Problem 1

```
function f = ffunctgravgradsc02(t,x,IMoIbody,norbit)
%
%  Copyright (c) 2019 Mark L. Psiaki.  All rights reserved.
%
%  This function implements a nonlinear dynamic model
%  of a rigid-body spacecraft that includes the
%  gravity-gradient torques produced by a 1/r^2
%  gravity field for a spacecraft that is flying
%  in a circular orbit.  This particular model
%  uses quaternion to parameterize the transformation
%  from local-level orbit following coordinates
%  to the body-fixed coordinates in which
%  the moment-of-inertia matrix IMoIbody is
%  defined so that the transformation from
%  local-level coordinates to body-fixed coordinates
%  is defined by the orthonormal rotation matrix:
%
%    R = R(q)
%
%
%  Inputs:
%
%    t                      The time, in seconds, at which f is
%                           to be computed.
%
%    x                      = [q1;q2;q3;q4;omegabody1;omegabody2;
%                           omegabody3], the 7-by-1 state
%                           vector of this system.  The first four
%                           elements give the non-dimensional unit-
%                           normalized attitude quaternion for the
%                           rotation from local-level coordinates
%                           to spacecraft body coordinates.  The last
%                           three elements give the body rotation rate
%                           vector relative to inertial coordinates
%                           in radians/second and expressed in
%                           the same body-fixed axes in which
%                           IMoIbody is defined.
%
%    IMoIbody               The 3-by-3 symmetric spacecraft
%                           moment-of-inertia matrix about the
%                           spacecraft center of mass and given
%                           in body-fixed coordinates in units
%                           of kg-m^2.
%
%    norbit                 The orbital motion rate in
%                           radians/sec.  This is known as
%                           the mean motion in Keplerian
%                           orbital dynamics parlance.
%
%  Outputs:
```

```
%
%    f                      =
 [q1dot;q2dot;q3dot;q4dot;omegabody1dot;...
%                          omegabody2dot;omegabody3dot],
%                          the 7-by-1 vector that contains the
%                          computed time derivatives of the state
%                          from the kinematics and dynamics models
%                          of the spacecraft.  f(1:4,1) is given
%                          in 1/second units.  f(5:7,1) is given in
%                          radians/second^2.
%

%
%  Determine the rotation matrix from local-level coordinates
%  to body-fixed coordinates.
%
   q = x(1:4);
   qnorm = q*(1/sqrt(sum(q.^2)));
   R = rotmatquaternion(qnorm);
%
%  Determine the rotation rate of the body-fixed coordinates
%  relative to local-level coordinates and given along
%  body axes.
%
   omegavec = x(5:7);
   deltaomegavec = omegavec - R*[0;-norbit;0];
%
%  Determine the quaternion time rate of change from
%  the quaternion kinematics model.
%
   Omegamat = [0 deltaomegavec(3) -deltaomegavec(2)
 deltaomegavec(1);...
               -deltaomegavec(3) 0 deltaomegavec(1)
 deltaomegavec(2);...
               deltaomegavec(2) -deltaomegavec(1) 0
 deltaomegavec(3);...
               -deltaomegavec(1) -deltaomegavec(2) -deltaomegavec(3)
 0];
   qdot = 0.5*Omegamat*q;
%
%  Compute the unit direction vector from the Earth to
%  the spacecraft and given in spacecraft body coordinates:
%
   rhatcmvec = R*[0 0 -1]';
%
%  Compute the gravity-gradient torque in body coordinates.
%
   IMoI_rhatcmvec = IMoIbody*rhatcmvec;
   Tgravgradvec = 3*(norbit^2)*cross(rhatcmvec,IMoI_rhatcmvec);
%
%  Compute the angular velocity rate using Euler's equation.
%
   hvec = IMoIbody*omegavec;
   omegavecdot = IMoIbody\(cross(-omegavec,hvec)+Tgravgradvec);
```

```
%
%  Assemble the computed state time derivative elements
%  into the output vector.
%
   f = [qdot;omegavecdot];
```

Published with MATLAB® R2019a

Problem 2

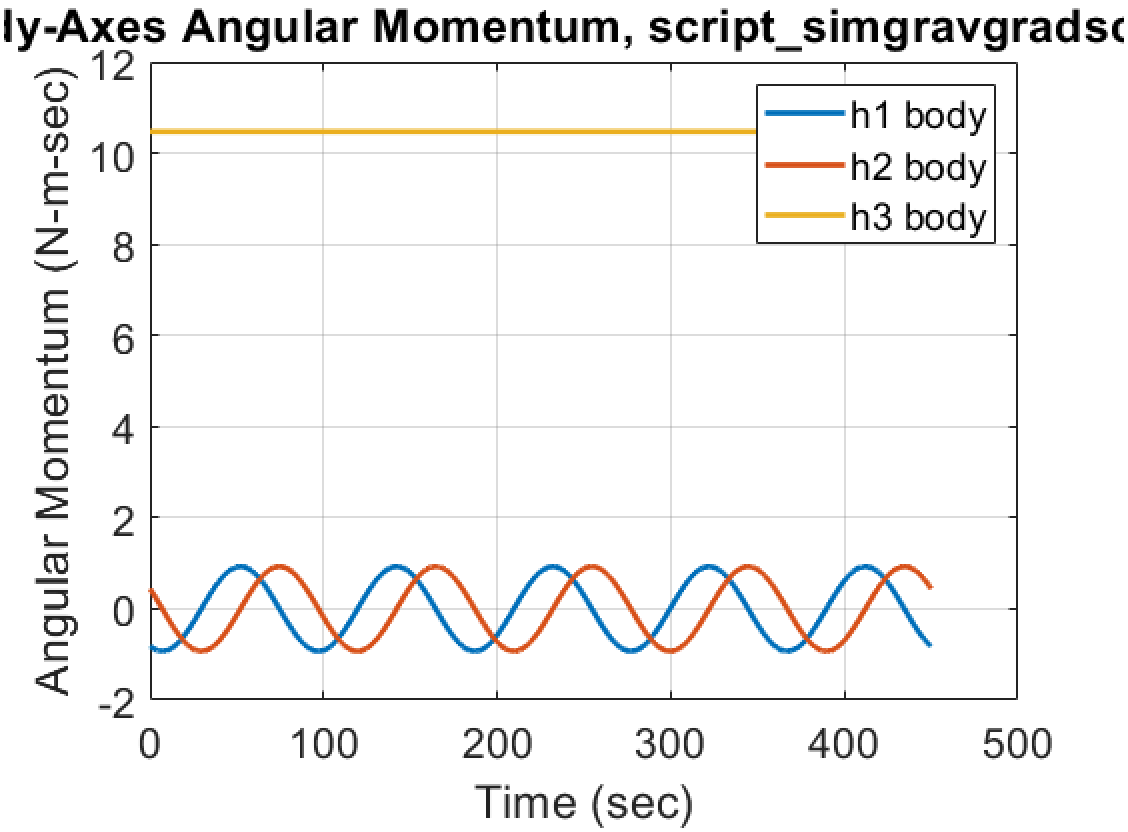
```
%script_simgravgradsc11.m
%
%  Copyright (c) 2019 Mark L. Psiaki.  All rights reserved.
%
%  This Matlab script simulates the torque-free motion of
%  an axi-symmetric spinning satellite.
%
%  This script makes a plot of the
%  angular momentum time history in body-fixed
%  coordinates and in inertial
%  coordinates.  It also makes plots
%  of the time histories of the 3 body-axis spin-
%  rate vector elements.
%
%  Clear the Matlab workspace.
%
   clear;clc;close all;
%
%  Set up the simulation parameters.
%
   Itr = 60;
   Ispin = 100;
   IMoIbody = diag([Itr;Itr;Ispin]);
   omegabody0 = [(-0.13*(2*pi/60));(0.07*(2*pi/60));(2*pi/60)];
   norbit = 0; % eliminate gravity-gradient torque and
               % rotation of the reference frame relative to which
               % x(1:4,1) defines the attitude quaternion so that
               % it becomes an inertial reference frame rather
               % than a non-inertial orbit-following local-level
               % reference frame.
   q0 = [0;0;0;1];
   x0 = [q0;omegabody0];
%
%  Define the aircraft dynamics function handle
%  in a form that is suitable for input to ode45.m.
%
   ffunctode45 = @(tdum,xdum) ...
             ffunctgravgradsc02(tdum,xdum,IMoIbody,norbit);
%
%  Define the time span of the simulation, computing outputs
%  every 0.5.  This time span should be large enough
%  to see several spins periods and several nutation periods.
%
   tspan = ((0:900)')*0.5;
%
%  Set up numerical integration options for ode45.m
%  in a way that uses a tighter relative tolerance than
%  is normally used.
%
   optionsode45 = odeset('RelTol',1.e-10);
```

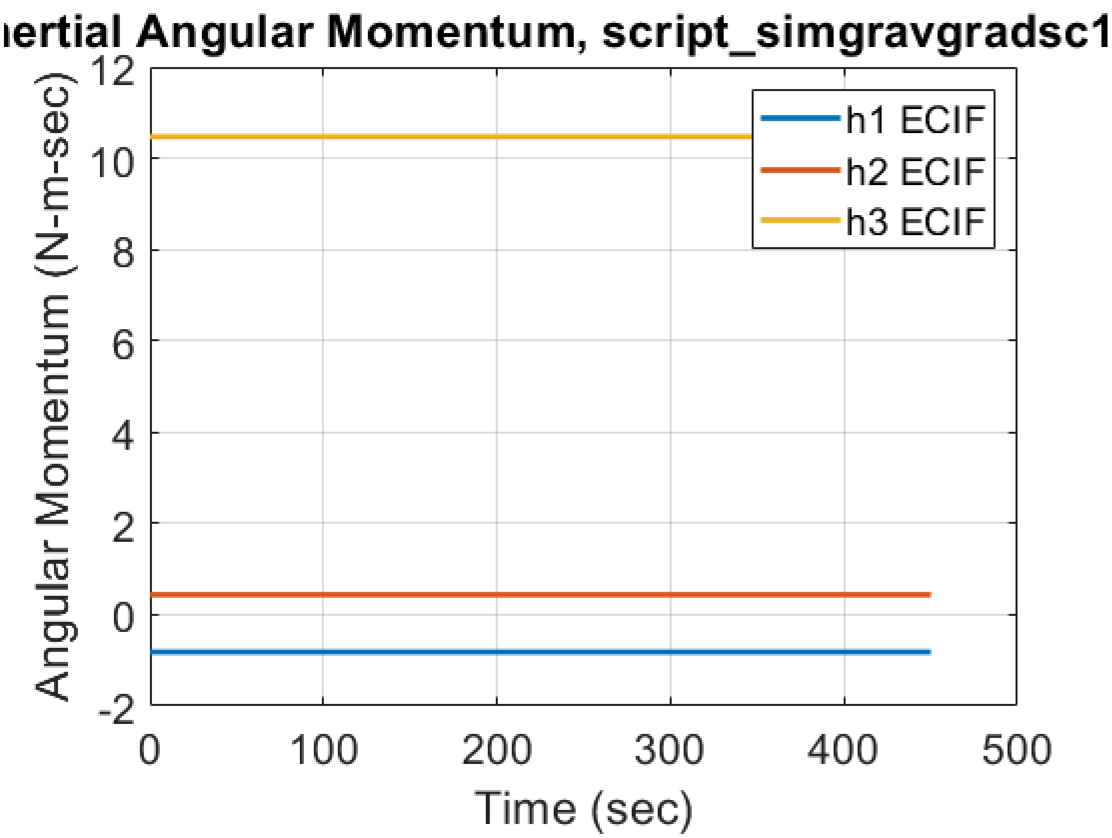
```
%
%  Call ode45.m in order to perform numerical integration.
%
   tic
   [thist,xhist] = ode45(ffunctode45,tspan,x0,optionsode45);
   timetosim = toc;
%
%  Compute the angular momentum vector time history in
%  inertial coordinates.
%
   tic
   N = size(thist,1);
   hvecbodyhist = zeros(N,3);
   hvecinertialhist = zeros(N,3);
   for k = 1:N
      xk = xhist(k,:)';
      hvecbodyk = IMoIbody*xk(5:7,1);
      hvecbodyhist(k,:) = hvecbodyk';
      qk = xk(1:4,1);
      qknorm = qk*(1/sqrt(sum(qk.^2)));
      Rk = rotmatquaternion(qknorm);
      hvecinertialk = (Rk')*hvecbodyk;
      hvecinertialhist(k,:) = hvecinertialk';
   end
   timetohvecinertial = toc
   clear k xk hvecbodyk qk qknorm Rk hvecinertialk
%
%  Compute the nutation frequency.
%
   omeganut = abs(Ispin-Itr)*omegabody0(3)/Itr;

%
%  Compute the theoretical body-axis spin vector component
%  time histories that are valid for this axially-symmetric
%  spacecraft.
%
   signItrminusIspin = sign(Itr - Ispin);
   omegabody1hist = omegabody0(1)*cos(omeganut*thist)
 +omegabody0(2)*signItrminusIspin*sin(omeganut*thist);
   omegabody2hist = omegabody0(2)*cos(omeganut*thist) -
omegabody0(1)*signItrminusIspin*sin(omeganut*thist);
   omegabody3hist = ones(N,1)*omegabody0(3);
%
%  Plot the body-axes angular momentum time history.
%
   figure(1)
   hold off
   plot(thist,hvecbodyhist,'LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Momentum (N-m-sec)')
   title(['Body-Axes Angular Momentum,',...
          ' script\_simgravgradsc11.m'])
```

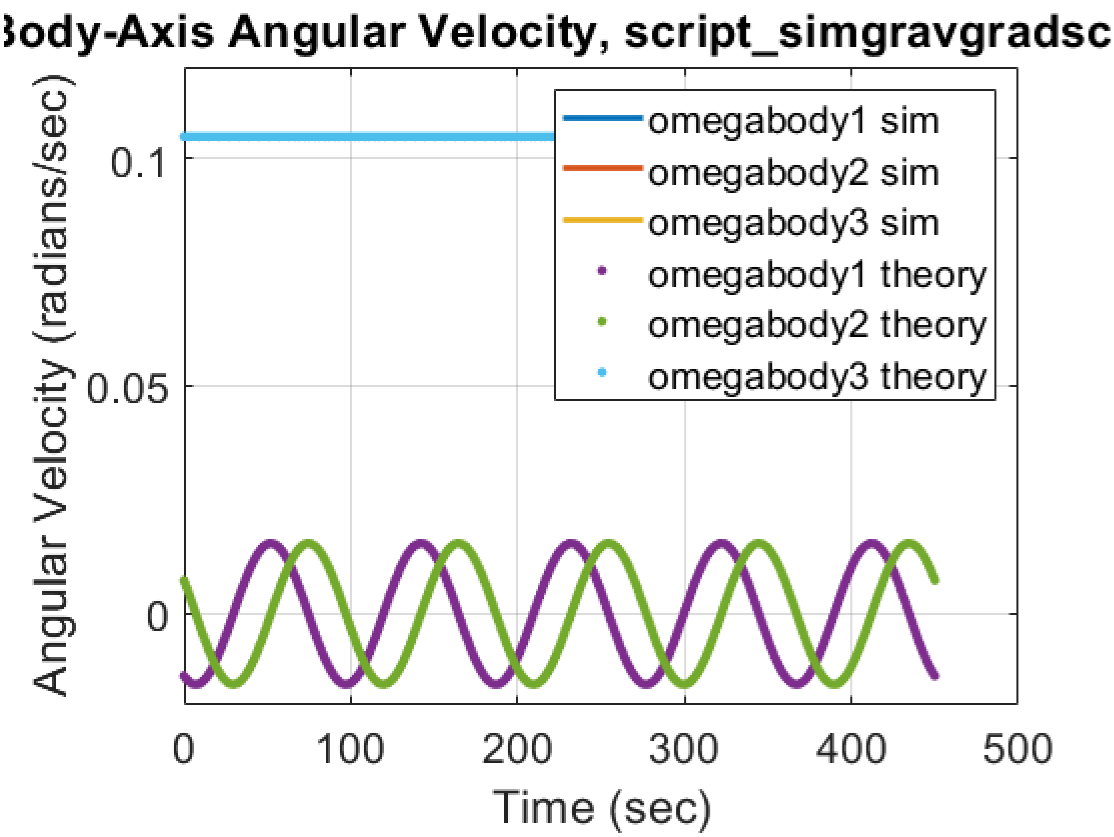
```
   legend('h1 body','h2 body','h3 body')
%
%  Plot the inertial angular momentum time history.
%
   figure(2)
   hold off
   plot(thist,hvecinertialhist,'LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Momentum (N-m-sec)')
   title(['Inertial Angular Momentum,',...
          ' script\_simgravgradsc11.m'])
   legend('h1 ECIF','h2 ECIF','h3 ECIF')
%
%  Plot the body-axis spin vector component time histories,
%  both from the numerical integration and the theoretical
%  values.
%
   figure(3)
   hold off
   plot(thist,xhist(:,5:7),'-','LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   hold on
   plot(thist,[omegabody1hist,omegabody2hist,omegabody3hist],'.',...
        'MarkerSize',10)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Velocity (radians/sec)')
   title(['Body-Axis Angular Velocity,',...
          ' script\_simgravgradsc11.m'])
   legend('omegabody1 sim','omegabody2 sim','omegabody3 sim',...
          'omegabody1 theory','omegabody2 theory','omegabody3 theory')
%
%  Save the results.
%
   textcommands = ['These data have been generated by the',...
                   ' commands in script_simgravgradsc11.m'];
   save simgravgradsc11

   format long
   xfinal = xhist(end,:)'
   qfinalmag = norm(xfinal(1:4,1))

   disp('To verify that angular mommentum is conserved, we subtract
 the initial value of intertial angular mommentum from the rest of
 its time history and compute the norm. The closer this is to zero the
 better the conservation of angular momentum is. This would imply zero
 external torque')
   disp('The norm on variation of inertial angular momentum is:')
   disp(norm(hvecinertialhist(:,1)-
hvecinertialhist(1,1))+norm(hvecinertialhist(:,2)-
hvecinertialhist(1,2))+norm(hvecinertialhist(:,3)-
hvecinertialhist(1,3)))
```

```
  disp('The norm of the difference in the time histories between
 theory and simulated angular velocities is:')
  disp(norm(omegabody1hist-xhist(:,5))+norm(omegabody2hist-
xhist(:,6))+norm(omegabody3hist-xhist(:,7)))
```

```
timetohvecinertial =

   0.010582900000000


xfinal =

   0.076778204795434
  -0.041341864928188
  -0.984339512015986
   0.153186692608215
  -0.013613444068028
   0.007330204591270
   0.104719755119660


qfinalmag =

   0.999997240113923

To verify that angular mommentum is conserved, we subtract the initial
 value of intertial angular mommentum from the rest of its time
 history and compute the norm. The closer this is to zero the better
 the conservation of angular momentum is. This would imply zero
 external torque
The norm on variation of inertial angular momentum is:
     1.320999550676519e-04

The norm of the difference in the time histories between theory and
 simulated angular velocities is:
     5.332291859654702e-06
```


ly-Axes Angular Momentum, script_simgravgradsc
Angular Momentum (N-m-sec)
Time (sec)
h1 body
h2 body
h3 body
12
10
8
6
4
2
0
-2
0
100
200
300
400
500


iertial Angular Momentum, script_simgravgradsc1
Angular Momentum (N-m-sec)
Time (sec)
h1 ECIF
h2 ECIF
h3 ECIF
12
10
8
6
4
2
0
-2
0
100
200
300
400
500



Published with MATLAB® R2019a

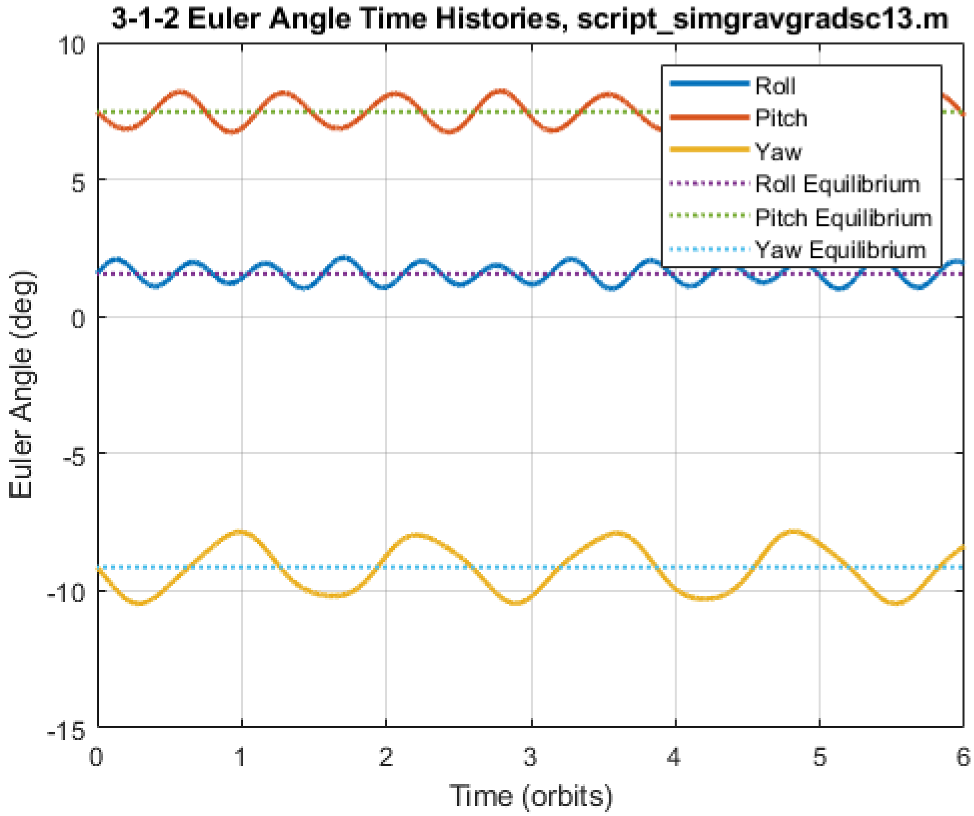
Problem 3

```
%script_simgravgradsc13.m
%
%  Copyright (c) 2019 Mark L. Psiaki.  All rights reserved.
%
%  This Matlab script simulates the motion of a
%  nadir-pointing satellite that is acting
%  under the influence of a gravity-gradient torque
%  in a circular Low-Earth Orbit (LEO) that has an
%  orbital period of 6000 sec.  The initial conditions
%  start with zero roll, pitch, and yaw angles relative
%  to the principal axes, but with non-zero initial roll,
%  pitch, and yaw rates relative to the princpal axes.
%
%  The principal axes are not exactly body axes, which is
%  why the point about which the satellite attitude
%  oscillates is not exactly zero for the roll, pitch, and
%  yaw angles.
%
%  This script also makes plots of the roll, pitch, and
%  yaw attitude time history.
%
%  Clear the Matlab workspace.
%
   clear;clc;close all;
%
%  Load simulation parameters IMoIbody, norbit, omegabody0
%  and q0 from the file simgravgradsc13_data.mat.
%
   load simgravgradsc13_data
%
%  Set up the initial state.  This initial state vector
%  has been chosen with a knowledge of the principal
%  axes coordinates, and q0 has been chosen so that
%  the principal axes are exactly aligned with
%  the local-level orbit-following coordinate
%  system so that, had omegabody0 also been chosen
%  correctly, q(t) and omegabody(t) would have
%  remainded constant.  A slightly perturbed
%  omegabody0 has been chosen to produce motion.
%  This allows the stability of the gravity-gradient
%  system to be tested.
%
   x0 = [q0;omegabody0];
%
%  Define the aircraft dynamics function handle
%  in a form that is suitable for input to ode45.m.
%
   ffunctode45 = @(tdum,xdum) ...
             ffunctgravgradsc02(tdum,xdum,IMoIbody,norbit);
%
```

```
%  Define the time span of the simulation, computing outputs
%  1000 times per orbit for 6 orbits.
%
   Torbit = 2*pi/norbit;
   tspan = (0:6000)'*(Torbit/1000);
%
%  Set up numerical integration options for ode45.m
%  in a way that uses a tighter relative tolerance than
%  is normally used.
%
   optionsode45 = odeset('RelTol',1.e-10);
%
%  Call ode45.m in order to perform numerical integration.
%
   tic
   [thist,xhist] = ode45(ffunctode45,tspan,x0,optionsode45);
   timetosim = toc;
%
%  Determine the 3-1-2 Euler angle time histories.  The function
%  yawpitchrollcalc02.m has been designed with the 3-1-2
%  assumption specifically in mind.  Note that the
%  computation of qknorm via normalization is included
%  in order to remove any small errors in the quaternion
%  unit normalization that may have built up due
%  to numerical intergration error of the quaternion
%  kinematics.
%
   N = size(thist,1);
   phihist = zeros(N,1);
   thetahist = zeros(N,1);
   psihist = zeros(N,1);
   for k = 1:N
      qk = xhist(k,1:4)';
      qknorm = qk*(1/sqrt(sum(qk.^2)));
      [phik,thetak,psik] = yawpitchrollcalc02(qknorm);
      phihist(k,1) = phik;
      thetahist(k,1) = thetak;
      psihist(k,1) = psik;
   end
   clear k qk qknorm phik thetak psik
%
%  Plot the roll, pitch, and yaw time histories.
%  An analysis of this case indicates that there should
%  be 8.05 pitch angle oscillations, 4.61 oscillations
%  of one of the coupled roll-yaw modes, and 11.4
%  oscillations of the other coupled roll-yaw modes.
%  Of course, the actual roll, pitch, and yaw
%  principal axes differ slightly from the
%  body axes used in this simulation.  Therefore,
%  the roll, pitch, and yaw angle time histories plotted
%  below are not pure principal axes quantities.
%  This fact causes slight additional coupling
%  between modes beyond the theoretical coupling of the
%  two roll/yaw modes of oscillation.
```

```
%
   figure(1)
   hold off
   plot(thist*(1/Torbit),[phihist,thetahist,psihist]*(180/pi),...
        'LineWidth',2)
   hold on
   plot(thist*(1/Torbit),ones(N,1)*...
        ([phihist(1,1),thetahist(1,1),psihist(1,1)]*(180/pi)),...
        ':','LineWidth',1.5)
   hold off
   set(get(gcf,'CurrentAxes'),'FontSize',10)
   grid
   xlabel('Time (orbits)')
   ylabel('Euler Angle (deg)')
   title(['3-1-2 Euler Angle Time Histories,',...
          ' script\_simgravgradsc13.m'])
   legend('Roll','Pitch','Yaw',...
          'Roll Equilibrium','Pitch Equilibrium',...
          'Yaw Equilibrium')
%
%  Save the results.
%
   textcommands = ['These data have been generated by the',...
                   ' commands in script_simgravgradsc13.m'];
   save simgravgradsc13
   format long
   xfinal = xhist(end,:)'
   qfinalmag = norm(xfinal(1:4,1))

   disp('The system produces a neutrally stable response, evident
 from the time histories showing non incresing crest and troughs in
 attitude over 6 orbits')
```

```
xfinal =

   0.021414146457538
   0.062360917783887
  -0.071803040606404
   0.995237366114879
   0.000136192378780
  -0.001052654303743
   0.000062483571553


qfinalmag =

   1.000000270643444

The system produces a neutrally stable response, evident from the time
 histories showing non incresing crest and troughs in attitude over 6
 orbits
```



Published with MATLAB® R2019a

Q4) The Euler's Eq

$$\frac{d\vec{h}}{dt} = \vec{T} \quad \text{where } \vec{h} \rightarrow \text{Angular momentum}$$
$$\vec{T} \rightarrow \text{External torque}$$

for $\vec{h}^b$, Angular momentum Expressed in a body frame Rotating as $\vec{\omega}^b$, the Euler Eq is as follows

$$\frac{d\vec{h}^b}{dt} + \vec{\omega}^b \times \vec{h}^b = \vec{T}^b$$

for a Torque-Free case $\vec{T}^b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Expanding $\vec{h}^b = I^b_{MOI} \vec{\omega}^b$

where I^b_{MOI} is a principle Frame of the body

$$I^b_{MOI} = \begin{bmatrix} I_{tr1} & 0 & 0 \\ 0 & I_{tr2} & 0 \\ 0 & 0 & I_{spin} \end{bmatrix}$$

So, the torque-free Rigid body Equations become:

$$\underbrace{\begin{bmatrix} I_{tr1} & 0 & 0 \\ 0 & I_{tr2} & 0 \\ 0 & 0 & I_{spin} \end{bmatrix}}_{I^b_{MOI}} \underbrace{\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix}}_{\dot{\vec{\omega}}^b} + \underbrace{\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}}_{\vec{\omega}^b} \times \left(\underbrace{\begin{bmatrix} I_{tr1} & 0 & 0 \\ 0 & I_{tr2} & 0 \\ 0 & 0 & I_{spin} \end{bmatrix}}_{I^b_{MOI}} \underbrace{\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}}_{\vec{\omega}^b} \right) = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\vec{T}=0}$$

further Simplifying,

$$\begin{bmatrix} I_{Tr1}\dot{\omega}_1 \\ I_{Tr2}\dot{\omega}_2 \\ I_{spin}\dot{\omega}_3 \end{bmatrix} + \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} I_{Tr1}\omega_1 \\ I_{Tr2}\omega_2 \\ I_{spin}\omega_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$I_{Tr1}\dot{\omega}_1 + \omega_2\omega_3(I_{spin} - I_{Tr2}) = 0$$

$$I_{Tr2}\dot{\omega}_2 + \omega_3\omega_1(I_{TR1} - I_{spin}) = 0$$

$$I_{spin}\dot{\omega}_3 + \omega_1\omega_2(I_{TR2} - I_{TR1}) = 0$$

$$\dot{\omega}_1 = \frac{(I_{TR2} - I_{spin})}{I_{Tr1}}\omega_2\omega_3$$

$$\dot{\omega}_2 = \frac{(I_{spin} - I_{TR1})}{I_{TR2}}\omega_1\omega_3$$

$$\dot{\omega}_3 = \frac{(I_{TR1} - I_{TR2})}{I_{spin}}\omega_1\omega_2$$

If ω_1 & ω_2 are very small, then $\omega_1\omega_2 << 0$

$\therefore \dot{\omega}_3 = 0$, so $\omega_3(t) = \text{const} = \omega_{3,avg}$

We substitute $\omega_3(t) = \omega_{3avg}$ in $\dot{\omega}_1$ & $\dot{\omega}_2$ Eq to get

$$\dot{\omega}_1 = \left(\frac{I_{TR2} - I_{spin}}{I_{TR1}}\right) \omega_{3avg}\, \omega_2$$

diffrentiation once w.r.t ~~to~~ time

$$\ddot{\omega}_1 = \left(\frac{I_{TR2} - I_{spin}}{I_{TR1}}\right) \omega_{3avg}\, \dot{\omega}_2$$

Substituting the Eq for $\dot{\omega}_2 = \left(\frac{I_{TR1} - I_{spin}}{I_{TR2}}\right) \omega_{3avg}\, \omega_1$

we get

$$\ddot{\omega}_1 = \frac{(I_{TR2} - I_{spin})(I_{TR1} - I_{spin})}{I_{TR1}\, I_{TR2}}\, \omega_{3avg}^2\, \omega_1$$

give the 2nd order ODE as follows.

$$\ddot{\omega}_1 + \underbrace{\frac{(I_{TR2} - I_{spin})(I_{TR1} - I_{spin})}{I_{TR1}\, I_{TR2}}\, \omega_{3avg}^2}_{\omega_{nut}^2}\, \omega_1 = 0$$

$\omega_{nut}^2 \rightarrow$ the nutation frequency

$$\boxed{\therefore\ \omega_{nut} = \sqrt{\frac{(I_{spin} - I_{TR1})(I_{spin} - I_{TR2})}{I_{TR1}\, I_{TR2}}}\ \omega_{3avg}\ \blacksquare}$$

the assumption here is $I_{spin} > I_{TR1}$ & $I_{spin} > I_{TR2}$
(or) $I_{spin} < I_{TR1}$ & $I_{spin} < I_{TR2}$

The general solutions to the 2^{nd} order PDEs are

$$\omega_1(t) = A \cos(\omega_{nut} t) + B \sin(\omega_{nut} t)$$

$$\omega_2(t) = C \cos(\omega_{nut} t) + D \sin(\omega_{nut} t)$$

$$\dot{\omega}_1(t) = -A\,\omega_{nut} \sin(\omega_{nut} t) + B\,\omega_{nut} \cos(\omega_{nut} t)$$

using the initial dynamics Eq

$$-A\,\omega_{nut} \sin(\omega_{nut} t) + B\,\omega_{nut} \cos(\omega_{nut} t)$$

$$= \frac{(I_{TR2} - I_{spin})}{I_{TR1}}\,\omega_{3avg}\,\omega_2$$

$$= \left(\frac{I_{TR2} - I_{spin}}{I_{TR1}}\right)\omega_{3avg}\left[C\cos(\omega_{nut}) + D\sin(\omega_{nut} t)\right]$$

Equating coeff of $\sin(\omega_{nut} t)$ & $\cos(\omega_{nut} t)$, as this relation need to hold for all t, we get

$$B\,\omega_{nut} = C\,\frac{(I_{TR2} - I_{spin})}{I_{TR1}}\,\omega_{3avg}$$

$$-A\,\omega_{nut} = D\,\frac{(I_{TR2} - I_{spin})}{I_{TR1}}\,\omega_{3avg}$$

$$B\sqrt{\frac{(I_{sp}-I_{TR1})(I_{sp}-I_{TR2})}{I_{TR1}I_{TR2}}} = C\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{(|I_{TR2}-I_{spin}|)^2}{I_{TR1}^2}}$$

$$-A\sqrt{\frac{(I_{sp}-I_{TR1})(I_{sp}-I_{TR2})}{I_{TR1}I_{TR2}}} = D\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{(|I_{TR2}-I_{spin}|)^2}{I_{TR1}^2}}$$

$$B = C\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{I_{TR2}(I_{spin}-I_{TR2})}{I_{TR1}(I_{spin}-I_{TR1})}}$$

$$D = -A\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{I_{TR1}(I_{spin}-I_{TR1})}{I_{TR2}(I_{spin}-I_{TR2})}}$$

we realize that when $t=0$

$$\omega_1(0) = A\cos(0) = A \quad \& \quad \omega_2(0) = C\cos(0) = C$$

$$\therefore \quad A = \omega_1(0) \quad \& \quad C = \omega_2(0)$$

So the final dynamics' Solution look like

$$\boxed{\begin{aligned} \omega_1(t) &= \omega_1(0)\cos(\omega_{nut}t) + \omega_2(0)\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{I_{TR2}(I_{spin}-I_{TR2})}{I_{TR1}(I_{spin}-I_{TR1})}}\sin(\omega_{nut}t) \\ \omega_2(t) &= \omega_2(0)\cos(\omega_{nut}t) - \omega_1(0)\,\text{Sign}(I_{TR2}-I_{spin})\sqrt{\frac{I_{TR1}(I_{spin}-I_{TR1})}{I_{TR2}(I_{spin}-I_{TR2})}}\sin(\omega_{nut}t) \end{aligned}}$$

$\blacksquare$

Problem 5

```
%script_simgravgradsc12.m
%
%  Copyright (c) 2019 Mark L. Psiaki.  All rights reserved.
%
%  This Matlab script simulates the torque-free motion of
%  a non-axi-symmetric spinning satellite.
%
%  This script makes a plot of the
%  angular momentum time history in body-fixed
%  coordinates and in inertial
%  coordinates.  It also makes plots
%  of the time histories of the 3 body-axis spin-
%  rate vector elements.
%
%  Clear the Matlab workspace.
%
   clear;clc;close all;
%
%  Set up the simulation parameters.  Load the body-axes moment-of-
%  inertia matrix and the initial body-axes angular velocity
%  from simgravgradsc12_data.mat.
%
   load simgravgradsc12_data
%
%  Set up the orbital angular rate.
%
   norbit = 0; % eliminate gravity-gradient torque and
               % rotation of the reference frame relative to which
               % x(1:4,1) defines the attitude quaternion so that
               % it becomes an inertial reference frame rather
               % than a non-inertial orbit-following local-level
               % reference frame.
   q0 = [0;0;0;1];
   x0 = [q0;omegabody0];
%
%  Define the aircraft dynamics function handle
%  in a form that is suitable for input to ode45.m.
%
   ffunctode45 = @(tdum,xdum) ...
             ffunctgravgradsc02(tdum,xdum,IMoIbody,norbit);
%
%  Define the time span of the simulation, computing outputs
%  every 0.5.  This time span should be large enough
%  to see several spins periods and several nutation periods.
%
   tspan = ((0:900)')*0.5;
%
%  Set up numerical integration options for ode45.m
%  in a way that uses a tighter relative tolerance than
%  is normally used.
```

```
%
   optionsode45 = odeset('RelTol',1.e-10);
%
%  Call ode45.m in order to perform numerical integration.
%
   tic
   [thist,xhist] = ode45(ffunctode45,tspan,x0,optionsode45);
   timetosim = toc;
%
%  Compute the angular momentum vector time history in
%  inertial coordinates.
%
   tic
   N = size(thist,1);
   hvecbodyhist = zeros(N,3);
   hvecinertialhist = zeros(N,3);
   for k = 1:N
      xk = xhist(k,:)';
      hvecbodyk = IMoIbody*xk(5:7,1);
      hvecbodyhist(k,:) = hvecbodyk';
      qk = xk(1:4,1);
      qknorm = qk*(1/sqrt(sum(qk.^2)));
      Rk = rotmatquaternion(qknorm);
      hvecinertialk = (Rk')*hvecbodyk;
      hvecinertialhist(k,:) = hvecinertialk';
   end
   timetohvecinertial = toc
   clear k xk hvecbodyk qk qknorm Rk hvecinertialk
%
%  Transform to principal axes and assume that the
%  principal axis whose moment of inertia is the most
%  different from the other two is the spin-axis
%  inertia.  This is a nearly axially-symmetric
%  spacecraft whose principal axes do not
%  exactly align with the body axes in which
%  the simulation has been conducted.
%
%  This eigenvalue decomposition computes the 3-by-3
%  matrixed Roldnew and IMoIbodynew such that
%  Roldnew*IMoIbodynew*inv(Roldnew) = IMoIbody
%  with IMoIbodynew being a diagonal matrix.  The symmetry
%  of IMoIbody should ensure that inv(Roldnew) = Roldnew'
%  so that Roldnew*IMoIbodynew*(Roldnew') = IMoIbody.
%  Symmetry should also ensure that IMoIbody is truly
%  diagonalizable (Some matrices can only be put into a
%  form known as Jordan form that is not completely
%  diagonal if there are repeated eigenvalues.)
%
   [Roldnew,IMoIbodynew] = eig(IMoIbody);
%
%  Check that Roldnew is orthonormal.
%
   errdum = norm(Roldnew*(Roldnew') - eye(3));
   if errdum > 1.e-12
```

```
      disp('Warning in script_simgravgradsc12.m: IMoIbody')
      disp(' does not appear to have orthonormal eigenvectors.')
      disp(' maybe it is not exactly diagonal.')
      disp(' ')
   end
   clear errdum
%
%  Extract the eigenvalues and arrange them in ascending order.
%
   Iprsvec = diag(IMoIbodynew);
   [Iprsvec,idumsortvec] = sort(Iprsvec);
   Roldnew = Roldnew(:,idumsortvec);
   if det(Roldnew) < 0
      Roldnew(:,3) = - Roldnew(:,3);
   end
%
%  Check that the physical constraint on the maximum
%  eigenvalue is respected.
%
   if Iprsvec(3,1) > (Iprsvec(1,1) + Iprsvec(2,1))
      disp('Warning in script_simgravgradsc12.m: IMoIbody''s')
      disp(' largest eigenvalue is more than the sum of it''s')
      disp(' other two eigenvalues.')
      disp(' ')
   end
   if Iprsvec(1,1) <= 0
      disp('Warning in script_simgravgradsc12.m: IMoIbody')
      disp(' has one or more non-positive eigenvalues.')
      disp(' ')
   end
%
%  Transform the body-axes spin rate vector time history
%  into the principal axis coordinate system.
%  Note that omegabodynewk = (Roldnew')*omegabodyk,
%  as should be the case, where omegabodyk = xhist(k,5:7)'
%  and omegabodynewk = omegabodynewhist(k,:)';
%
   omegabodynewhist = xhist(:,5:7)*Roldnew;
%
%  Determine whether the spacecraft is spinning
%  primarily about its minor axis or about its
%  major axis.  Make the third axis be the
%  spin axis in either case.
%
   if mean(abs(omegabodynewhist(:,1))) > ...
              mean(abs(omegabodynewhist(:,3)))
      idumsortvec = [2;3;1];
      Iprsvec = Iprsvec(idumsortvec,1);
      Roldnew = Roldnew(:,idumsortvec);
      omegabodynewhist = omegabodynewhist(:,idumsortvec);
      clear idumsortvec
   end
%
%  Make change the sign definitions on the first
```

```
%  and third axes, if necessary, in order
%  to ensure that the spin is about the positive
%  third axis.
%
   if mean(omegabodynewhist(:,3)) < 0
      Roldnew(:,1) = - Roldnew(:,1);
      Roldnew(:,3) = - Roldnew(:,3);
      omegabodynewhist(:,1) = - omegabodynewhist(:,1);
      omegabodynewhist(:,3) = - omegabodynewhist(:,3);
   end
%
%  Check that the new diagonal moment-of-inertia matrix
%  and rhe rotation matrix agree with the original
%  moment-of-inertia matrix.
%
   IMoIbodynew = diag(Iprsvec);
   IMoIbody_re = Roldnew*IMoIbodynew*(Roldnew');
   errdum = norm(IMoIbody_re - IMoIbody)/norm(IMoIbody);
   if errdum > 1.e-12
      disp('Warning in script_simgravgradsc12.m: There is')
      disp(' an inaccuracy in the principal axes model')
      disp(' ')
   end
   clear errdum
%
%  Compute an approximate nutation frequency based
%  on an approximate model that assumes axial symmetry
%  even though this assumption is not quite right.
%  The calculation of the mean omegaspin should
%  average over an integer number of nutation periods.
%  The length of the simulation has been chosen to
%  make this likely.
%
   omegaspinavg = mean(omegabodynewhist(:,3));
   Itr1 = Iprsvec(1,1);
   Itr2 = Iprsvec(2,1);
   Ispin = Iprsvec(3,1);
   omeganut = omegaspinavg*sqrt( (Ispin - Itr1)*(Ispin - Itr2)/Itr2/
Itr1);
%
%  Compute the theoretical body-axis spin vector component
%  time histories that are valid for this axially-symmetric
%  spacecraft.  This analysis assumes that
%
%   omegabodynewapprox1(t) = A*cos(omeganut*t) + B*sin(omeganut*t)
%
%   omegabodynewapprox2(t) = C*cos(omeganut*t) + D*sin(omeganut*t)
%
%  The constant DoverA = D/A and the constant BoverC = B/C
%  prove helpful in writing these approximation theoretical
%  solutions in terms of the initial values omegabodynewhist(1,1)
%  and omegabodynewhist(1,2).
%
```

```
  DoverA = -sign(Itr2-Ispin)*sqrt(Itr1*(Ispin - Itr1)/Itr2/(Ispin -
Itr2));
  BoverC = - (1/DoverA);
  omegabodynewapprox1hist = omegabodynewhist(1,1)*cos(omeganut*thist)
+ omegabodynewhist(1,2)*BoverC*sin(omeganut*thist);
  omegabodynewapprox2hist = omegabodynewhist(1,2)*cos(omeganut*thist)
+ omegabodynewhist(1,1)*DoverA*sin(omeganut*thist);
  omegabodynewapprox3hist = omegaspinavg*ones(N,1);
  clear omegatRmaghist omegatRmagmean omegatrmagratio0 ...
        omeganewbody10approx omeganewbody20approx
%
%  Plot the body-axes angular momentum time history.
%
   figure(1)
   hold off
   plot(thist,hvecbodyhist,'LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',10)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Momentum (N-m-sec)')
   title(['Original Body-Axes Angular Momentum,',...
          ' script\_simgravgradsc12.m'])
   legend('h1 body','h2 body','h3 body')
%
%  Plot the inertial angular momentum time history.
%
   figure(2)
   hold off
   plot(thist,hvecinertialhist,'LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',10)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Momentum (N-m-sec)')
   title(['Inertial Angular Momentum,',...
          ' script\_simgravgradsc12.m'])
   legend('h1 ECIF','h2 ECIF','h3 ECIF')
%
%  Plot the body-axis spin vector component time histories,
%  both from the numerical integration and the theoretical
%  values.
%
   figure(3)
   hold off
   plot(thist,omegabodynewhist,'-','LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',8)
   hold on
   plot(thist,[omegabodynewapprox1hist,omegabodynewapprox2hist,...
               omegabodynewapprox3hist],'.','MarkerSize',10)
   hold off
   grid
   xlabel('Time (sec)')
   ylabel('Angular Velocity (radians/sec)')
   title(['Principal Axes Angular Velocity,',...
          ' script\_simgravgradsc12.m'])
```

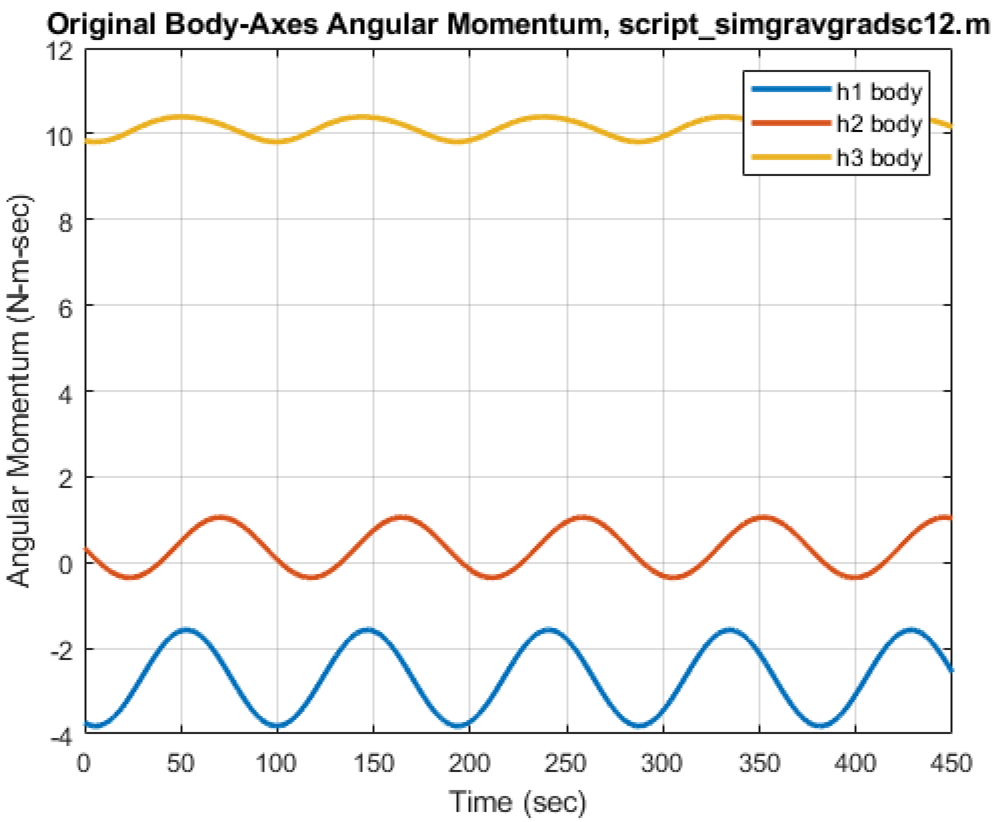
```
  legend('omegabodynew1 sim','omegabodynew2 sim','omegabodynew3
sim',...
         'omegabody1 approx. theory','omegabody2 approx. theory',...
         'omegabody3 approx. theory')
%
%  Plot the differencese between the nonlinear simulation
%  of the velocity vector components and the theoretical
%  models in order to focus in on the errors.
%
   figure(4)
   hold off
   omegaapproxerrhist = ...
             [omegabodynewapprox1hist,omegabodynewapprox2hist,...
                   omegabodynewapprox3hist] - omegabodynewhist;
   plot(thist,omegaapproxerrhist,'-','LineWidth',2)
   set(get(gcf,'CurrentAxes'),'FontSize',10)
   grid
   xlabel('Time (sec)')
   ylabel('Angular Velocity Approx. Errors (radians/sec)')
   title(['Principal-Axes Ang. Vel. Errors,',...
          ' script\_simgravgradsc12.m'])
   legend('omegabody1 approx.-true','omegabody2 approx.-true',...
          'omegabody3 approx.-true')
%
%  Save the results.
%
   textcommands = ['These data have been generated by the',...
                   ' commands in script_simgravgradsc12.m'];
   save simgravgradsc12
   format long
   xfinal = xhist(end,:)'

   disp('The following are the differences compared to
script_simgravgradsc11.m:')
   disp('1) The max amplitudes for angular velocities and angular
momentums are different between w1 and w2. This is because of the
difference I_tr1 and I_tr2');
   disp('2) The w3 is not constant in the ode45, true to the dynamics.
But the theoretical w3 is constant because of the assumption of small
w1 and w2');


timetohvecinertial =

   0.010446900000000


xfinal =

   0.300577539317991
  -0.074095730362301
  -0.935199318657734
   0.171928828717826
  -0.026762379581941
```

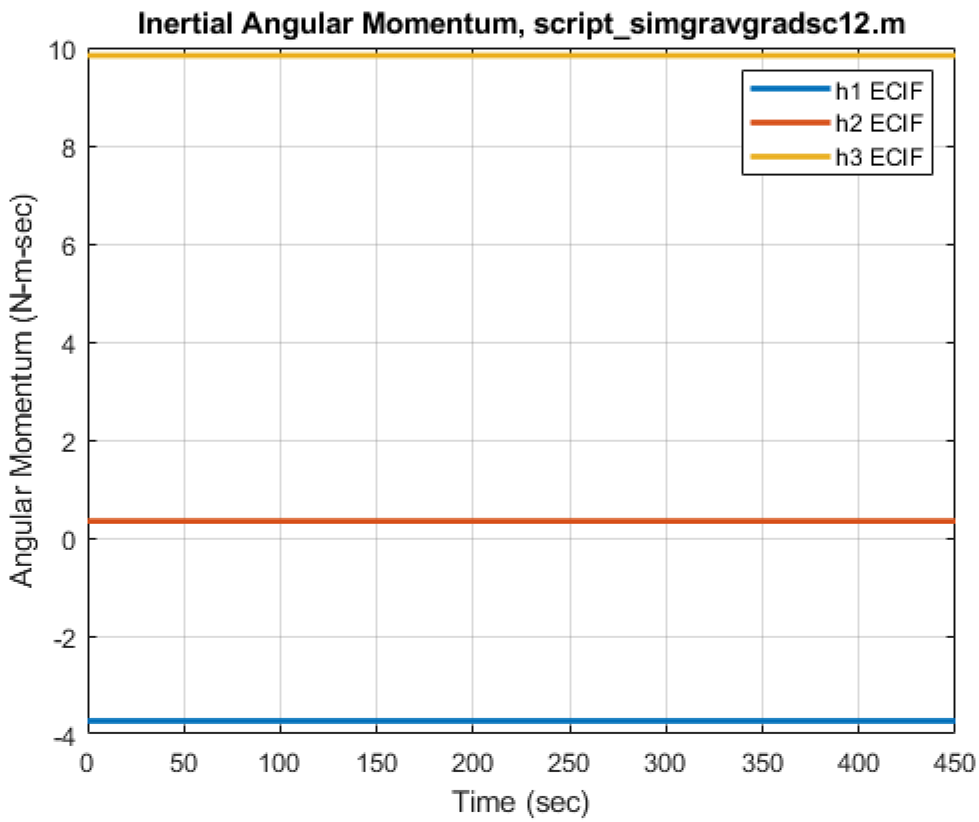
```
   0.017598485086061
   0.101028409577155
```

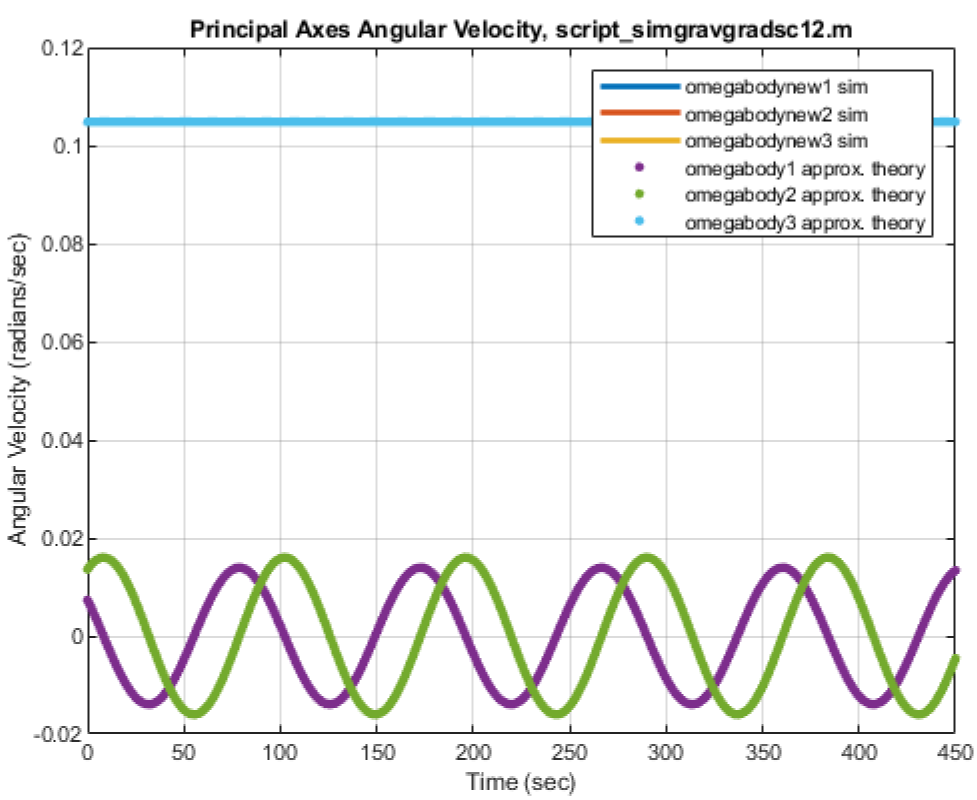
The following are the differences compared to script_simgravgradsc11.m:

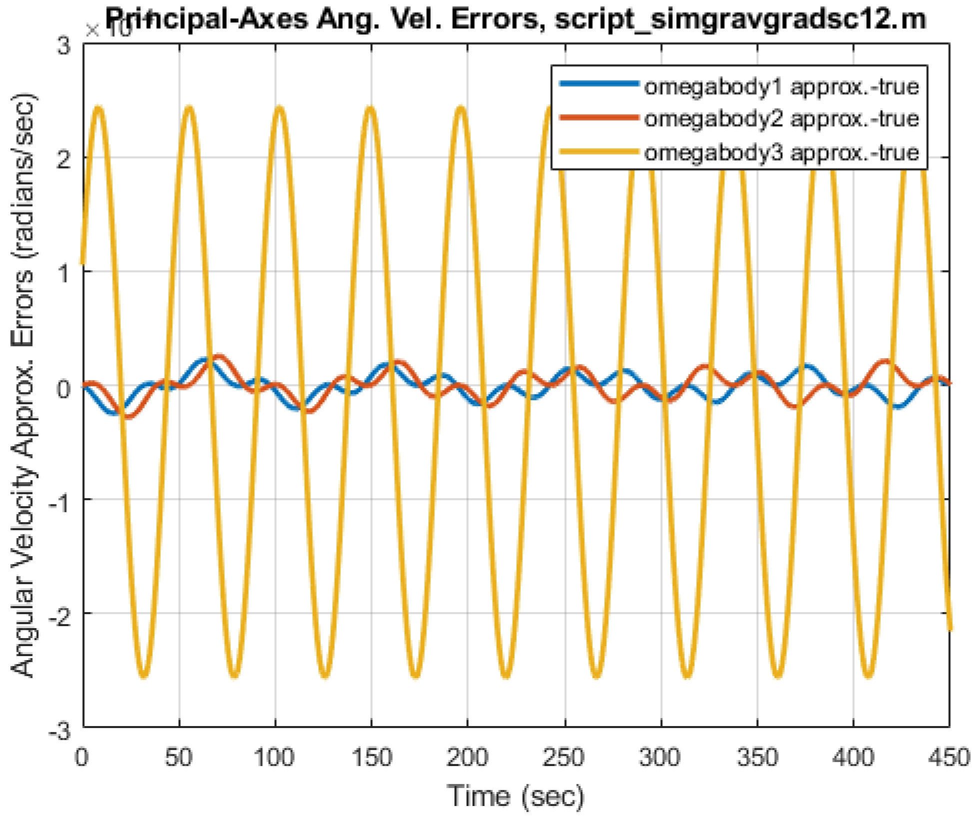
1) The max amplitudes for angular velocities and angular momentums are different between w1 and w2. This is because of the difference I_tr1 and I_tr2

2) The w3 is not constant in the ode45, true to the dynamics. But the theoretical w3 is constant because of the assumption of small w1 and w2




Inertial Angular Momentum, script_simgravgradsc12.m
Angular Momentum (N-m-sec)
Time (sec)
h1 ECIF
h2 ECIF
h3 ECIF
10
8
6
4
2
0
-2
-4
0
50
100
150
200
250
300
350
400
450

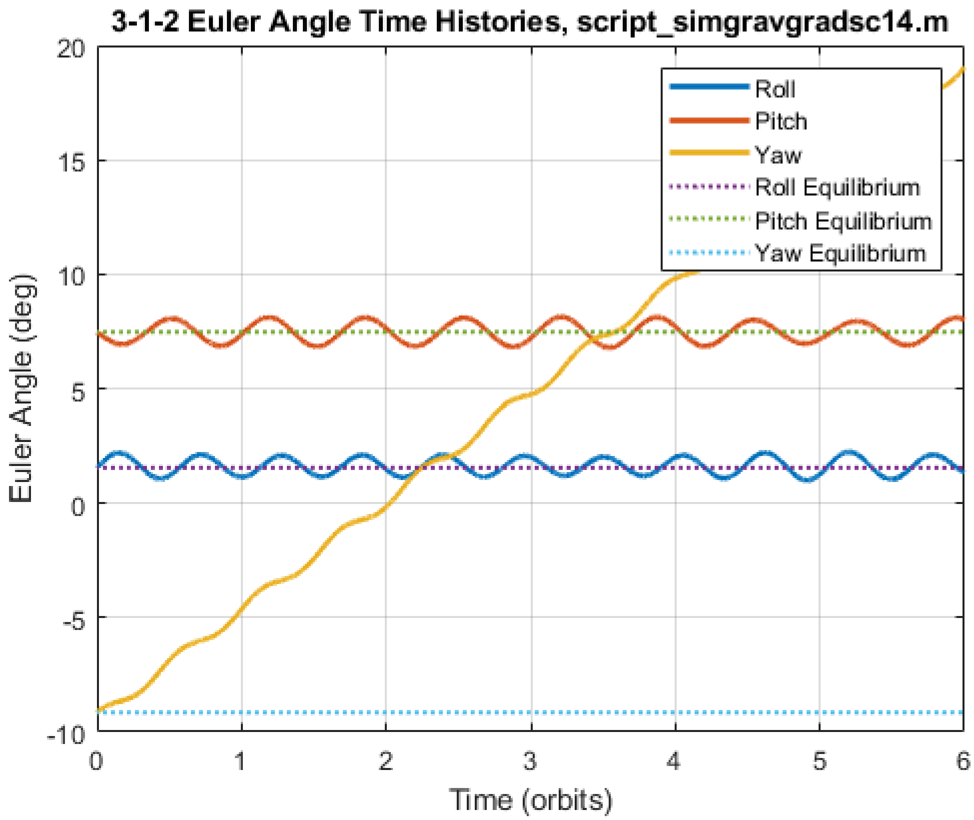

Principal Axes Angular Velocity, script_simgravgradsc12.m
Angular Velocity (radians/sec)
Time (sec)
omegabodynew1 sim
omegabodynew2 sim
omegabodynew3 sim
omegabody1 approx. theory
omegabody2 approx. theory
omegabody3 approx. theory
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
0
50
100
150
200
250
300
350
400
450



Published with MATLAB® R2019a

Problem 6

```
xfinal =

  -0.000024674605850
   0.070758815234848
   0.165798947726374
   0.983617801330580
  -0.000353720202840
  -0.000998579313148
  -0.000003710776134

The system is unstable, evident from diverging yaw angle
```

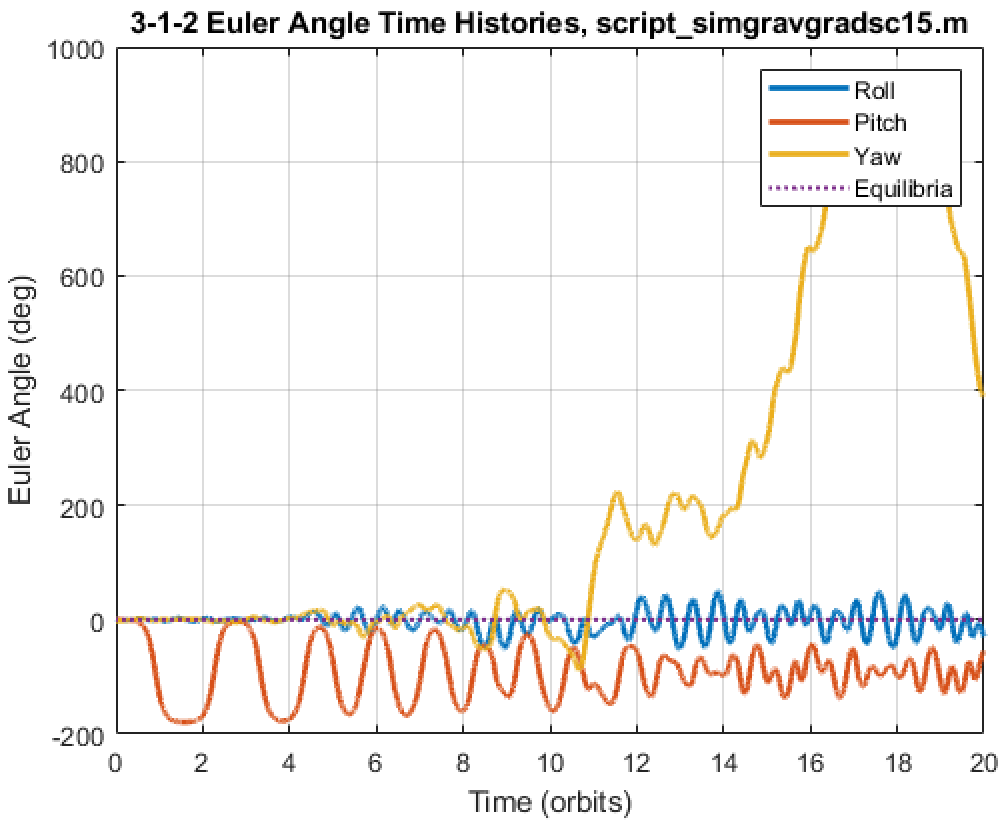


Published with MATLAB® R2019a

Problem 7

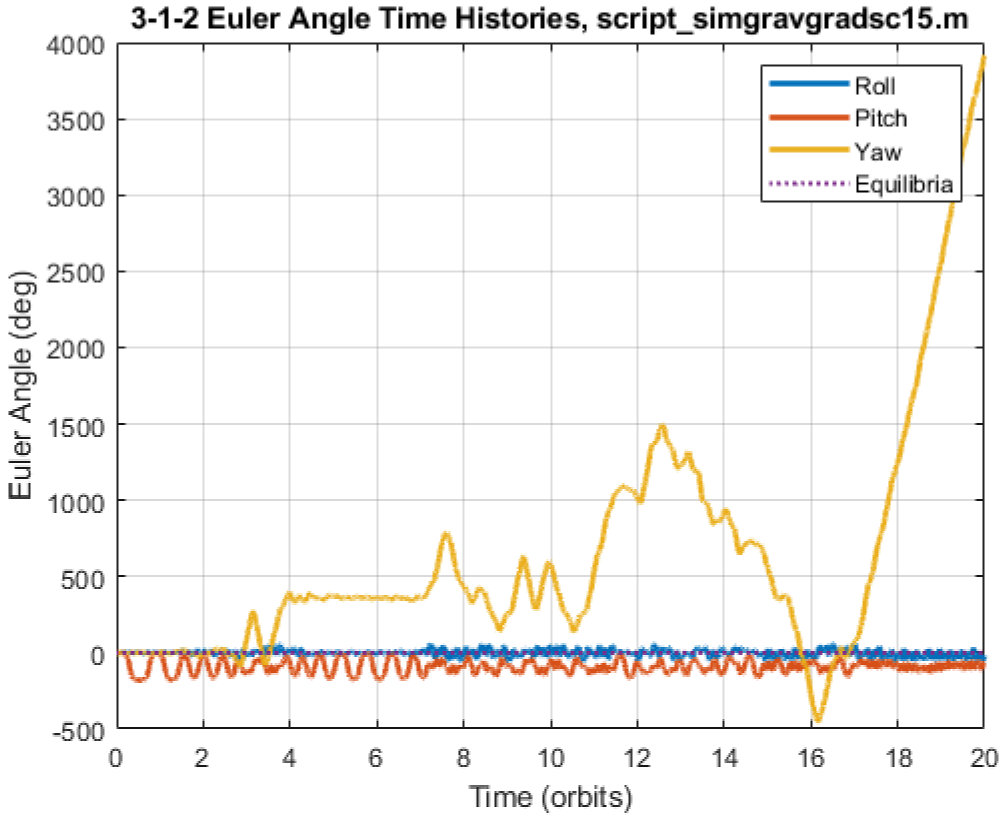
```
xfinal =

   0.117496010185310
   0.480952819413493
  -0.323683410192931
  -0.806297823411958
  -0.001205763144471
  -0.000264064182243
   0.000547611740791
```

The system is unstable, evident from diverging yaw angle. Though the time history shows behavior of marginally stable system, it reveals an chaotic behavior is larger time scales



Published with MATLAB® R2019a

Problem 7 (longer simulation)



Published with MATLAB® R2019a