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
```

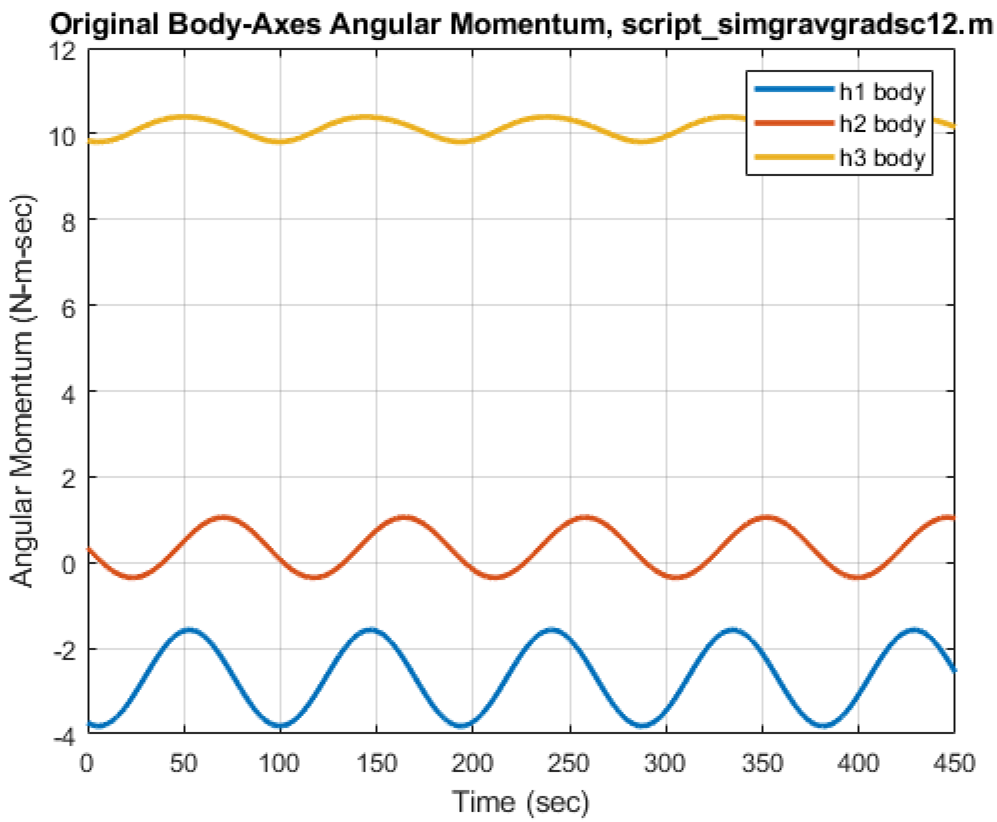
```
      disp('Warning in script_simgravgradsc12.m: IMoIbody')
      disp(' does not appear to have orthonormal eigenvectors.')
      disp(' maybe it is not exactly diagonal.')
      disp(' ')
   end
   clear errdum
%
%  Extract the eigenvalues and arrange them in ascending order.
%
   Iprsvec = diag(IMoIbodynew);
   [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
```

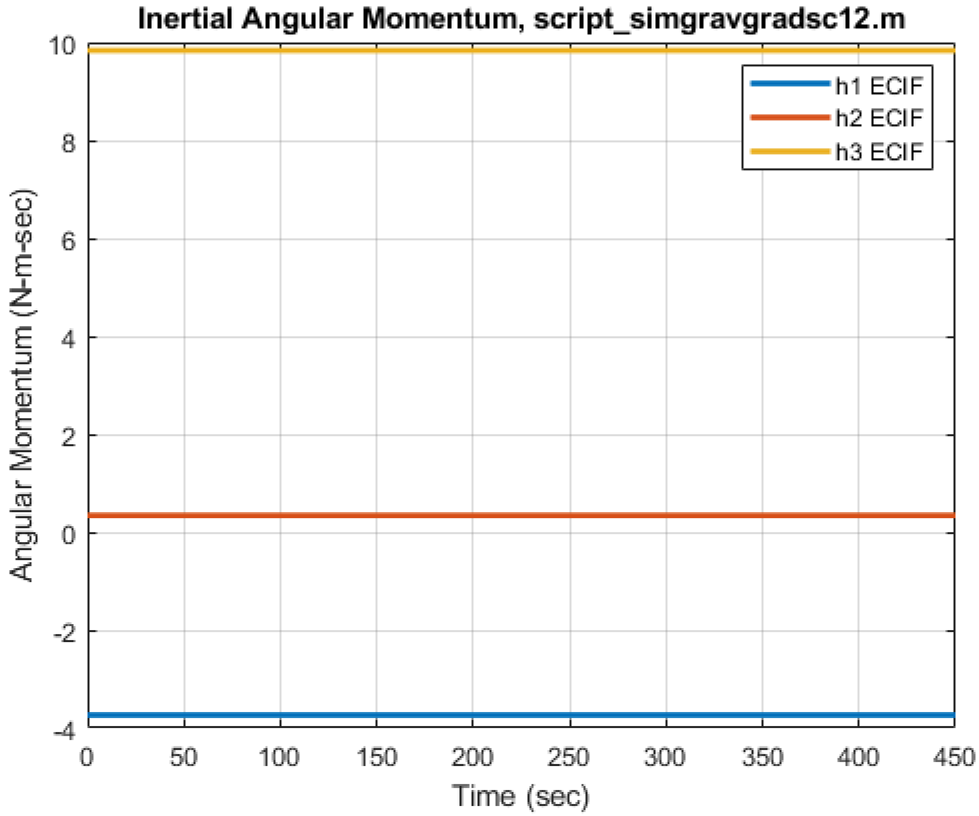
```
%  and third axes, if necessary, in order
%  to ensure that the spin is about the positive
%  third axis.
%
   if mean(omegabodynewhist(:,3)) < 0
      Roldnew(:,1) = - Roldnew(:,1);
      Roldnew(:,3) = - Roldnew(:,3);
      omegabodynewhist(:,1) = - omegabodynewhist(:,1);
      omegabodynewhist(:,3) = - 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).
%
```

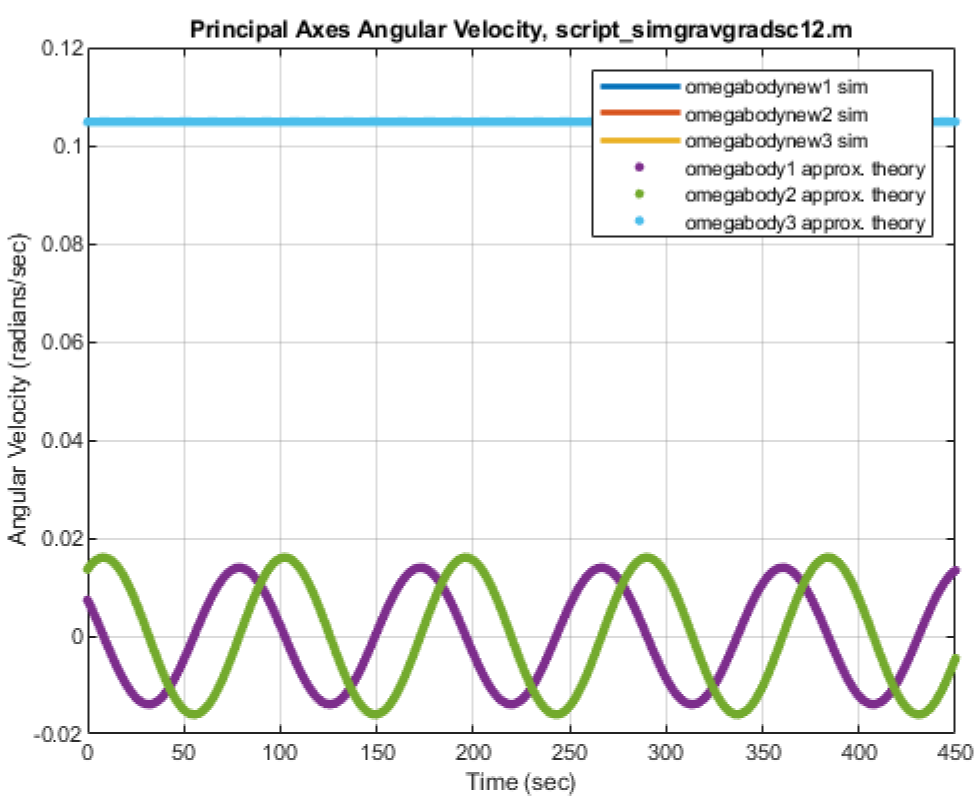
```
  DoverA = -sign(Itr2-Ispin)*sqrt(Itr1*(Ispin - Itr1)/Itr2/(Ispin -
Itr2));
  BoverC = - (1/DoverA);
  omegabodynewapprox1hist = omegabodynewhist(1,1)*cos(omeganut*thist)
+ omegabodynewhist(1,2)*BoverC*sin(omeganut*thist);
  omegabodynewapprox2hist = omegabodynewhist(1,2)*cos(omeganut*thist)
+ omegabodynewhist(1,1)*DoverA*sin(omeganut*thist);
  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'])
```

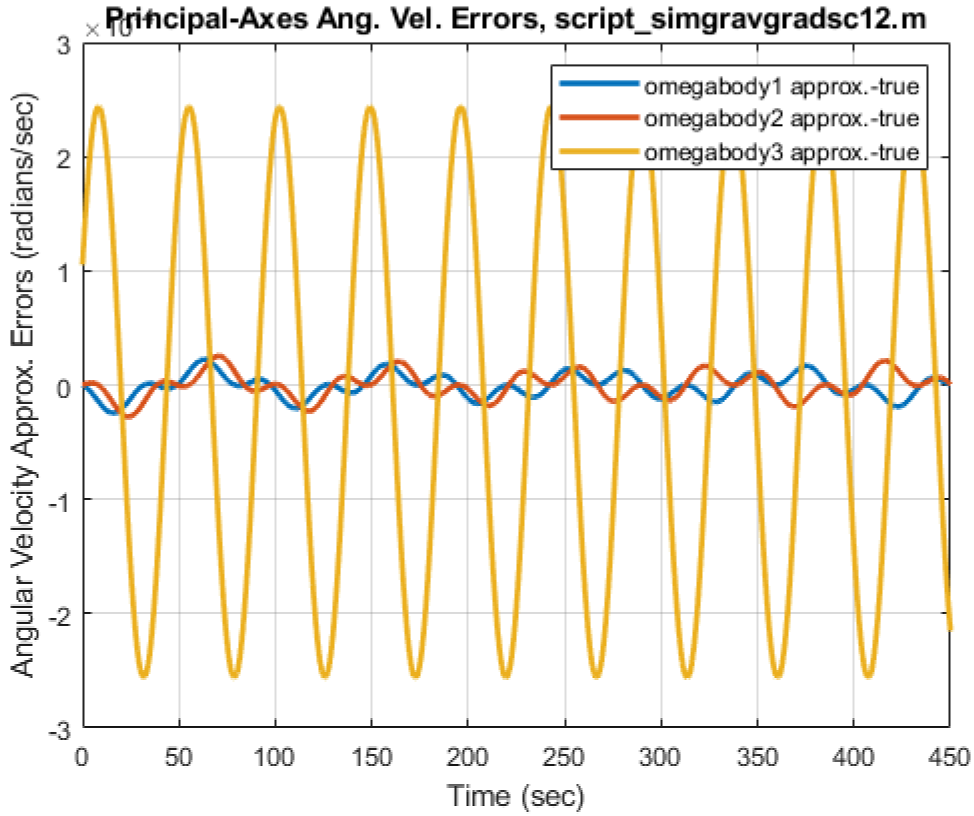
```
   legend('omegabodynew1 sim','omegabodynew2 sim','omegabodynew3
 sim',...
          'omegabody1 approx. theory','omegabody2 approx. theory',...
          'omegabody3 approx. theory')
%
%  Plot the differencese between the nonlinear simulation
%  of the velocity vector components and the theoretical
%  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
```

```
   0.0175984850860610
   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


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



Published with MATLAB® R2019a