vec{\omega} - \sum_{i=1}^{N}\hat{g}_{si}u_{si} - \vec{\omega}\times\sum_{i=1}^{N}I_{ws_i}\hat{g}_{si}(\omega_{s_i} + \Omega_i) + L}$$

We can AGAIN break this into matrix multiplications

$$[G_s] = [\hat{g}_{s_1}\ \cdots\ \hat{g}_{sN}],\quad [\vec{h}_s] = \begin{bmatrix} I_{ws_1}(\omega_{s_1}+\Omega_1) \\ \vdots \\ I_{wsN}(\omega_{sN}+\Omega_N) \end{bmatrix},\quad u_s = \begin{bmatrix} u_{s_1} \\ u_{sN} \end{bmatrix}$$

$$\therefore \boxed{[I_{RB}]\dot{\vec{\omega}} = -\vec{\omega}\times[I_{RW}]\vec{\omega} - [G_s]\vec{u}_s - \omega\times[G_s]\vec{h}_s + \vec{L}}$$

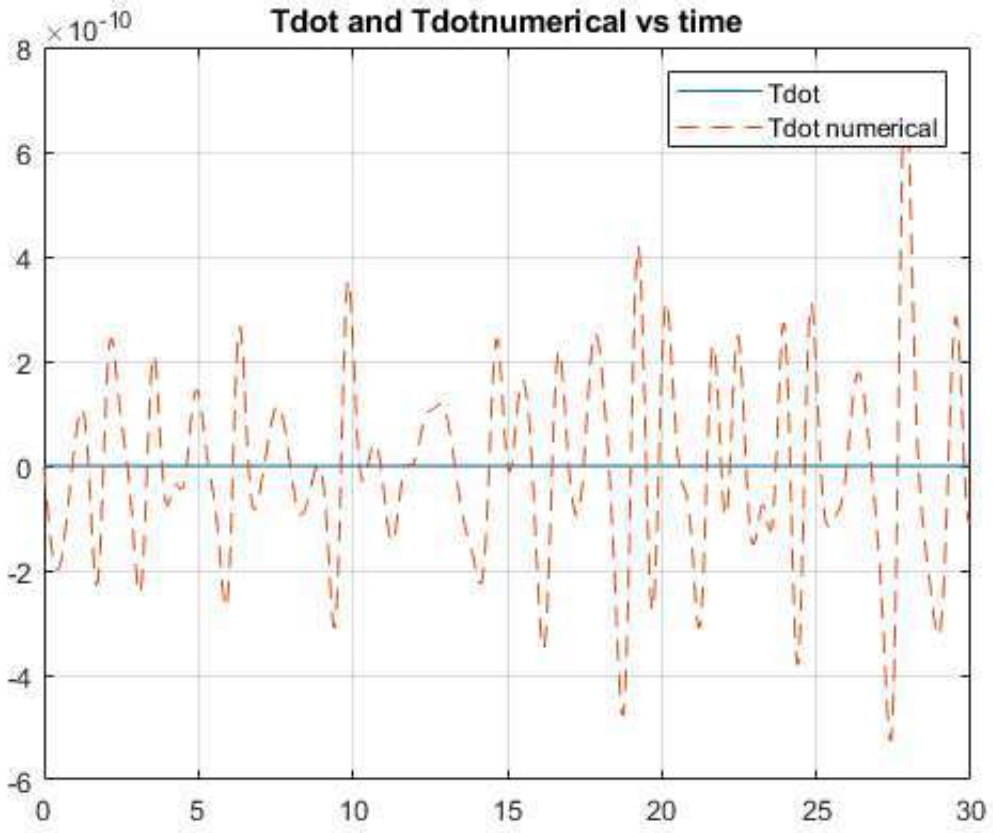
Question #2: Trying each of the cases to test my simulation

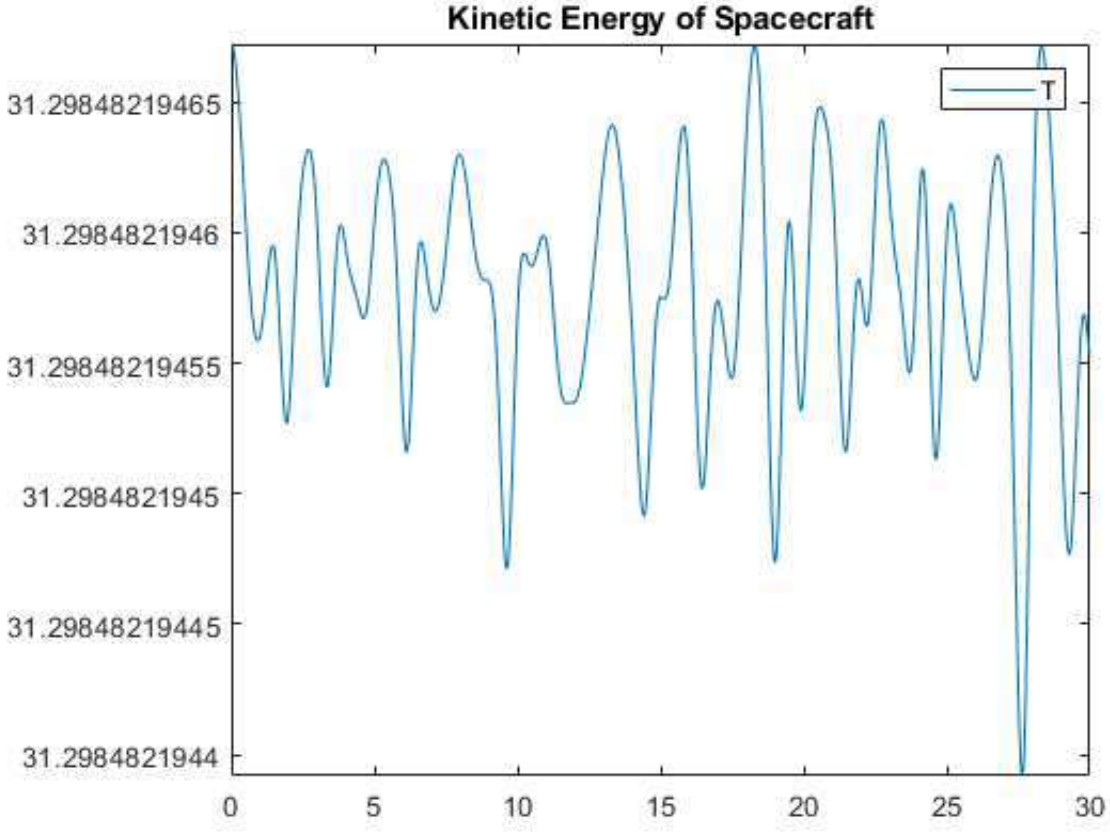
```
%%ASEN6010 Homework 2: Padraig Lysandrou
% In this situation, L,us,ug are all zero
clc; close all; clear all;

% Set up devices, SC inertia, timing
Is = diag([86.215 85.070 113.565]);
VSCMG_setup;
dt = 0.01;
t = 0:dt:30;
npoints = length(t);
acs_mode = 0; % in VSCMG mode

sys_state = zeros(6+3*num_devices, npoints);
sigma_0 = [0.414 0.3 0.2].';
omega_0 = [0.03 0.05 -0.01].';
gam0 = [VSCMG(1:num_devices).gam0].';
dgam0 = [VSCMG(1:num_devices).dgam0].';
rtr_spd0 = [VSCMG(1:num_devices).Omega0].';
sys_state(:,1) = [sigma_0.' omega_0.' gam0.' dgam0.' rtr_spd0.'].';

Gs0 = [VSCMG(1:num_devices).gs0];
Gt0 = [VSCMG(1:num_devices).gt0];
Gg = [VSCMG(1:num_devices).gg];
Js = [VSCMG(1:num_devices).Js].'; % !! use element-wise multiplication
Jt = [VSCMG(1:num_devices).Jt].';
Jg = [VSCMG(1:num_devices).Jg].';
Iws= [VSCMG(1:num_devices).Iws].';
Iwt= [VSCMG(1:num_devices).Iwt].';
Igs= [VSCMG(1:num_devices).Igs].';
Igt= [VSCMG(1:num_devices).Igt].';
Igg= [VSCMG(1:num_devices).Igg].';

% just to set things up..
ug = zeros(num_devices,npoints);
us = zeros(num_devices,npoints);

% what services will the dynamics function need?
p.VSCMG = VSCMG; p.nd = num_devices;
p.Gs0 = Gs0; p.Gt0 = Gt0; p.Gg = Gg; p.Js = Js; p.Jt = Jt; p.Jg = Jg;
p.gam0 = gam0; p.Iws = Iws; p.Is = Is;
f_dot = @(t_in,state_in,param) sc_dyn(t_in,state_in,param);

% some remaining setup for the loop
Gs = Gs0; Gt = Gt0;
L = zeros(3,1); p.L = L;
Tdot = zeros(1,npoints-1);
Tdot_numer = zeros(1,npoints-2);
T = zeros(1,npoints-1);
H = zeros(3,npoints-1);
ws = zeros(p.nd,1);
wt = zeros(p.nd,1);
wg = zeros(p.nd,1);

tic
for i = 1:npoints-1
```

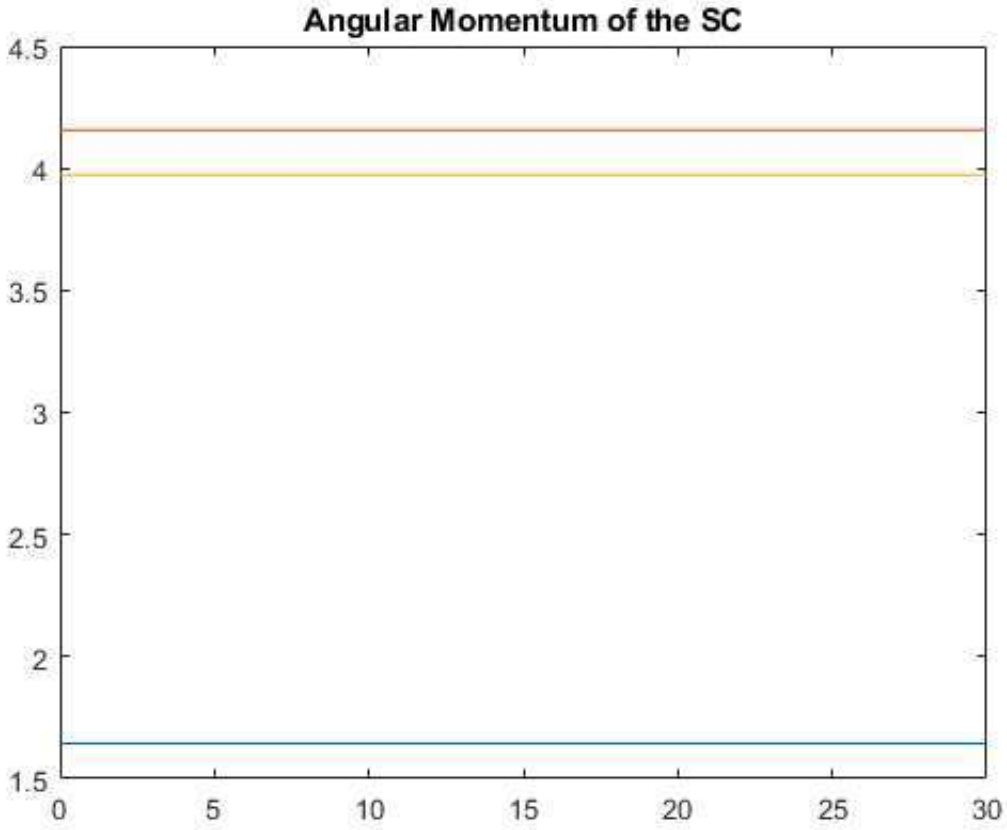
```
    sigma = sys_state(1:3,i);
    omega = sys_state(4:6,i);
    gam = sys_state(7:6+p.nd,i);
    dgam= sys_state(7+p.nd:9+p.nd,i);
    rtr_spd = sys_state(10+p.nd:end,i);

    Gs = (cos(gam - gam0).' .* Gs0) + (sin(gam - gam0).' .*Gt0);
    Gt = (-sin(gam - gam0).' .* Gs0) + (cos(gam - gam0).' .*Gt0);
    for j = 1:p.nd
        ws(j) = Gs(:,j).' * omega;
        wt(j) = Gt(:,j).' * omega;
        wg(j) = Gg(:,j).' * omega;
    end

    % here we need to update the ug and us apriori, and constant through
    L = zeros(3,1);
    p.L = L;
    p.us = zeros(3,1);
    p.ug = zeros(3,1) ;

    % look at the energy, energy rate, and the angular momentum
    Tdot(i) = omega.'*L + sum(dgam.*p.ug) + sum(rtr_spd.*p.us);
    T(i)= (0.5*omega.'*Is*omega) + 0.5*sum(Iws.*((rtr_spd + ws).^2) + ...
        (Igs.*(ws.^2)) + (Jt.*(wt.^2)) + (Jg.*((wg + dgam).^2)));
    if i>1
        Tdot_numer(i-1) = (T(i)-T(i-1))/dt;
    end
    Hg = sum((Igs.*ws).' .*Gs,2) + sum((Igt.*wt).' .*Gt,2) + ...
        sum((Igg.*(wg+dgam)).' .*Gg,2);
    Hw = sum((Iws.*(ws+rtr_spd)).' .*Gs,2) + sum((Iwt.*wt).' .*Gt,2) + ...
        sum((Iwt.*(wg+dgam)).' .*Gg,2);
    Hb = Is*omega;
    NtoB = eye(3) + ((8.*(skew(sigma)*skew(sigma)) - ...
        (4.*(1-norm(sigma)^2)*skew(sigma)))/((1+norm(sigma)^2)^2));
    H(:,i) = NtoB.' *(Hb + Hg + Hw);

    % the first thing is the time, and then all other are derivatives
    k_1 = f_dot(t(i),sys_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, sys_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(sys_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(sys_state(:,i)+k_3*dt), p);
    sys_state(:,i+1) = sys_state(:,i) + (1/6)*(k_1+2*k_2+2*k_3+k_4)*dt;

    % nonsingular attitude check
    s = norm(sys_state(1:3,i+1));
    if s > 1
        sys_state(1:3,i+1) = -(sys_state(1:3,i+1) ./(s^2));
    end
end
toc

% for L, ug, us = 0, lets plot T, Tdot, and Hi
t_span = t(1:end-1);
figure;
plot(t_span,Tdot); hold on;
plot(t_span(1:end-1),Tdot_numer,'--'); hold off;
title('Tdot and Tdotnumerical vs time');
```

```
legend('Tdot','Tdot numerical');
grid on;

% plot the energy by itself
figure;
plot(t_span,T);
legend('T');
title('Kinetic Energy of Spacecraft');

% show that the inertial angmom is constant!
figure;
plot(t_span,H);
title('Angular Momentum of the SC');
```

```
Elapsed time is 1.052874 seconds.
```




Kinetic Energy of Spacecraft
T
31.29848219465
31.2984821946
31.29848219455
31.2984821945
31.29848219445
31.2984821944
0
5
10
15
20
25
30


Angular Momentum of the SC
4.5
4
3.5
3
2.5
2
1.5
0
5
10
15
20
25
30

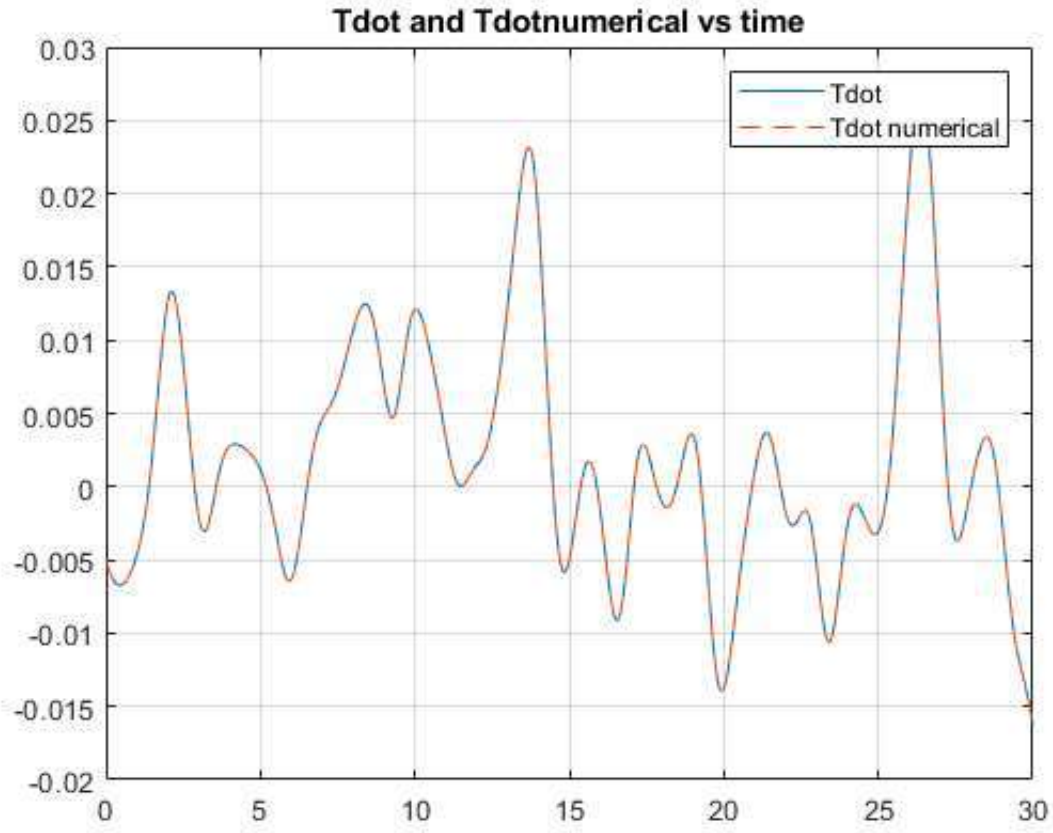
Published with MATLAB® R2018a

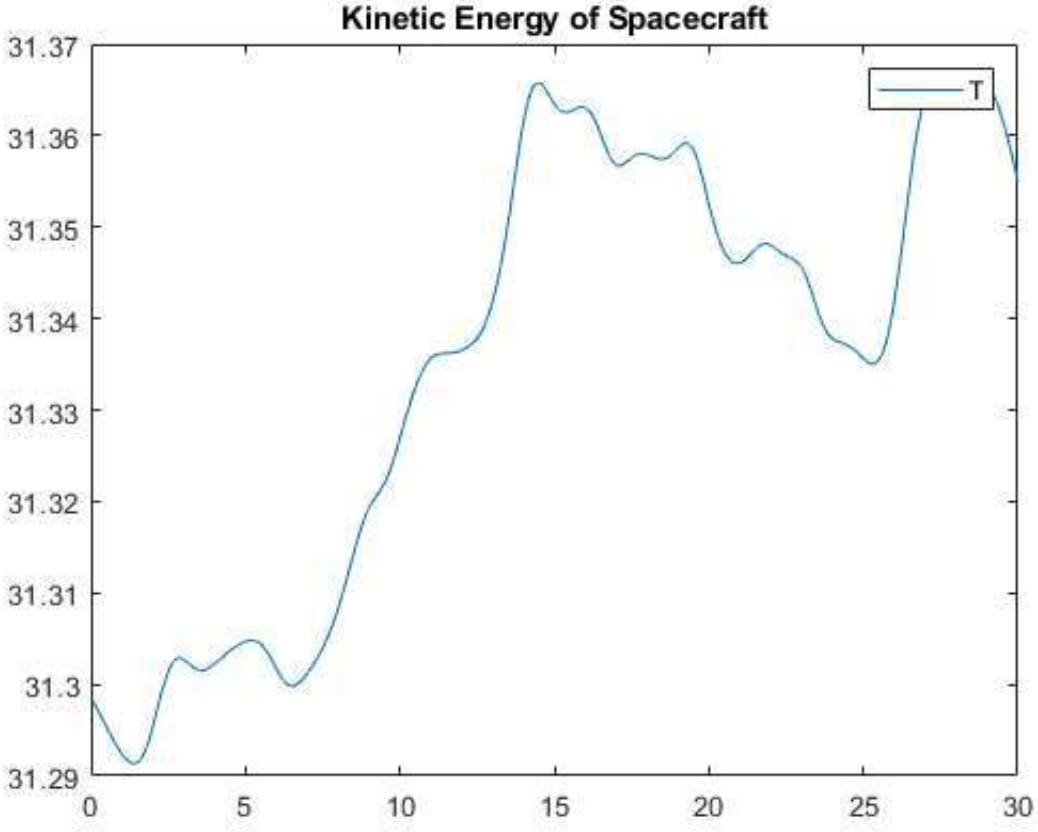
```
%%ASEN6010 Homework 2: Padraig Lysandrou
% In this situation, L is time varying, and the inputs are zero
clc; close all; clear all;

% Set up devices, SC inertia, timing
Is = diag([86.215 85.070 113.565]);
VSCMG_setup;
dt = 0.001;
t = 0:dt:30;
npoints = length(t);
acs_mode = 0; % in VSCMG mode

sys_state = zeros(6+3*num_devices, npoints);
sigma_0 = [0.414 0.3 0.2].';
omega_0 = [0.03 0.05 -0.01].';
gam0 = [VSCMG(1:num_devices).gam0].';
dgam0 = [VSCMG(1:num_devices).dgam0].';
rtr_spd0 = [VSCMG(1:num_devices).Omega0].';
sys_state(:,1) = [sigma_0.' omega_0.' gam0.' dgam0.' rtr_spd0.'].';

Gs0 = [VSCMG(1:num_devices).gs0];
Gt0 = [VSCMG(1:num_devices).gt0];
Gg = [VSCMG(1:num_devices).gg];
Js = [VSCMG(1:num_devices).Js].'; % !! use element-wise multiplication
Jt = [VSCMG(1:num_devices).Jt].';
Jg = [VSCMG(1:num_devices).Jg].';
Iws= [VSCMG(1:num_devices).Iws].';
Iwt= [VSCMG(1:num_devices).Iwt].';
Igs= [VSCMG(1:num_devices).Igs].';
Igt= [VSCMG(1:num_devices).Igt].';
Igg= [VSCMG(1:num_devices).Igg].';

% just to set things up..
ug = zeros(num_devices,npoints);
us = zeros(num_devices,npoints);

% what services will the dynamics function need?
p.VSCMG = VSCMG; p.nd = num_devices;
p.Gs0 = Gs0; p.Gt0 = Gt0; p.Gg = Gg; p.Js = Js; p.Jt = Jt; p.Jg = Jg;
p.gam0 = gam0; p.Iws = Iws; p.Is = Is;
f_dot = @(t_in,state_in,param) sc_dyn(t_in,state_in,param);

% some remaining setup for the loop
Gs = Gs0; Gt = Gt0;
L = zeros(3,1); p.L = L;
Tdot = zeros(1,npoints-1);
Tdot_numer = zeros(1,npoints-2);
T = zeros(1,npoints-1);
H = zeros(3,npoints-1);
ws = zeros(p.nd,1);
wt = zeros(p.nd,1);
wg = zeros(p.nd,1);

tic
for i = 1:npoints-1
```

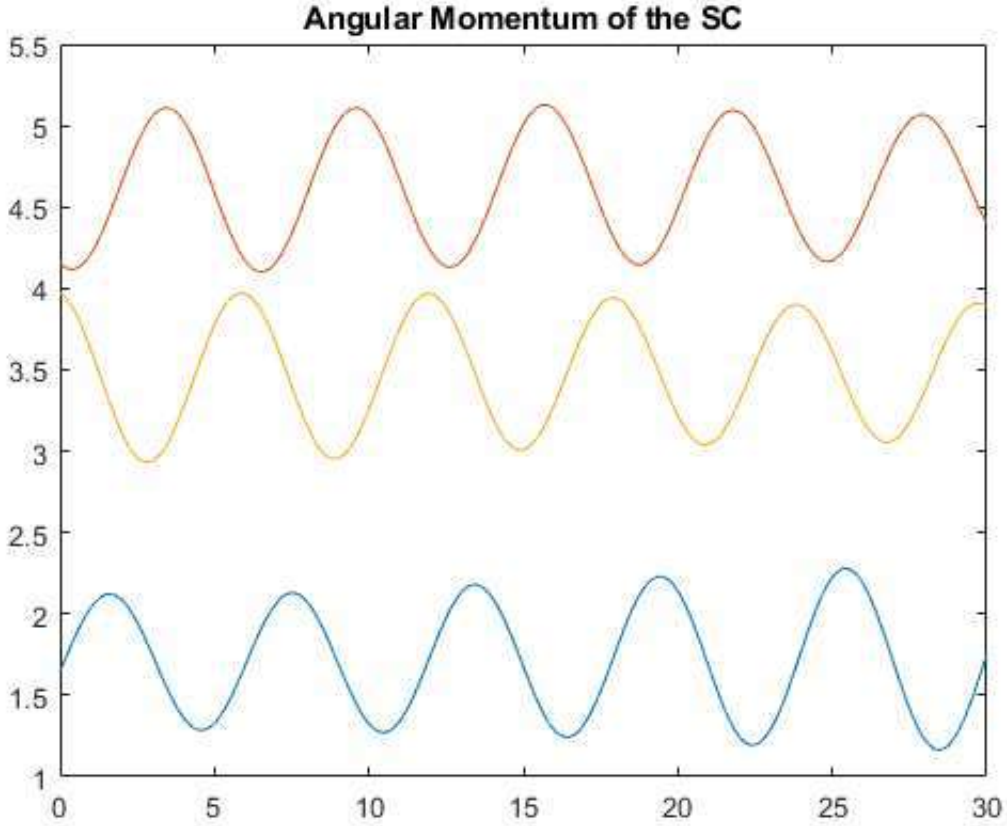
```
    sigma = sys_state(1:3,i);
    omega = sys_state(4:6,i);
    gam = sys_state(7:6+p.nd,i);
    dgam= sys_state(7+p.nd:9+p.nd,i);
    rtr_spd = sys_state(10+p.nd:end,i);

    Gs = (cos(gam - gam0).' .* Gs0) + (sin(gam - gam0).' .*Gt0);
    Gt = (-sin(gam - gam0).' .* Gs0) + (cos(gam - gam0).' .*Gt0);
    for j = 1:p.nd
        ws(j) = Gs(:,j).' * omega;
        wt(j) = Gt(:,j).' * omega;
        wg(j) = Gg(:,j).' * omega;
    end

    % here we need to update the ug and us apriori, and constant through
    L = [0.5*sin(t(i)) -0.5*sin(t(i)) 0.5*cos(t(i))].' ;
    p.L = L;
    p.us = zeros(3,1) ;
    p.ug = zeros(3,1) ;

    % look at the energy, energy rate, and the angular momentum
    Tdot(i) = omega.'*L + sum(dgam.*p.ug) + sum(rtr_spd.*p.us);
    T(i)= (0.5*omega.'*Is*omega) + 0.5*sum(Iws.*((rtr_spd + ws).^2) + ...
        (Igs.*(ws.^2)) + (Jt.*(wt.^2)) + (Jg.*((wg + dgam).^2)));
    if i>1
        Tdot_numer(i-1) = (T(i)-T(i-1))/dt;
    end
    Hg = sum((Igs.*ws).' .*Gs,2) + sum((Igt.*wt).' .*Gt,2) + ...
        sum((Igg.*(wg+dgam)).' .*Gg,2);
    Hw = sum((Iws.*(ws+rtr_spd)).' .*Gs,2) + sum((Iwt.*wt).' .*Gt,2) + ...
        sum((Iwt.*(wg+dgam)).' .*Gg,2);
    Hb = Is*omega;
    NtoB = eye(3) + ((8.*(skew(sigma)*skew(sigma)) - ...
        (4.*(1-norm(sigma)^2)*skew(sigma)))/((1+norm(sigma)^2)^2));
    H(:,i) = NtoB.' *(Hb + Hg + Hw);

    % the first thing is the time, and then all other are derivatives
    k_1 = f_dot(t(i),sys_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, sys_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(sys_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(sys_state(:,i)+k_3*dt), p);
    sys_state(:,i+1) = sys_state(:,i) + (1/6)*(k_1+2*k_2+2*k_3+k_4)*dt;

    % nonsingular attitude check
    s = norm(sys_state(1:3,i+1));
    if s > 1
        sys_state(1:3,i+1) = -(sys_state(1:3,i+1) ./(s^2));
    end
end
toc

% for L, ug, us = 0, lets plot T, Tdot, and Hi
t_span = t(1:end-1);
figure;
plot(t_span,Tdot); hold on;
plot(t_span(1:end-1),Tdot_numer,'--'); hold off;
title('Tdot and Tdotnumerical vs time');
```

```
legend('Tdot','Tdot numerical');
grid on;

% plot the energy by itself
figure;
plot(t_span,T);
legend('T');
title('Kinetic Energy of Spacecraft');

% show that the inertial angmom is constant!
figure;
plot(t_span,H);
title('Angular Momentum of the SC');
```

```
Elapsed time is 9.139903 seconds.
```




Kinetic Energy of Spacecraft
T
31.37
31.36
31.35
31.34
31.33
31.32
31.31
31.3
31.29
0
5
10
15
20
25
30


Angular Momentum of the SC
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
5
10
15
20
25
30

```
%%ASEN6010 Homework 2: Padraig Lysandrou
% In this situation, L is zero, and the inputs are varying
clc; close all; clear all;

% Set up devices, SC inertia, timing
Is = diag([86.215 87 113.565]);
VSCMG_setup;
dt = 0.01;
t = 0:dt:80;
npoints = length(t);

% System gains
% K = 75;
% Kgd=  4.*[0.1 0.1 0.1].';
% Ko = 4.*[0.1 0.1 0.1].';
% P = 5.*[13.13 13.04 15.08].';
K = 1.7;
Kgd=  10.*[0.1 0.1 0.1].';
Ko = 10.*[0.1 0.1 0.1].';
P = 1.*[13.13 13.04 15.08].';
ke = 0.1;
kappa_db = 3;

sys_state = zeros(6+(3*num_devices), npoints);
sigma_0 = [0.414 0.3 0.2].';
omega_0 = [0.03 0.05 -0.01].';
gam0 = [VSCMG(1:num_devices).gam0].';
dgam0 = [VSCMG(1:num_devices).dgam0].';
rtr_spd0 = [VSCMG(1:num_devices).Omega0].';
sys_state(:,1) = [sigma_0.' omega_0.' gam0.' dgam0.' rtr_spd0.'].';

Gs0= [VSCMG(1:num_devices).gs0];
Gt0= [VSCMG(1:num_devices).gt0];
Gg = [VSCMG(1:num_devices).gg] ;
Js = [VSCMG(1:num_devices).Js].'; % !! use element-wise multiplication
Jt = [VSCMG(1:num_devices).Jt].';
Jg = [VSCMG(1:num_devices).Jg].';
Iws= [VSCMG(1:num_devices).Iws].';
Iwt= [VSCMG(1:num_devices).Iwt].';
Igs= [VSCMG(1:num_devices).Igs].';
Igt= [VSCMG(1:num_devices).Igt].';
Igg= [VSCMG(1:num_devices).Igg].';

% just to set things up..
rtr_spd_des = zeros(num_devices, npoints);
rtr_spd_des(:,1) = rtr_spd0;
ug = zeros(num_devices, npoints);
us = zeros(num_devices, npoints);

% what services will the dynamics function need?
p.VSCMG = VSCMG; p.nd = num_devices;
p.Gs0 = Gs0; p.Gt0 = Gt0; p.Gg = Gg; p.Js = Js; p.Jt = Jt; p.Jg = Jg;
p.gam0 = gam0; p.Iws = Iws; p.Is = Is;
f_dot = @(t_in,state_in,param) sc_dyn(t_in,state_in,param);
```

```
% some remaining setup for the loop
Gs = Gs0; Gt = Gt0;
mu = 1;
hbar = Iws(1)*rtr_spd0(1);
Wsi0 = 2*ones(num_devices,1);
Wgi = ones(num_devices,1);
L = zeros(3,1); p.L = L;
Tdot = zeros(1,npoints-1);
Tdot_numer = zeros(1,npoints-2);
T = zeros(1,npoints-1);
H = zeros(3,npoints-1);
ws = zeros(p.nd,1);
wt = zeros(p.nd,1);
wg = zeros(p.nd,1);
drtr_des = zeros(p.nd,npoints);
dgam_des = zeros(p.nd,npoints);
ddgam_des = zeros(p.nd,1);
sigma_err = zeros(3,npoints);
omega_err = zeros(3,npoints);
Lr_s = zeros(3,npoints);
kappa = zeros(1,npoints);

% Generating arbitrary sigma input; wish had more time to do this better
sigma_ref = 0.*[0.1*sin(0.2*t); 0.01*cos(0.2*t); -0.01*sin(0.2*t)];
omega_ref = zeros(3,npoints);
omega_ref(:,1) = omega_0;
% note that initcond is zero here.
for qq = 1:npoints-1
    sr = sigma_ref(:,qq);
    srdot = ((sigma_ref(:,qq+1)-sigma_ref(:,qq))./dt);
    omega_ref(:,qq+1) = (4/((1+sr.'*sr)^2)).* (((1 -(sr.'*sr))*eye(3) -  ...
        2*skew(sr) + (2*sr*(sr.'))) *srdot);
end
omega_ref(:,1) = omega_ref(:,2);


tic
for i = 1:npoints-1
    % Distribute the states from the last cycle for computation
    sigma = sys_state(1:3,i);
    omega = sys_state(4:6,i);
    gam = sys_state(7:6+p.nd,i);
    dgam= sys_state(7+p.nd:6+2*p.nd,i);
    rtr_spd = sys_state(7+2*p.nd:end,i);

    % Updating the pointing vectors of each of the VSCMG devices and wi's
    Gs = (cos(gam - gam0).' .* Gs0) + (sin(gam - gam0).' .*Gt0);
    Gt = (-sin(gam - gam0).' .* Gs0) + (cos(gam - gam0).' .*Gt0);
    ws = Gs.' * omega;
    wt = Gt.' * omega;
    wg = Gg.' * omega;

    % Lets update the inertia matrix of the whole spacecraft
    Ivscmg = 0;
    for k = 1:num_devices
        Ivscmg = p.Js(k)*(Gs(:,k)*(Gs(:,k).')) + p.Jt(k)*(Gt(:,k)*(Gt(:,k).')) ...
            + p.Jg(k)*(Gg(:,k)*(Gg(:,k).')) + Ivscmg;
```

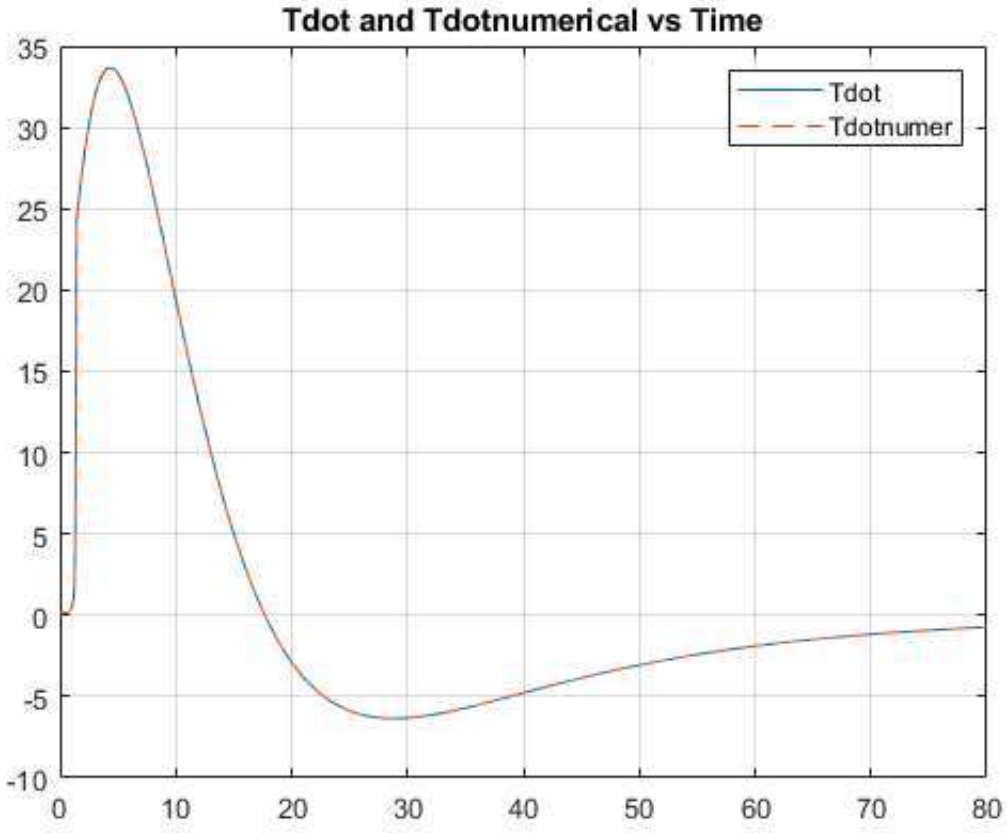
```
end
I = p.Is + Ivscmg;

% Update exogenous torques
L =  zeros(3,1);  p.L = L;

% perform the relMRP and find the DCM
sigma_err(:,i) = relMRP(sigma_ref(:,i), sigma);
DCMerr = MRP2C(sigma_err(:,i));
BFomega_ref = DCMerr*omega_ref(:,i);

% Calculate all of the matrices needed for the attitude control law
D0 = (Iws.') .* Gs;
D1 = (((Iws.*rtr_spd)+((Js./2).*ws)).' .*Gt) + (((Js./2).*wt).' .*Gs);
D2 = (((Jt./2).*wt).' .*Gs) + (((Jt./2).*ws).' .*Gt);
D3 = ((Jg.*wt).' .*Gs) - ((Jg.*ws).' .*Gt) ;
D4 = zeros(3,num_devices);
for k = 1:num_devices
    D4(:,k) = (0.5*(Js(k)-Jt(k)))*((((Gs(:,k)*(Gt(:,k).'))*BFomega_ref)) ...
        + ((Gt(:,k)*(Gs(:,k).'))*BFomega_ref));
end
D = D1 - D2 + D3 + D4;
B =  Jg.' .* Gg;

% Calculate the Required Torque
omega_err(:,i) = omega - BFomega_ref;
domega_ref= (omega_ref(:,i+1) - omega_ref(:,i))./dt;
% calculate the last term of Lr
sum_var = [0 0 0].';
for k = 1:num_devices
    sum_var = Iws(k)*(((rtr_spd(k)*wg(k)).*Gt(:,k)) - ...
        ((rtr_spd(k)*wt(k)).*Gg(:,k))) + sum_var;
end
BFdomega_ref = DCMerr*domega_ref; % put into B frame...
Lr = (-K.*sigma_err(:,i)) -(P.*(omega_err(:,i))) -L  ...
    + ((skew(omega))*I*omega) ...
    + (I*(BFdomega_ref - ((skew(omega))*BFomega_ref))) ...
    + sum_var;
Lr_s(:,i) = Lr;

% Now solving the null motion problem. Here I dont have a preferred
% gimbal setting, but I would like the rotar speed to stick around the
% initial conditions!
Q = [D0 D];
arw = eye(num_devices);
zNN = zeros(num_devices, num_devices);
acmg = eye(num_devices);
A = [arw zNN; zNN acmg];
W_hat = eye(2*num_devices);

[U,S,V] = svd(D);
s3 = S(3,3); s1 = S(1,1);
kappa(i) = s1/s3;
chi = (-(Iws.*rtr_spd + Js.*ws).' .*Gs) + ((Js.*wt).' .*Gt);
ds1dg1 = ((U(:,1).' * chi(:,1))*V(1,1));
ds1dg2 = ((U(:,1).' * chi(:,2))*V(2,1));
ds1dg3 = ((U(:,1).' * chi(:,3))*V(3,1));
```

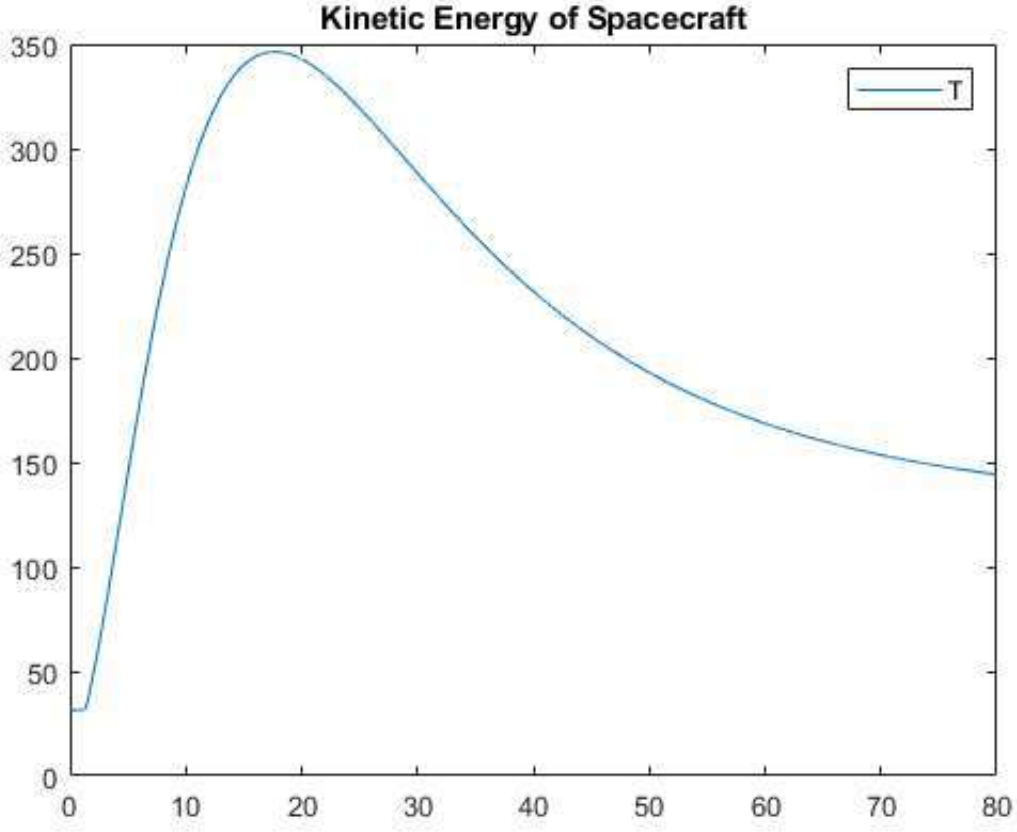
```
ds3dg1 = ((U(:,3).' * chi(:,1))*V(1,3));
ds3dg2 = ((U(:,3).' * chi(:,2))*V(2,3));
ds3dg3 = ((U(:,3).' * chi(:,3))*V(3,3));
dkdg = [((ds1dg1/s3)-((s1/(s3^2))*ds3dg1)); ...
    ((ds1dg2/s3)-((s1/(s3^2))*ds3dg2)); ...
    ((ds1dg3/s3)-((s1/(s3^2))*ds3dg3))];
alpha = 1;
if kappa(i) < kappa_db
    alpha = 0;
end
del_gamma = -alpha.*(1-kappa(i)).*dkdg;
del_vector = [(rtr_spd - rtr_spd0).' del_gamma.'].';
%ke = 0;
etadot_n = ke.*((W_hat*(Q.')*((Q*W_hat*(Q.'))\Q)-eye(2*num_devices)) ...
    *A*del_vector);

delta = det((1/(hbar^2))*(D1 * (D1.')));
Wsi = Wsi0 .* exp(-mu*delta);
W = diag([Wsi.' Wgi.']);
etadot_c = W*(Q.')*((Q*(W*(Q.')))\-Lr);
etadot = etadot_c + etadot_n;
drtr_des(:,i+1) = etadot(1:3);
dgam_des(:,i+1) = etadot(4:6);


% must numerically differentiate and integrate for ddgam_des and
% rtr_spd_des respectively.
rtr_spd_des(:,i+1) = rtr_spd_des(:,i) + dt*drtr_des(:,i+1);
if i > 1
    ddgam_des = (dgam_des(:,i+1) - dgam_des(:,i))./dt;
end
delrtr_spd = rtr_spd - rtr_spd_des(:,i+1);
delgamd = dgam - dgam_des(:,i+1);
p.us = Iws .* (drtr_des(:,i+1) - (Ko .* delrtr_spd) + (dgam.*wt));
p.ug = Jg.*(ddgam_des - (Kgd.*delgamd)) - ((Js-Jt).*(ws.*wt)) ...
    - (Iws.*(rtr_spd.*wt));
us(:,i) = p.us;
ug(:,i) = p.ug;


% Everything below here was Q2 work...
% Lets look at our Energy, energy rate, and inertial frame momentum
Tdot(i) = omega.'*L + sum(dgam.*p.ug) + sum(rtr_spd.*p.us);
T(i)= (0.5*omega.'*Is*omega) + 0.5*sum(Iws.*((rtr_spd + ws).^2) + ...
    (Igs.*(ws.^2)) + (Jt.*(wt.^2)) + (Jg.*((wg + dgam).^2)));
if i>1
    Tdot_numer(i-1) = (T(i)-T(i-1))/dt;
end
Hg = sum((Igs.*ws).' .*Gs,2) + sum((Igt.*wt).' .*Gt,2) + ...
    sum((Igg.*(wg+dgam)).' .*Gg,2);
Hw = sum((Iws.*(ws+rtr_spd)).' .*Gs,2) + sum((Iwt.*wt).' .*Gt,2) + ...
    sum((Iwt.*(wg+dgam)).' .*Gg,2);
Hb = Is*omega;
NtoB = eye(3) + ((8.*(skew(sigma)*skew(sigma)) - ...
    (4.*(1-norm(sigma)^2)*skew(sigma)))/((1+norm(sigma)^2)^2));
H(:,i) = NtoB.' *(Hb + Hg + Hw);
```

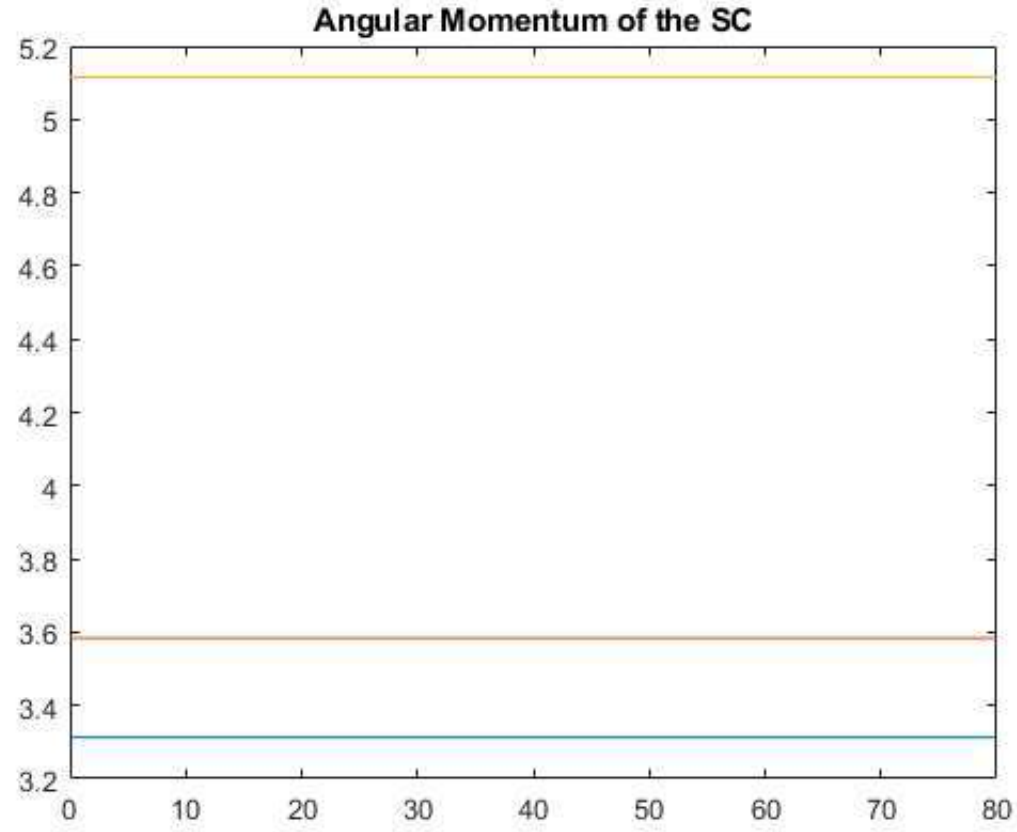
```
    % RK4 step for the spacecraft dynamics
    k_1 = f_dot(t(i),sys_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, sys_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(sys_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(sys_state(:,i)+k_3*dt), p);
    sys_state(:,i+1) = sys_state(:,i) + (1/6)*(k_1+(2*k_2)+(2*k_3)+k_4)*dt;

    % Perform the nonsingular MRP propagation attitude check
    s = norm(sys_state(1:3,i+1));
    if s > 1
        sys_state(1:3,i+1) = -(sys_state(1:3,i+1) ./(s^2));
    end
end
toc

t_span = t(1:end-1);
figure;
plot(t_span, Tdot); hold on;
plot(t_span(1:end-1), Tdot_numer, '--');
title('Tdot and Tdotnumerical vs Time');
legend('Tdot', 'Tdotnumer');
grid on;

figure;
plot(t_span,T);
legend('T');
title('Kinetic Energy of Spacecraft');

figure;
plot(t_span,H);
title('Angular Momentum of the SC');

% % plot the attitude
% figure;
% subplot(3,1,1);
% plot(t,sys_state(1,:)); hold on;
% plot(t,sigma_ref(1,:)); hold off
% title('\sigma_1 vs \sigma_{r1}')
% legend('\sigma_1','\sigma_{r1}');
% subplot(3,1,2);
% plot(t,sys_state(2,:)); hold on;
% plot(t,sigma_ref(2,:)); hold off
% title('\sigma_2 vs \sigma_{r2}')
% legend('\sigma_2','\sigma_{r2}');
% subplot(3,1,3);
% plot(t,sys_state(3,:)); hold on;
% plot(t,sigma_ref(3,:)); hold off
% title('\sigma_3 vs \sigma_{r3}')
% legend('\sigma_3','\sigma_{r3}');
%
% % plot the angular rates
% figure;
% subplot(3,1,1);
% plot(t,sys_state(4,:)); hold on;
% plot(t,omega_ref(1,:)); hold off
% title('\omega_1 vs \omega_{r1}')
% legend('\omega_1','\omega_{r1}');
```

```
% figure;
% plot(t,kappa); hold off;
% title('Singularity index vs time with null motion');
%
%
%
%
```

```
Elapsed time is 3.600604 seconds.
```




Kinetic Energy of Spacecraft
T
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80


Angular Momentum of the SC
5.2
5
4.8
4.6
4.4
4.2
4
3.8
3.6
3.4
3.2
0
10
20
30
40
50
60
70
80

Question 3/4 Work Demonstrated Here:

```
%%ASEN6010 Homework 2: Padraig Lysandrou
% The only situation I is that varying, and the inputs are zero
clc; close all; clear all;

% Set up devices, SC inertia, timing
Is = diag([20 10 15]);
VSCMG_setup;
dt = 0.01;
t = 0:dt:200;
npoints = length(t);

% System gains
% K = 75;
% Kgd=  4.*[0.1 0.1 0.1].';
% Ko = 4.*[0.1 0.1 0.1].';
% P = 5.*[13.13 13.04 15.08].';
K = 1.7;
Kgd=  10.*[0.1 0.1 0.1].';
Ko = 10.*[0.1 0.1 0.1].';
P = 1.*[13.13 13.04 15.08].';

sys_state = zeros(6+(3*num_devices), npoints);
sigma_0 = [0.414 0.3 0.2].';
omega_0 = [0.03 0.05 -0.01].';
gam0 = [VSCMG(1:num_devices).gam0].';
dgam0 = [VSCMG(1:num_devices).dgam0].';
rtr_spd0 = [VSCMG(1:num_devices).Omega0].';
sys_state(:,1) = [sigma_0.' omega_0.' gam0.' dgam0.' rtr_spd0.'].';

Gs0= [VSCMG(1:num_devices).gs0];
Gt0= [VSCMG(1:num_devices).gt0];
Gg = [VSCMG(1:num_devices).gg] ;
Js = [VSCMG(1:num_devices).Js].'; % !! use element-wise multiplication
Jt = [VSCMG(1:num_devices).Jt].';
Jg = [VSCMG(1:num_devices).Jg].';
Iws= [VSCMG(1:num_devices).Iws].';
Iwt= [VSCMG(1:num_devices).Iwt].';
Igs= [VSCMG(1:num_devices).Igs].';
Igt= [VSCMG(1:num_devices).Igt].';
Igg= [VSCMG(1:num_devices).Igg].';

% just to set things up..
rtr_spd_des = zeros(num_devices, npoints);
rtr_spd_des(:,1) = rtr_spd0;
ug = zeros(num_devices, npoints);
us = zeros(num_devices, npoints);

% what services will the dynamics function need?
p.VSCMG = VSCMG; p.nd = num_devices;
p.Gs0 = Gs0; p.Gt0 = Gt0; p.Gg = Gg; p.Js = Js; p.Jt = Jt; p.Jg = Jg;
p.gam0 = gam0; p.Iws = Iws; p.Is = Is;
f_dot = @(t_in,state_in,param) sc_dyn(t_in,state_in,param);

% some remaining setup for the loop
Gs = Gs0; Gt = Gt0;
```

```
mu = 1;
hbar = Iws(1)*rtr_spd0(1);
Wsi0 = 2*ones(num_devices,1);
Wgi = ones(num_devices,1);
L = zeros(3,1); p.L = L;
Tdot = zeros(1,npoints-1);
Tdot_numer = zeros(1,npoints-2);
T = zeros(1,npoints-1);
H = zeros(3,npoints-1);
ws = zeros(p.nd,1);
wt = zeros(p.nd,1);
wg = zeros(p.nd,1);
drtr_des = zeros(p.nd,npoints);
dgam_des = zeros(p.nd,npoints);
ddgam_des = zeros(p.nd,1);
sigma_err = zeros(3,npoints);
omega_err = zeros(3,npoints);
Lr_s = zeros(3,npoints);

% Generating arbitrary sigma input; wish had more time to do this better
sigma_ref = 1.*[0.1*sin(0.2*t); 0.01*cos(0.2*t); -0.01*sin(0.2*t)];
omega_ref = zeros(3,npoints);
omega_ref(:,1) = omega_0;
% note that initcond is zero here.
for qq = 1:npoints-1
    sr = sigma_ref(:,qq);
    srdot = ((sigma_ref(:,qq+1)-sigma_ref(:,qq))./dt);
    omega_ref(:,qq+1) = (4/((1+sr.'*sr)^2)).* (((1 -(sr.'*sr))*eye(3) -  ...
        2*skew(sr) + (2*sr*(sr.'))) *srdot);
end
omega_ref(:,1) = omega_ref(:,2);


tic
for i = 1:npoints-1
    % Distribute the states from the last cycle for computation
    sigma = sys_state(1:3,i);
    omega = sys_state(4:6,i);
    gam = sys_state(7:6+p.nd,i);
    dgam= sys_state(7+p.nd:9+p.nd,i);
    rtr_spd = sys_state(10+p.nd:end,i);

    % Updating the pointing vectors of each of the VSCMG devices and wi's
    Gs = (cos(gam - gam0).' .* Gs0) + (sin(gam - gam0).' .*Gt0);
    Gt = (-sin(gam - gam0).' .* Gs0) + (cos(gam - gam0).' .*Gt0);
    ws = Gs.' * omega;
    wt = Gt.' * omega;
    wg = Gg.' * omega;

    % Lets update the inertia matrix of the whole spacecraft
    Ivscmg = 0;
    for k = 1:num_devices
        Ivscmg = p.Js(k)*(Gs(:,k)*(Gs(:,k).')) + p.Jt(k)*(Gt(:,k)*(Gt(:,k).')) ...
            + p.Jg(k)*(Gg(:,k)*(Gg(:,k).')) + Ivscmg;
    end
    I = p.Is + Ivscmg;
```

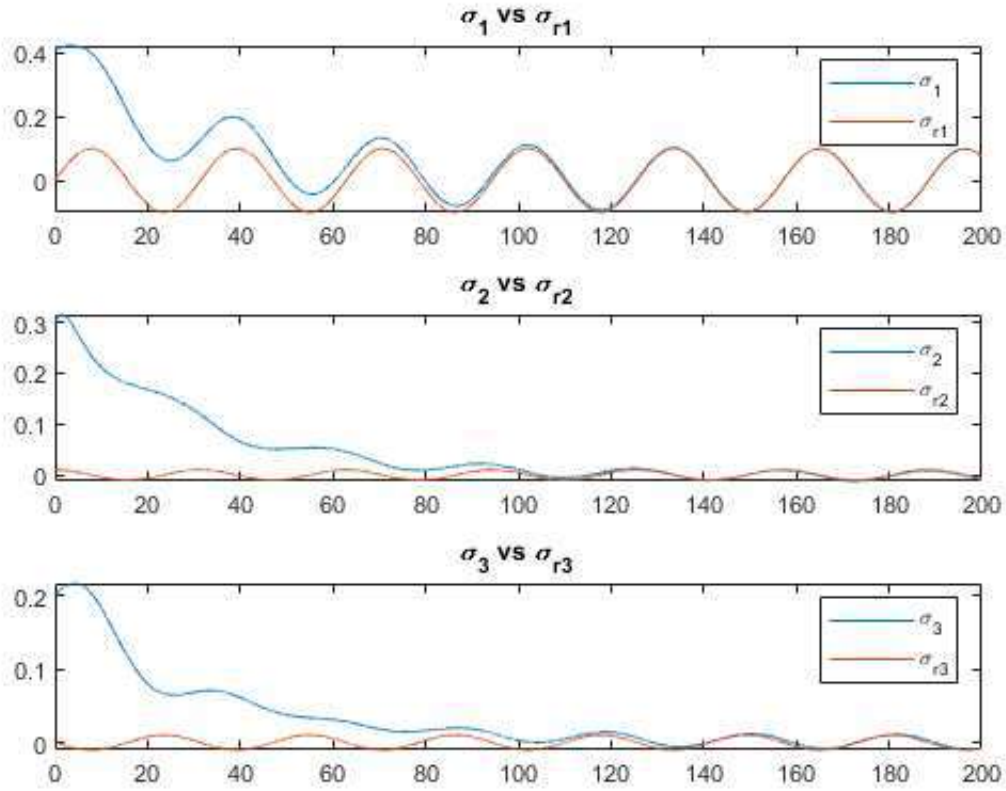
```
% Update exogenous torques
L =  zeros(3,1);  p.L = L;

% perform the relMRP and find the DCM
sigma_err(:,i) = relMRP(sigma_ref(:,i), sigma);
DCMerr = MRP2C(sigma_err(:,i));
BFomega_ref = DCMerr*omega_ref(:,i);

% Calculate all of the matrices needed for the attitude control law
D0 = (Iws.') .* Gs;
D1 = (((Iws.*rtr_spd)+((Js./2).*ws)).' .*Gt) + (((Js./2).*wt).' .*Gs);
D2 = (((Jt./2).*wt).' .*Gs) + (((Jt./2).*ws).' .*Gt);
D3 = ((Jg.*wt).' .*Gs) - ((Jg.*ws).' .*Gt) ;
D4 = zeros(3,num_devices);
for k = 1:num_devices
    D4(:,k) = (0.5*(Js(k)-Jt(k)))*((((Gs(:,k)*(Gt(:,k).'))*BFomega_ref)) ...
        + ((Gt(:,k)*(Gs(:,k).'))*BFomega_ref));
end
D = D1 - D2 + D3 + D4;
B =  Jg.' .* Gg;

% Calculate the Required Torque
omega_err(:,i) = omega - BFomega_ref;
domega_ref= (omega_ref(:,i+1) - omega_ref(:,i))./dt;
% calculate the last term of Lr
sum_var = [0 0 0].';
for k = 1:num_devices
    sum_var = Iws(k)*(((rtr_spd(k)*wg(k)).*Gt(:,k)) - ...
        ((rtr_spd(k)*wt(k)).*Gg(:,k))) + sum_var;
end
BFdomega_ref = DCMerr*domega_ref; % put into B frame...
Lr = (-K.*sigma_err(:,i)) -(P.*(omega_err(:,i))) -L  ...
    + ((skew(omega))*I*omega) ...
    + (I*(BFdomega_ref - ((skew(omega))*BFomega_ref))) ...
    + sum_var;
Lr_s(:,i) = Lr;

% Here we solve the inverse problem, note that initcond is zero here...
Q = [D0 D];
delta = det((1/(hbar^2))*(D1 * (D1.')));
Wsi = Wsi0 .* exp(-mu*delta);
W = diag([Wsi.' Wgi.']);
etadot = W*(Q.')*((Q*(W*(Q.')))\-Lr);
drtr_des(:,i+1) = etadot(1:3);
dgam_des(:,i+1) = etadot(4:6);


% must numerically differentiate and integrate for ddgam_des and
% rtr_spd_des respectively.
rtr_spd_des(:,i+1) = rtr_spd_des(:,i) + dt*drtr_des(:,i+1);
if i > 1
    ddgam_des = (dgam_des(:,i+1) - dgam_des(:,i))./dt;
end
delrtr_spd = rtr_spd - rtr_spd_des(:,i+1);
delgamd = dgam - dgam_des(:,i+1);
p.us = Iws .* (drtr_des(:,i+1) - (Ko .* delrtr_spd) + (dgam.*wt));
p.ug = Jg.*(ddgam_des - (Kgd.*delgamd)) - ((Js-Jt).*(ws.*wt)) ...
```

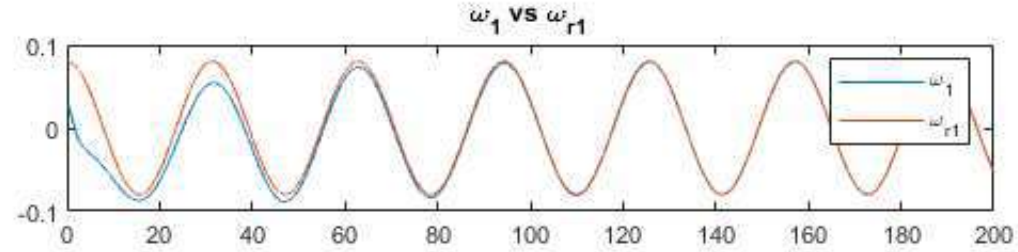
```
        - (Iws.*(rtr_spd.*wt));
    us(:,i) = p.us;
    ug(:,i) = p.ug;
%     p.us = zeros(num_devices,1);
%     p.ug = zeros(num_devices,1);

    % Everything below here was Q2 work...
    % Lets look at our Energy, energy rate, and inertial frame momentum
%     Tdot(i) = omega.'*L + sum(dgam.*p.ug) + sum(rtr_spd.*p.us);
%     T(i)= (0.5*omega.'*Is*omega) + 0.5*sum(Iws.*((rtr_spd + ws).^2) + ...
%         (Igs.*(ws.^2)) + (Jt.*(wt.^2)) + (Jg.*((wg + dgam).^2)));
%     if i>1
%         Tdot_numer(i-1) = (T(i)-T(i-1))/dt;
%     end
%     Hg = sum((Igs.*ws).' .*Gs,2) + sum((Igt.*wt).' .*Gt,2) + ...
%         sum((Igg.*(wg+dgam)).' .*Gg,2);
%     Hw = sum((Iws.*(ws+rtr_spd)).' .*Gs,2) + sum((Iwt.*wt).' .*Gt,2) + ...
%         sum((Iwt.*(wg+dgam)).' .*Gg,2);
%     Hb = Is*omega;
%     NtoB = eye(3) + ((8.*(skew(sigma)*skew(sigma)) - ...
%         (4.*(1-norm(sigma)^2)*skew(sigma)))/((1+norm(sigma)^2)^2));
%     H(:,i) = NtoB.' *(Hb + Hg + Hw);

    % RK4 step for the spacecraft dynamics
    k_1 = f_dot(t(i),sys_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, sys_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(sys_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(sys_state(:,i)+k_3*dt), p);
    sys_state(:,i+1) = sys_state(:,i) + (1/6)*(k_1+(2*k_2)+(2*k_3)+k_4)*dt;

    % Perform the nonsingular MRP propagation attitude check
    s = norm(sys_state(1:3,i+1));
    if s > 1
        sys_state(1:3,i+1) = -(sys_state(1:3,i+1) ./(s^2));
    end
end
toc

% for L, ug, us = 0, lets plot T, Tdot, and Hi
% t_span = t(1:end-1);
% figure;
% plot(t_span,Tdot); hold on;
% plot(t_span(1:end-1),Tdot_numer,'--'); hold off;
% title('Tdot and Tdotnumerical vs time');
% legend('Tdot','Tdot numerical');
% grid on;
%
% % plot the energy by itself
% figure;
% plot(t_span,T);
% legend('T');
% title('Kinetic Energy of Spacecraft');
%
% % show that the inertial angmom is constant!
% figure;
% plot(t_span,H);
% title('Angular Momentum of the SC');
```

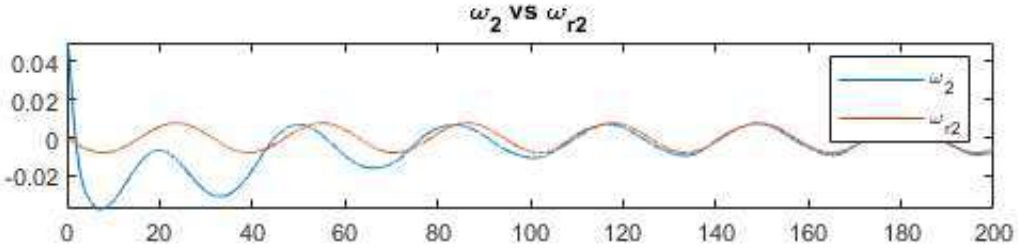
```
% plot the attitude
figure;
subplot(3,1,1);
plot(t,sys_state(1,:)); hold on;
plot(t,sigma_ref(1,:)); hold off
title('\sigma_1 vs \sigma_{r1}')
legend('\sigma_1','\sigma_{r1}');
subplot(3,1,2);
plot(t,sys_state(2,:)); hold on;
plot(t,sigma_ref(2,:)); hold off
title('\sigma_2 vs \sigma_{r2}')
legend('\sigma_2','\sigma_{r2}');
subplot(3,1,3);
plot(t,sys_state(3,:)); hold on;
plot(t,sigma_ref(3,:)); hold off
title('\sigma_3 vs \sigma_{r3}')
legend('\sigma_3','\sigma_{r3}');

% plot the angular rates
figure;
subplot(3,1,1);
plot(t,sys_state(4,:)); hold on;
plot(t,omega_ref(1,:)); hold off
title('\omega_1 vs \omega_{r1}')
legend('\omega_1','\omega_{r1}');
subplot(3,1,2);
plot(t,sys_state(5,:)); hold on;
plot(t,omega_ref(2,:)); hold off
title('\omega_2 vs \omega_{r2}')
legend('\omega_2','\omega_{r2}');
subplot(3,1,3);
plot(t,sys_state(6,:)); hold on;
plot(t,omega_ref(3,:)); hold off
title('\omega_3 vs \omega_{r3}')
legend('\omega_3','\omega_{r3}');


% plot sigmaBR and omegaBR
figure;
subplot(3,1,1);
plot(t,sigma_err(1,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\sigma_1 BR')
subplot(3,1,2);
plot(t,sigma_err(2,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\sigma_2 BR')
subplot(3,1,3);
plot(t,sigma_err(3,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\sigma_3 BR')


% plot sigmaBR and omegaBR
figure;
subplot(3,1,1);
```

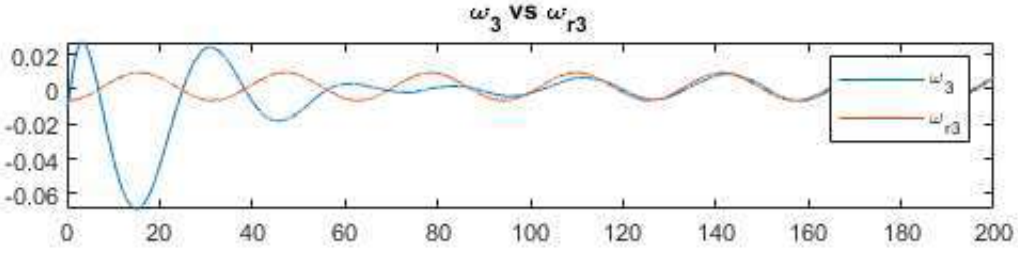
```
plot(t,omega_err(1,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_1 BR')
subplot(3,1,2);
plot(t,omega_err(2,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_2 BR')
subplot(3,1,3);
plot(t,omega_err(3,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_3 BR')

% plot us and ug
figure;
plot(t,us); hold on;
plot(t,ug); hold off;
title('Control inputs Us and Ug vs time');
legend('us', 'ug');

% plot the rotor speeds
 figure;
 plot(t,sys_state(10+p.nd:end,:)); hold off;
title('Rotor speeds vs time');
legend('\Omega_1', '\Omega_2','\Omega_3');
```

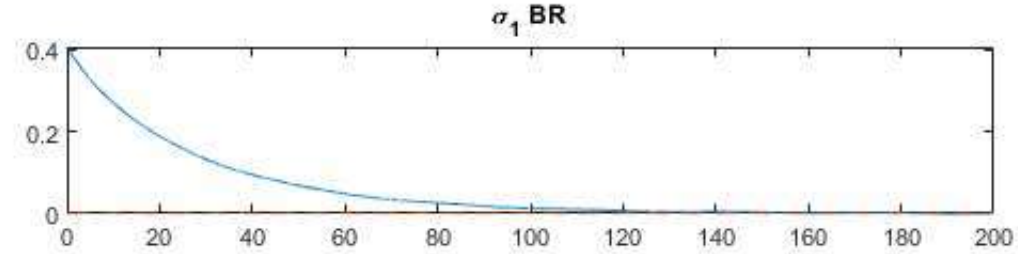
```
Elapsed time is 6.966939 seconds.
```

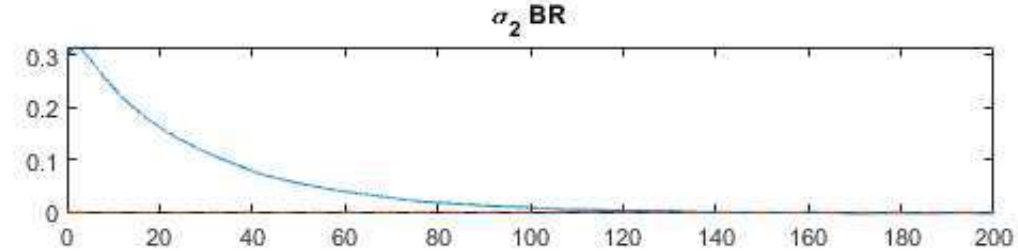


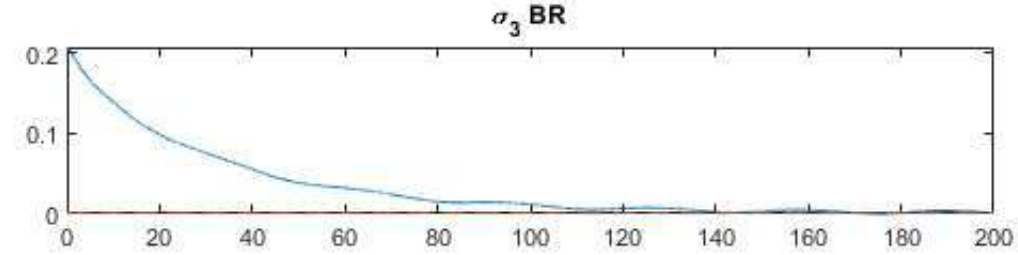

ω_1 vs ω_{r1}
ω_1
ω_{r1}

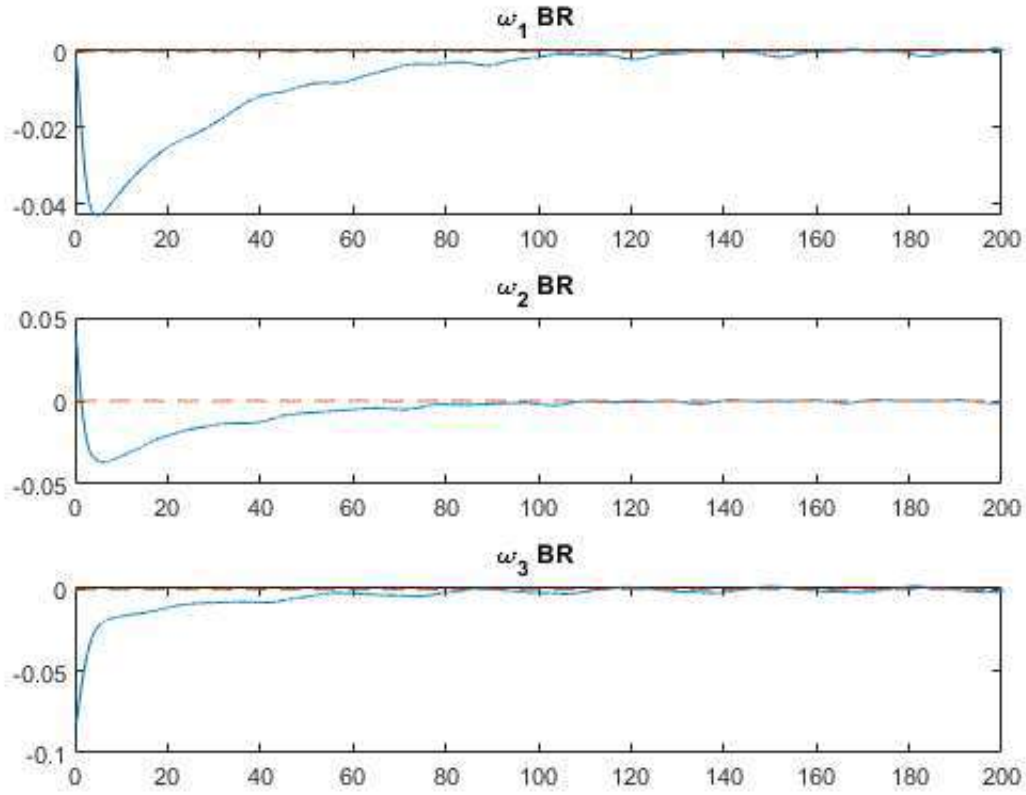

ω_2 vs ω_{r2}
ω_2
ω_{r2}

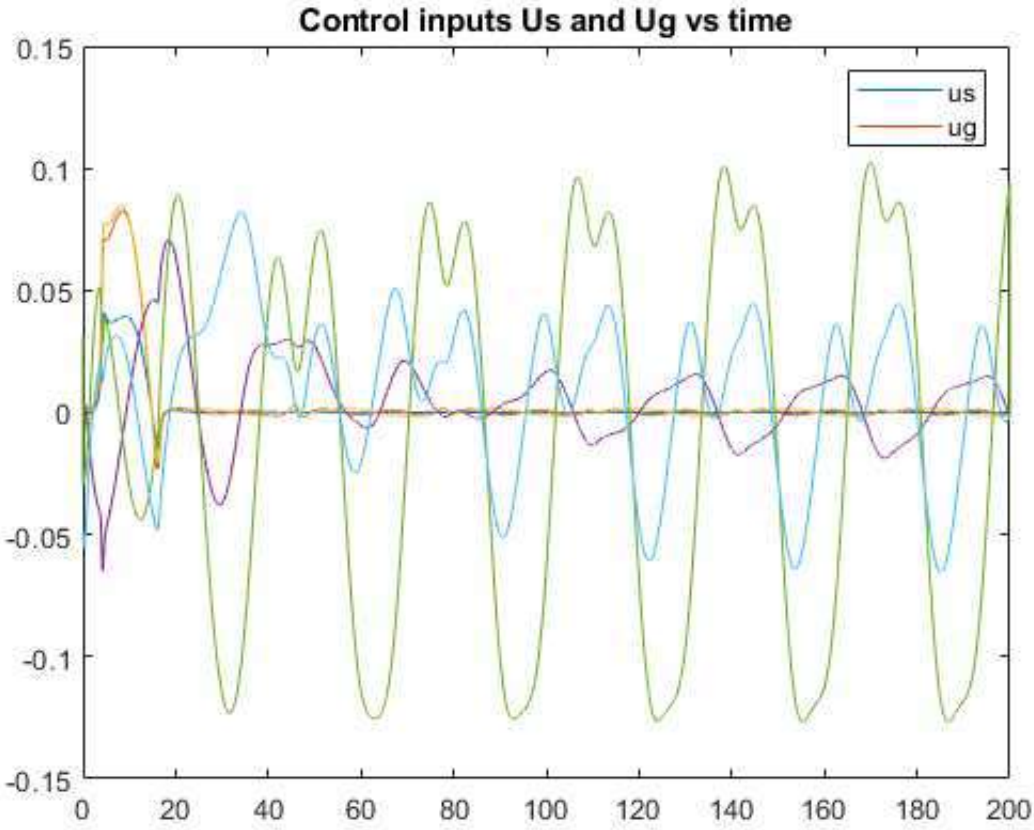

ω_3 vs ω_{r3}
ω_3
ω_{r3}

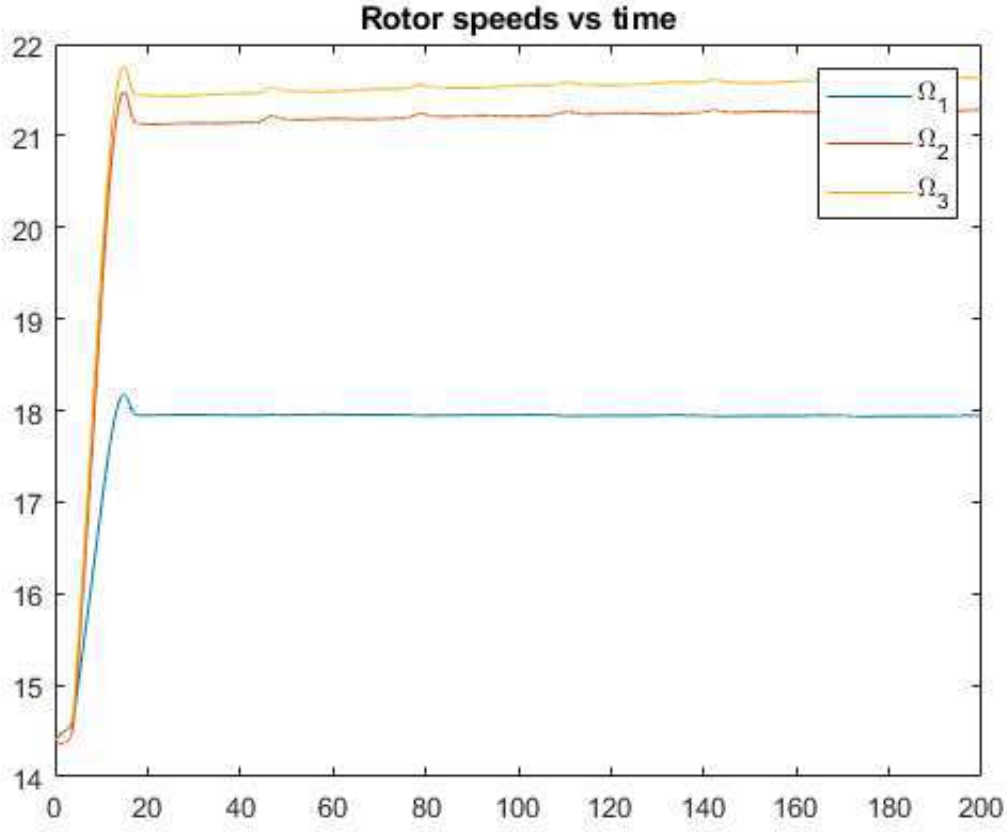

σ_1 BR


σ_2 BR


σ_3 BR


ω_1 BR
0
-0.02
-0.04
0 20 40 60 80 100 120 140 160 180 200
ω_2 BR
0.05
0
-0.05
0 20 40 60 80 100 120 140 160 180 200
ω_3 BR
0
-0.05
-0.1
0 20 40 60 80 100 120 140 160 180 200


Control inputs Us and Ug vs time
us
ug
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0 20 40 60 80 100 120 140 160 180 200



Published with MATLAB® R2018a

Question 5: Null Motion Implementation

```
%%ASEN6010 Homework 2: Padraig Lysandrou
% In this situation, Is is the 20 10 15, and the inputs are zero
clc; close all; clear all;

% Set up devices, SC inertia, timing
Is = diag([20 10 15]);
VSCMG_setup;
dt = 0.01;
t = 0:dt:80;
npoints = length(t);

% System gains
% K = 75;
% Kgd=  4.*[0.1 0.1 0.1].';
% Ko = 4.*[0.1 0.1 0.1].';
% P = 5.*[13.13 13.04 15.08].';
K = 1.7;
Kgd=  10.*[0.1 0.1 0.1].';
Ko = 10.*[0.1 0.1 0.1].';
P = 1.*[13.13 13.04 15.08].';
ke = 0.1;
kappa_db = 3;

sys_state = zeros(6+(3*num_devices), npoints);
sigma_0 = [0.414 0.3 0.2].';
omega_0 = [0.03 0.05 -0.01].';
gam0 = [VSCMG(1:num_devices).gam0].';
dgam0 = [VSCMG(1:num_devices).dgam0].';
rtr_spd0 = [VSCMG(1:num_devices).Omega0].';
sys_state(:,1) = [sigma_0.' omega_0.' gam0.' dgam0.' rtr_spd0.'].';

Gs0= [VSCMG(1:num_devices).gs0];
Gt0= [VSCMG(1:num_devices).gt0];
Gg = [VSCMG(1:num_devices).gg] ;
Js = [VSCMG(1:num_devices).Js].'; % !! use element-wise multiplication
Jt = [VSCMG(1:num_devices).Jt].';
Jg = [VSCMG(1:num_devices).Jg].';
Iws= [VSCMG(1:num_devices).Iws].';
Iwt= [VSCMG(1:num_devices).Iwt].';
Igs= [VSCMG(1:num_devices).Igs].';
Igt= [VSCMG(1:num_devices).Igt].';
Igg= [VSCMG(1:num_devices).Igg].';

% just to set things up..
rtr_spd_des = zeros(num_devices, npoints);
rtr_spd_des(:,1) = rtr_spd0;
ug = zeros(num_devices, npoints);
us = zeros(num_devices, npoints);

% what services will the dynamics function need?
p.VSCMG = VSCMG; p.nd = num_devices;
p.Gs0 = Gs0; p.Gt0 = Gt0; p.Gg = Gg; p.Js = Js; p.Jt = Jt; p.Jg = Jg;
p.gam0 = gam0; p.Iws = Iws; p.Is = Is;
f_dot = @(t_in,state_in,param) sc_dyn(t_in,state_in,param);
```

```
% some remaining setup for the loop
Gs = Gs0; Gt = Gt0;
mu = 1;
hbar = Iws(1)*rtr_spd0(1);
Wsi0 = 2*ones(num_devices,1);
Wgi = ones(num_devices,1);
L = zeros(3,1); p.L = L;
Tdot = zeros(1,npoints-1);
Tdot_numer = zeros(1,npoints-2);
T = zeros(1,npoints-1);
H = zeros(3,npoints-1);
ws = zeros(p.nd,1);
wt = zeros(p.nd,1);
wg = zeros(p.nd,1);
drtr_des = zeros(p.nd,npoints);
dgam_des = zeros(p.nd,npoints);
ddgam_des = zeros(p.nd,1);
sigma_err = zeros(3,npoints);
omega_err = zeros(3,npoints);
Lr_s = zeros(3,npoints);
kappa = zeros(1,npoints);

% Generating arbitrary sigma input; wish had more time to do this better
sigma_ref = 0.*[0.1*sin(0.2*t); 0.01*cos(0.2*t); -0.01*sin(0.2*t)];
omega_ref = zeros(3,npoints);
omega_ref(:,1) = omega_0;
% note that initcond is zero here.
for qq = 1:npoints-1
    sr = sigma_ref(:,qq);
    srdot = ((sigma_ref(:,qq+1)-sigma_ref(:,qq))./dt);
    omega_ref(:,qq+1) = (4/((1+sr.'*sr)^2)).* (((1 -(sr.'*sr))*eye(3) -  ...
        2*skew(sr) + (2*sr*(sr.'))) *srdot);
end
omega_ref(:,1) = omega_ref(:,2);


tic
for i = 1:npoints-1
    % Distribute the states from the last cycle for computation
    sigma = sys_state(1:3,i);
    omega = sys_state(4:6,i);
    gam = sys_state(7:6+p.nd,i);
    dgam= sys_state(7+p.nd:6+2*p.nd,i);
    rtr_spd = sys_state(7+2*p.nd:end,i);

    % Updating the pointing vectors of each of the VSCMG devices and wi's
    Gs = (cos(gam - gam0).' .* Gs0) + (sin(gam - gam0).' .*Gt0);
    Gt = (-sin(gam - gam0).' .* Gs0) + (cos(gam - gam0).' .*Gt0);
    ws = Gs.' * omega;
    wt = Gt.' * omega;
    wg = Gg.' * omega;

    % Lets update the inertia matrix of the whole spacecraft
    Ivscmg = 0;
    for k = 1:num_devices
        Ivscmg = p.Js(k)*(Gs(:,k)*(Gs(:,k).')) + p.Jt(k)*(Gt(:,k)*(Gt(:,k).')) ...
            + p.Jg(k)*(Gg(:,k)*(Gg(:,k).')) + Ivscmg;
```

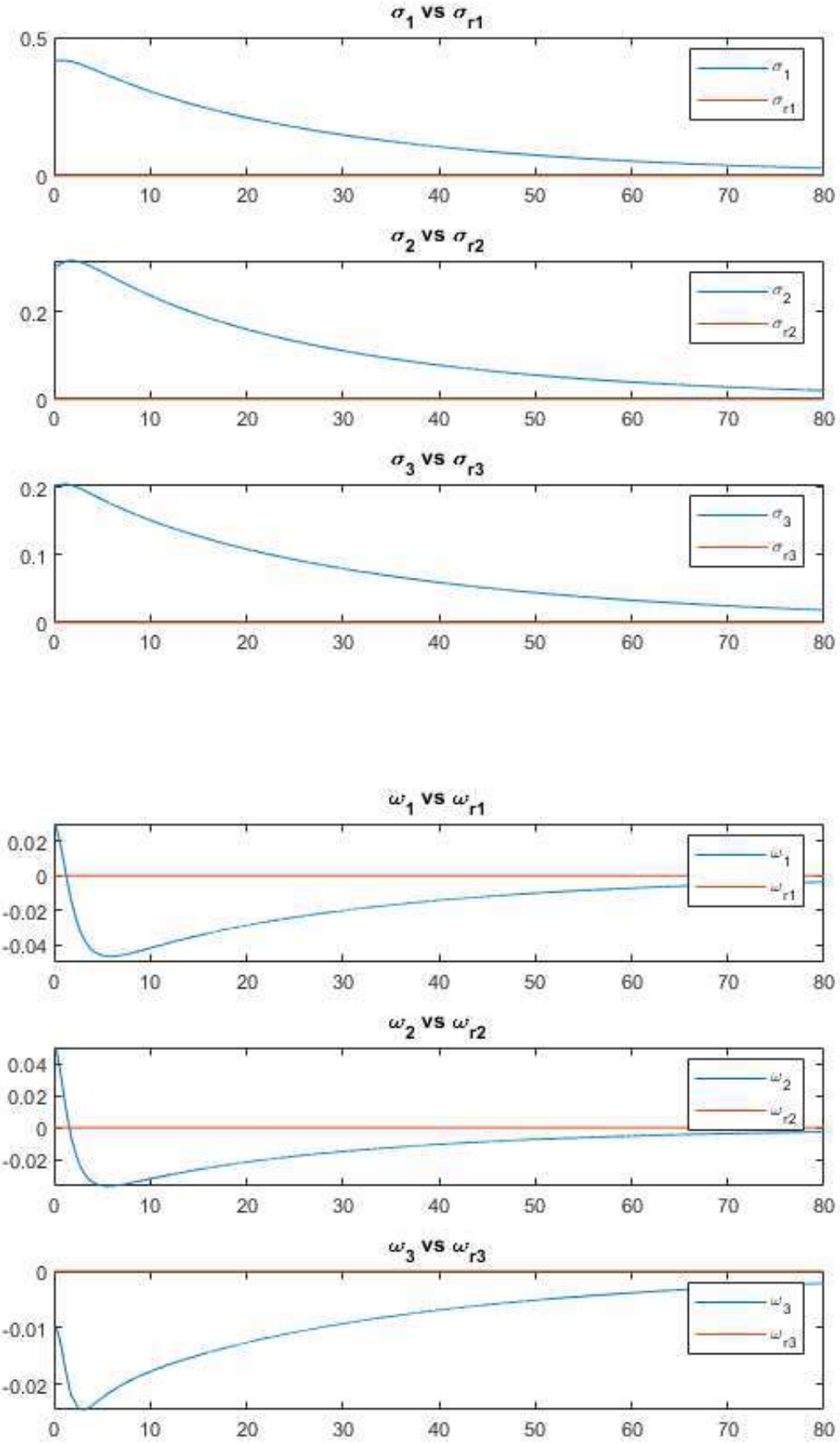
```
    end
    I = p.Is + Ivscmg;

    % Update exogenous torques
    L =  zeros(3,1);  p.L = L;

    % perform the relMRP and find the DCM
    sigma_err(:,i) = relMRP(sigma_ref(:,i), sigma);
    DCMerr = MRP2C(sigma_err(:,i));
    BFomega_ref = DCMerr*omega_ref(:,i);

    % Calculate all of the matrices needed for the attitude control law
    D0 = (Iws.') .* Gs;
    D1 = (((Iws.*rtr_spd)+((Js./2).*ws)).' .*Gt) + (((Js./2).*wt).' .*Gs);
    D2 = (((Jt./2).*wt).' .*Gs) + (((Jt./2).*ws).' .*Gt);
    D3 = ((Jg.*wt).' .*Gs) - ((Jg.*ws).' .*Gt) ;
    D4 = zeros(3,num_devices);
    for k = 1:num_devices
        D4(:,k) = (0.5*(Js(k)-Jt(k)))*((((Gs(:,k)*(Gt(:,k).'))*BFomega_ref)) ...
            + ((Gt(:,k)*(Gs(:,k).'))*BFomega_ref));
    end
    D = D1 - D2 + D3 + D4;
    B =  Jg.' .* Gg;

    % Calculate the Required Torque
    omega_err(:,i) = omega - BFomega_ref;
    domega_ref= (omega_ref(:,i+1) - omega_ref(:,i))./dt;
    % calculate the last term of Lr
    sum_var = [0 0 0].';
    for k = 1:num_devices
        sum_var = Iws(k)*(((rtr_spd(k)*wg(k)).*Gt(:,k)) - ...
            ((rtr_spd(k)*wt(k)).*Gg(:,k))) + sum_var;
    end
    BFdomega_ref = DCMerr*domega_ref; % put into B frame...
    Lr = (-K.*sigma_err(:,i)) -(P.*(omega_err(:,i))) -L  ...
        + ((skew(omega))*I*omega) ...
        + (I*(BFdomega_ref - ((skew(omega))*BFomega_ref))) ...
        + sum_var;
    Lr_s(:,i) = Lr;

    % Now solving the null motion problem. Here I dont have a preferred
    % gimbal setting, but I would like the rotar speed to stick around the
    % initial conditions!
    Q = [D0 D];
    arw = eye(num_devices);
    zNN = zeros(num_devices, num_devices);
    acmg = eye(num_devices);
    A = [arw zNN; zNN acmg];
    W_hat = eye(2*num_devices);

    [U,S,V] = svd(D);
    s3 = S(3,3); s1 = S(1,1);
    kappa(i) = s1/s3;
    chi = (-(Iws.*rtr_spd + Js.*ws).' .*Gs) + ((Js.*wt).' .*Gt);
    ds1dg1 = ((U(:,1).' * chi(:,1))*V(1,1));
    ds1dg2 = ((U(:,1).' * chi(:,2))*V(2,1));
    ds1dg3 = ((U(:,1).' * chi(:,3))*V(3,1));
```

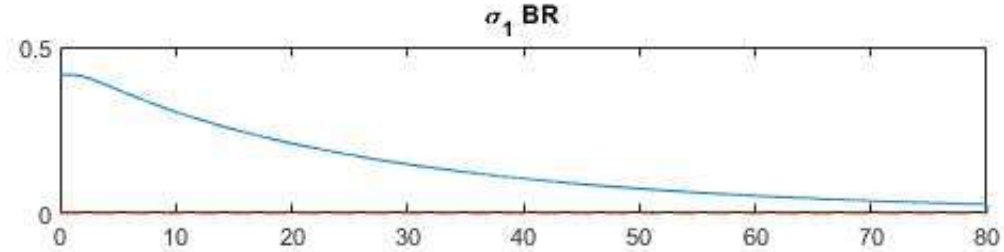
```
    ds3dg1 = ((U(:,3).' * chi(:,1))*V(1,3));
    ds3dg2 = ((U(:,3).' * chi(:,2))*V(2,3));
    ds3dg3 = ((U(:,3).' * chi(:,3))*V(3,3));
    dkdg = [((ds1dg1/s3)-((s1/(s3^2))*ds3dg1)); ...
        ((ds1dg2/s3)-((s1/(s3^2))*ds3dg2)); ...
        ((ds1dg3/s3)-((s1/(s3^2))*ds3dg3))];
    alpha = 1;
    if kappa(i) < kappa_db
        alpha = 0;
    end
    del_gamma = -alpha.*(1-kappa(i)).*dkdg;
    del_vector = [(rtr_spd - rtr_spd0).' del_gamma.'].';
    %ke = 0;
    etadot_n = ke.*((W_hat*(Q.')*((Q*W_hat*(Q.'))\Q)-eye(2*num_devices)) ...
        *A*del_vector);

    delta = det((1/(hbar^2))*(D1 * (D1.')));
    Wsi = Wsi0 .* exp(-mu*delta);
    W = diag([Wsi.' Wgi.']);
    etadot_c = W*(Q.')*((Q*(W*(Q.')))\-Lr);
    etadot = etadot_c + etadot_n;
    drtr_des(:,i+1) = etadot(1:3);
    dgam_des(:,i+1) = etadot(4:6);


    % must numerically differentiate and integrate for ddgam_des and
    % rtr_spd_des respectively.
    rtr_spd_des(:,i+1) = rtr_spd_des(:,i) + dt*drtr_des(:,i+1);
    if i > 1
        ddgam_des = (dgam_des(:,i+1) - dgam_des(:,i))./dt;
    end
    delrtr_spd = rtr_spd - rtr_spd_des(:,i+1);
    delgamd = dgam - dgam_des(:,i+1);
    p.us = Iws .* (drtr_des(:,i+1) - (Ko .* delrtr_spd) + (dgam.*wt));
    p.ug = Jg.*(ddgam_des - (Kgd.*delgamd)) - ((Js-Jt).*(ws.*wt)) ...
        - (Iws.*(rtr_spd.*wt));
    us(:,i) = p.us;
    ug(:,i) = p.ug;


    % Everything below here was Q2 work...
    % Lets look at our Energy, energy rate, and inertial frame momentum
%     Tdot(i) = omega.'*L + sum(dgam.*p.ug) + sum(rtr_spd.*p.us);
%     T(i)= (0.5*omega.'*Is*omega) + 0.5*sum(Iws.*((rtr_spd + ws).^2) + ...
%         (Igs.*(ws.^2)) + (Jt.*(wt.^2)) + (Jg.*((wg + dgam).^2)));
%     if i>1
%         Tdot_numer(i-1) = (T(i)-T(i-1))/dt;
%     end
%     Hg = sum((Igs.*ws).' .*Gs,2) + sum((Igt.*wt).' .*Gt,2) + ...
%         sum((Igg.*(wg+dgam)).' .*Gg,2);
%     Hw = sum((Iws.*(ws+rtr_spd)).' .*Gs,2) + sum((Iwt.*wt).' .*Gt,2) + ...
%         sum((Iwt.*(wg+dgam)).' .*Gg,2);
%     Hb = Is*omega;
%     NtoB = eye(3) + ((8.*(skew(sigma)*skew(sigma)) - ...
%         (4.*(1-norm(sigma)^2)*skew(sigma)))/((1+norm(sigma)^2)^2));
%     H(:,i) = NtoB.' *(Hb + Hg + Hw);
```

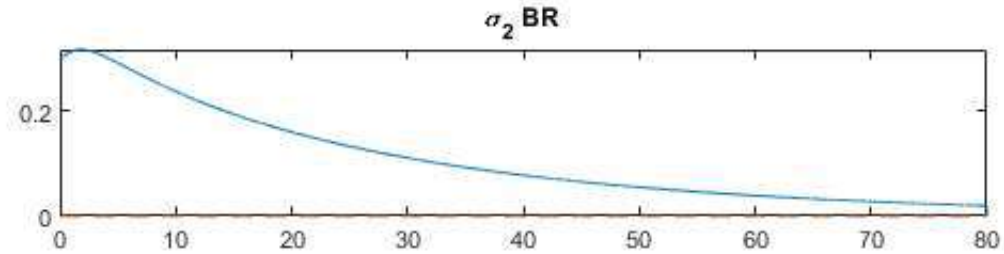
```
    % RK4 step for the spacecraft dynamics
    k_1 = f_dot(t(i),sys_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, sys_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(sys_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(sys_state(:,i)+k_3*dt), p);
    sys_state(:,i+1) = sys_state(:,i) + (1/6)*(k_1+(2*k_2)+(2*k_3)+k_4)*dt;

    % Perform the nonsingular MRP propagation attitude check
    s = norm(sys_state(1:3,i+1));
    if s > 1
        sys_state(1:3,i+1) = -(sys_state(1:3,i+1) ./(s^2));
    end
end
toc

% plot the attitude
figure;
subplot(3,1,1);
plot(t,sys_state(1,:)); hold on;
plot(t,sigma_ref(1,:)); hold off
title('\sigma_1 vs \sigma_{r1}')
legend('\sigma_1','\sigma_{r1}');
subplot(3,1,2);
plot(t,sys_state(2,:)); hold on;
plot(t,sigma_ref(2,:)); hold off
title('\sigma_2 vs \sigma_{r2}')
legend('\sigma_2','\sigma_{r2}');
subplot(3,1,3);
plot(t,sys_state(3,:)); hold on;
plot(t,sigma_ref(3,:)); hold off
title('\sigma_3 vs \sigma_{r3}')
legend('\sigma_3','\sigma_{r3}');

% plot the angular rates
figure;
subplot(3,1,1);
plot(t,sys_state(4,:)); hold on;
plot(t,omega_ref(1,:)); hold off
title('\omega_1 vs \omega_{r1}')
legend('\omega_1','\omega_{r1}');
subplot(3,1,2);
plot(t,sys_state(5,:)); hold on;
plot(t,omega_ref(2,:)); hold off
title('\omega_2 vs \omega_{r2}')
legend('\omega_2','\omega_{r2}');
subplot(3,1,3);
plot(t,sys_state(6,:)); hold on;
plot(t,omega_ref(3,:)); hold off
title('\omega_3 vs \omega_{r3}')
legend('\omega_3','\omega_{r3}');


% plot sigmaBR and omegaBR
figure;
subplot(3,1,1);
plot(t,sigma_err(1,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
```

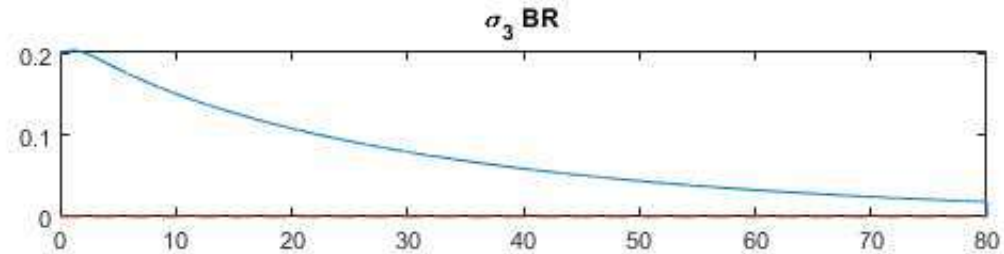
```
title('\sigma_1 BR')
subplot(3,1,2);
plot(t,sigma_err(2,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\sigma_2 BR')
subplot(3,1,3);
plot(t,sigma_err(3,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\sigma_3 BR')


% plot sigmaBR and omegaBR
figure;
subplot(3,1,1);
plot(t,omega_err(1,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_1 BR')
subplot(3,1,2);
plot(t,omega_err(2,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_2 BR')
subplot(3,1,3);
plot(t,omega_err(3,:)); hold on;
plot(t,zeros(1,npoints),'--'); hold off
title('\omega_3 BR')

% plot us and ug
figure;
plot(t,vecnorm(us)); hold on;
plot(t,vecnorm(ug)); hold off;
title('Control inputs Us and Ug vs time w/ NM');
legend('us', 'ug');

% plot the rotor speeds
 figure;
 plot(t,sys_state(10+p.nd:end,:)); hold off;
title('Rotor speeds vs time w/ NM');
legend('\Omega_1', '\Omega_2','\Omega_3');

 % plot the singularity index
figure;
plot(t,kappa); hold off;
title('Singularity index vs time with null motion');
```

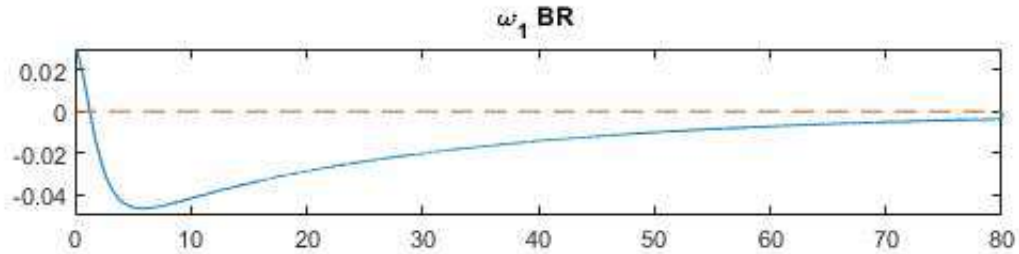
```
Elapsed time is 3.299008 seconds.
```

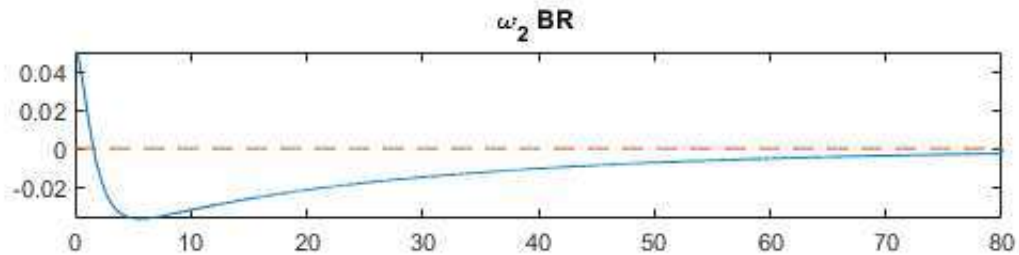

σ1 vs σr1
σ1
σr1
σ2 vs σr2
σ2
σr2
σ3 vs σr3
σ3
σr3
ω1 vs ωr1
ω1
ωr1
ω2 vs ωr2
ω2
ωr2
ω3 vs ωr3
ω3
ωr3

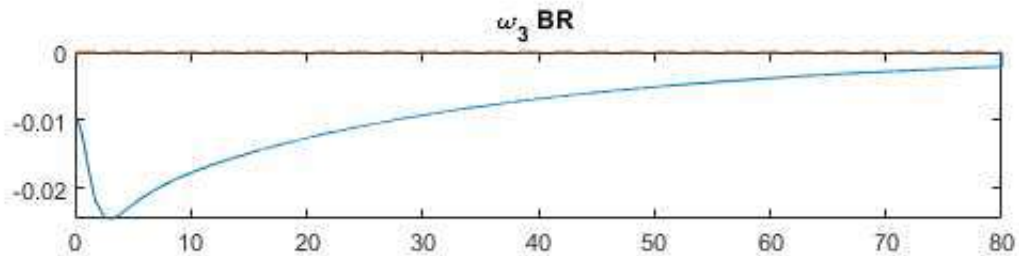

σ_1 BR
0.5
0
0
10
20
30
40
50
60
70
80

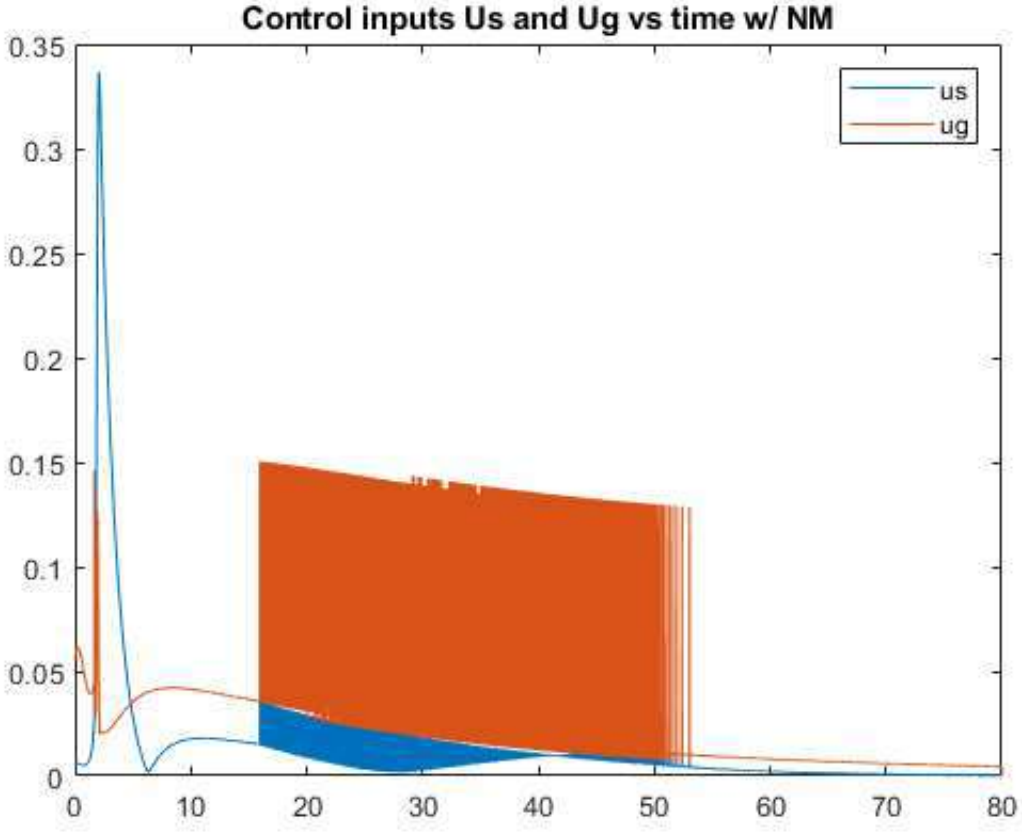

σ_2 BR
0.2
0
0
10
20
30
40
50
60
70
80

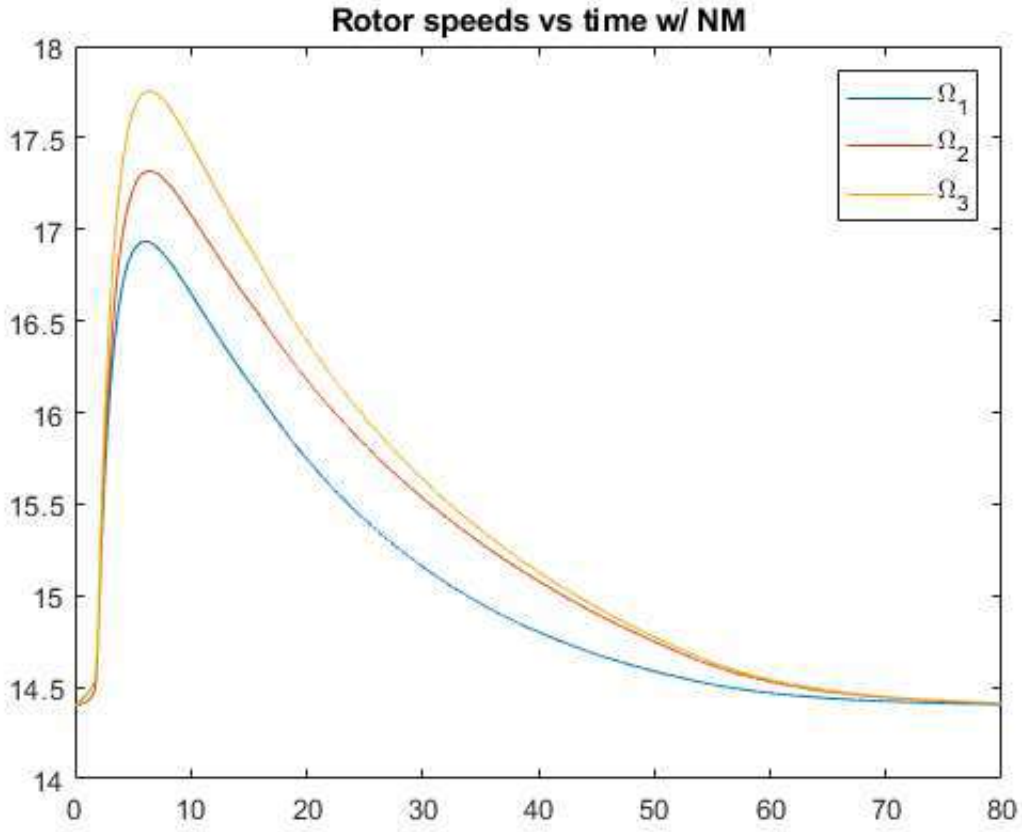

σ_3 BR
0.2
0.1
0
0
10
20
30
40
50
60
70
80

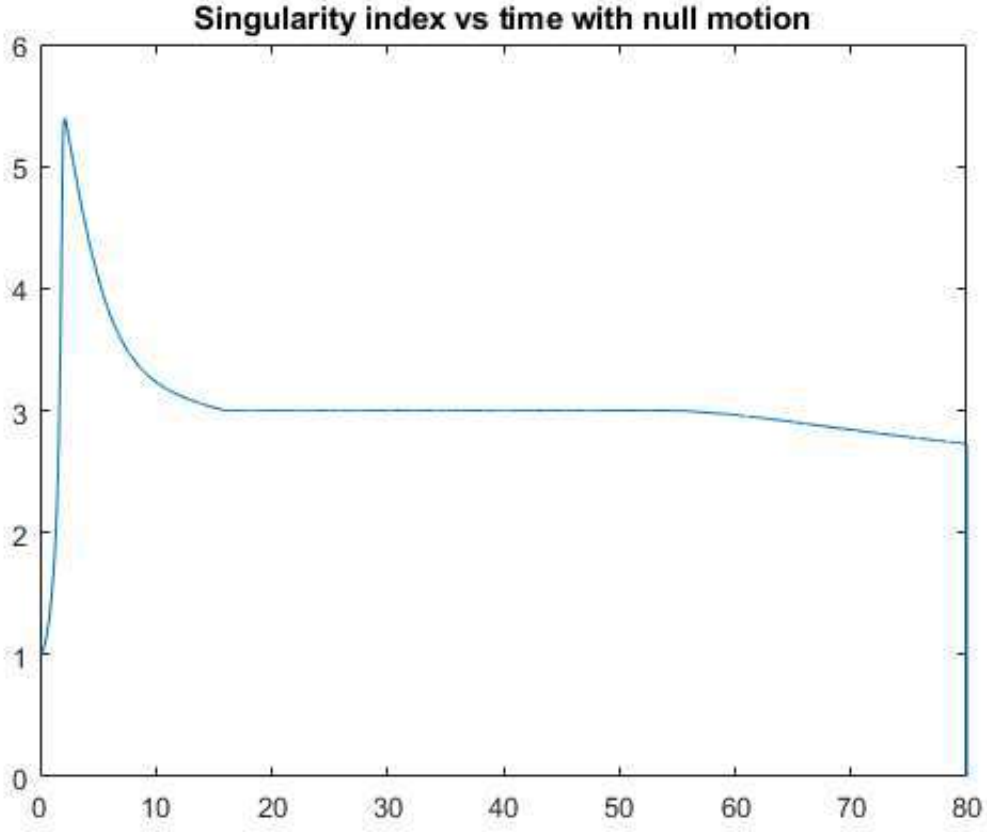

ω_1 BR
0.02
0
-0.02
-0.04
0
10
20
30
40
50
60
70
80


ω_2 BR
0.04
0.02
0
-0.02
0
10
20
30
40
50
60
70
80


ω_3 BR
0
-0.01
-0.02
0
10
20
30
40
50
60
70
80


Control inputs Us and Ug vs time w/ NM
us
ug
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
70
80


Rotor speeds vs time w/ NM
Ω_1
Ω_2
Ω_3
18
17.5
17
16.5
16
15.5
15
14.5
14
0
10
20
30
40
50
60
70
80



Published with MATLAB® R2018a

ASEN6010 Homework 2

University of Colorado
Boulder

Padraig Lysandrou

October 23, 2018

Question 5

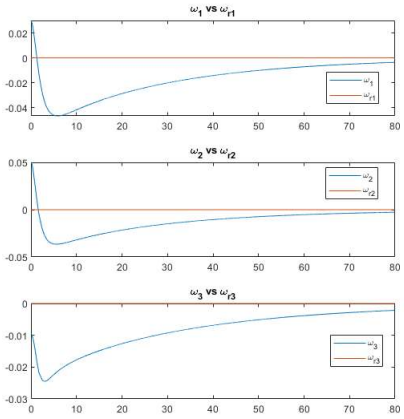
Discuss both how the null motion is implemented, and numerically simulate and test this null motion control strategy:

For this problem, I implemented the singular value decomposition method described on page 501 of the fourth edition of Analytical Mechanics of Space Systems. The D1 matrix can be decomposed in the regulation problem and studied for a useful parameter κ, or the condition number of the matrix.

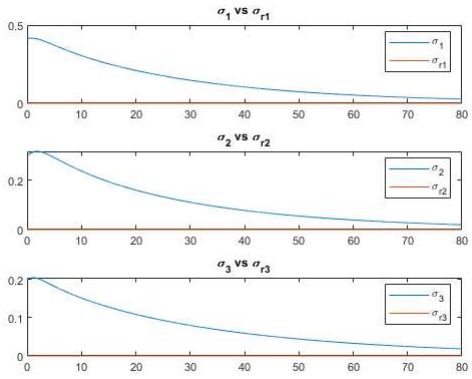
The initial null motion strategy was former as such: $\dot{\eta} = \dot{\eta}_c + \dot{\eta}_n$ where $\dot{\eta}_c$ is described inially on page 488 as the function $\dot{\eta}_c = WQ^T(QWQ^T)^{-1}(-L_r)$. The W matrices are weighting matrices associated with how active each of the reaction wheel/CMG modes are. Once L_r is computed, this is simple to find knowing that $Q = [D_0 \quad D]$. The second part of this sum is defined as $\dot{\eta}_n = k_e(\hat{W}Q^T(Q\hat{W}Q^T)^{-1}Q - I_{2N\times 2N})A\left(\frac{\Delta\Omega}{\Delta\gamma}\right)$, where $\Delta\Omega = \Omega - \Omega_f$ and similarly $\Delta\gamma = \gamma - \gamma_f$ where the subscript-f variables describe the preferred quantities. Their use cases include wanting to traverse to an attitude but remain at a nominal wheel speed Ω_f afterwards. Without this tool, the new steady state wheel speeds will be different from the nominal initial condition, which could possibly be annoying. The harder part to calculate is $\Delta\gamma$. Here we employ singular value decomposition.

I first generate the Q matrix and A matrix (which is I_{2NN} because I care about following both preferred wheel rates and gimbal angles). I then generate the $\hat{W}$ matrix which is also identity for the time being. I then take the SVD using MATLAB command $[U, S, V] = svd(D)$;. This nicely sorts the values and their companion vectors from maximum to minimum. Therefore, $\sigma_1 = S(1,1)$ and similarly $\sigma_3 = S(3,3)$. Our singularity condition number is $\kappa = \frac{\sigma_1}{\sigma_3}$. We can then generate the χ vectrix as $(-(Iws. * rtr_spd + Js. * ws).'. * Gs) + ((Js. * wt).'. * Gt)$. Note that this method of element-wise multiplication, transpose, and then additional element wise multiplication is invariant to N, and therefore is expandable to the implementation of N VSCMG actuators. You will see this throughout my code. Once this is completed, I perform the $\frac{\partial\sigma_1}{\partial\gamma_i} \quad \forall i$ and perform the similar operation for σ_3. Let us note that $\frac{\partial\sigma_j}{\partial\gamma_i} = (u_j^T\chi_i)V(i,j)$ where each u is a column vector the SVD pre-multiplication U matrix. We can then calculate $\frac{\partial\kappa}{\partial\gamma_i} = \frac{1}{\sigma_3}\frac{\partial\sigma_1}{\partial\gamma_i} - \frac{\sigma_1}{\sigma_3^2}\frac{\partial\sigma_3}{\partial\gamma_i}$. Finally we can calculate $\Delta\gamma = -\alpha(1-\kappa(t))\frac{\partial\kappa}{\partial\gamma}$ where α is zero if $\kappa(t) < \kappa_{db}$. Intuitively, we do not want to employ the use of null motion control if the singularity condition is not large (meaning not close to a singularity), otherwise we will use the help of this control.

I have numerically simulated this and have included some plots below for your consideration.

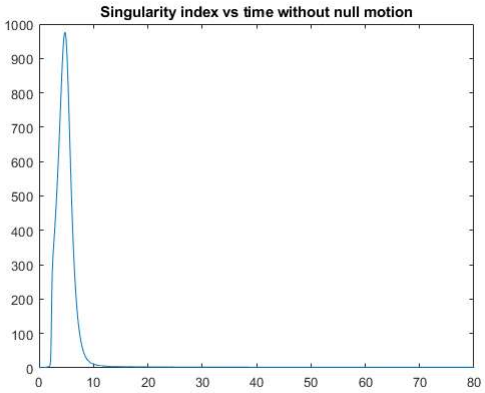


(a) The angular rate $\omega_{B/N}$ with and without null motion control (they're the same)

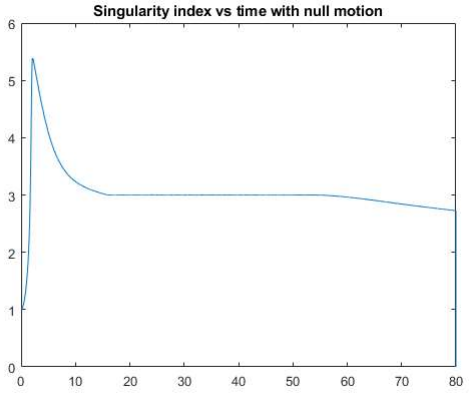


(b) The MRP attitude with and without null motion control (they're the same)

Figure 1



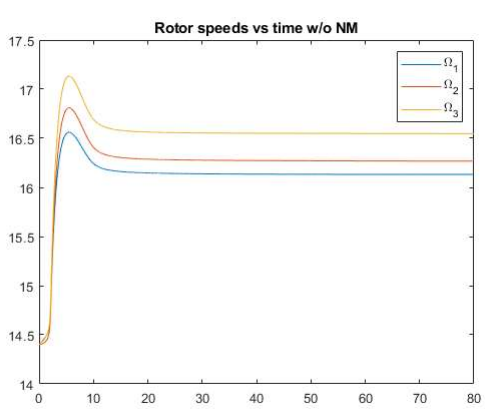
(a) κ without null motion.



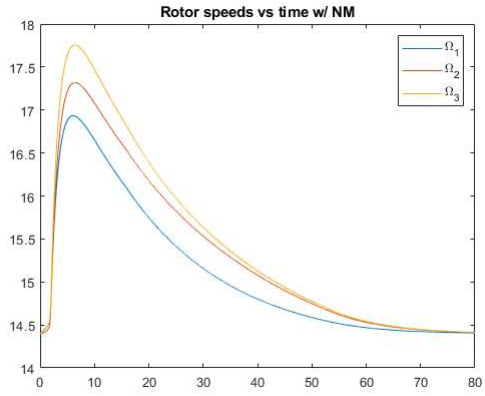
(b) κ with null motion. Notice the curve is brought back to my κ_{DB} of 3.

Figure 2

The initial conditions and other parameters can be found in the attached MATLAB publish document.

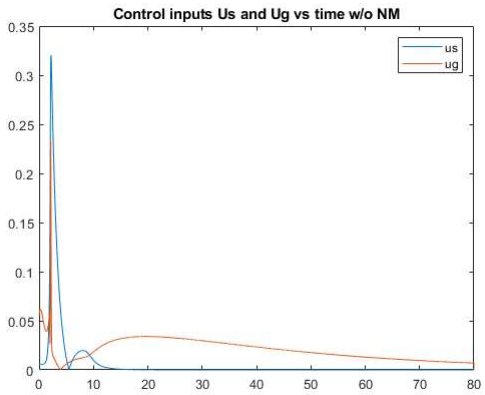


(a) Ω without null motion.

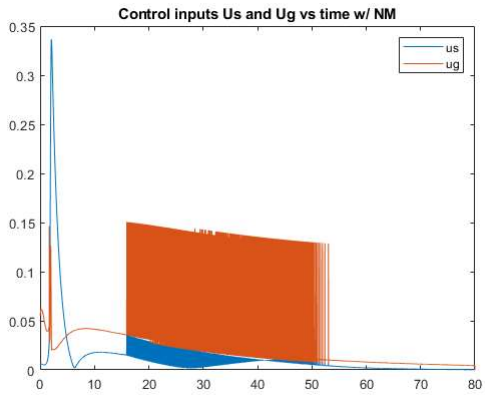


(b) Ω with null motion. Notice it sticks to my preferred Ω_0 at steady state.

Figure 3



(a) Norm of inputs w/o null motin.



(b) Norm of inputs without null motin.

Figure 4: In this scenario, it seems that the null motion input values oscillate quite a but towards the center of the timeline.