Controlling Spacecraft Attitude with Reaction Wheels

Xiaozheng Xu

Dynamics

Olin College of Engineering

December 14, 2015

Abstract

1 Background

This project simulates how reaction wheels control the attitude of satellites and other spacecrafts. Using electric motors, reaction wheels apply a torque to a spacecraft through changing the wheel rotation speed. One of Kepler's reaction wheel is shown in Figure.1. Reaction wheels are ideal for precise attitude adjustments. They come in different sizes and have different maximum torques. They usually have a saturation speed at around 5000 rpm, and momentum dumping through thrusters or other means are needed to reduce the wheel speeds.

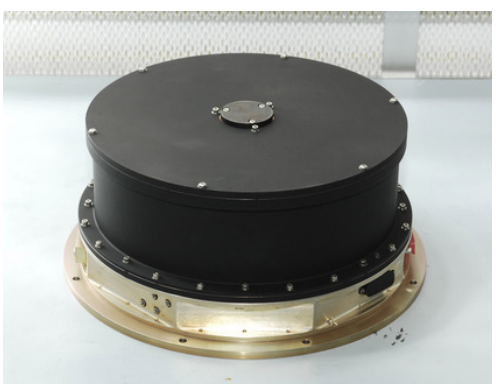

Figure 1: A reaction wheel for the Kepler Telescope.[4]

2 Learning Objectives

I want to learn about how spacecrafts control its attitude while in orbit. In particular, I want to simulate how much torque each reaction wheel should apply to the spacecraft to achieve a certain attitude given 3 orthogonal wheels. I originally wanted to apply Euler angles, angular momentum conservation and rotation matrices.

3 System Model

The model, shown in Figure.2, consists of a spacecraft with three reaction wheels orthogonally aligned with the body frame of the space craft (b_1, b_2, b_3). The inertial frame is (i, j, k). The position of the center of mass of the space craft is not considered. The orientation of the spacecraft is expressed in quaternions, where $\mathbf{e}$ is the Euler axis and θ is the rotation angle around the axis.

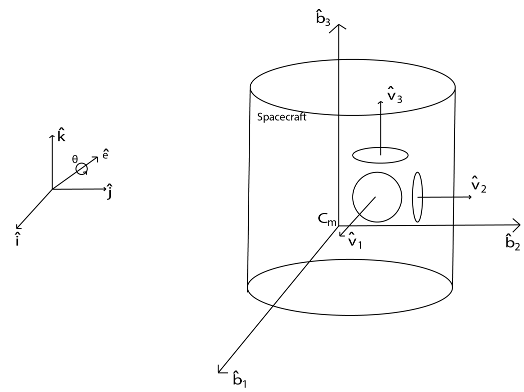

Figure 2: Diagram showing the body frame of the space craft (b_1, b_2, b_3) and inertial frame(i, j, k). A spacecraft is represented as a cylinder and has 3 orthogonal reaction wheels aligned to the body axis. $\mathbf{e}$ and θ are the Euler axis and angle of the quaternion rotation.

3.1 Equations of motion

The equations of motion for the spacecraft is as follow [2]:

$$\begin{aligned} J_{sc}\dot{w} &= (J_{sc}w + J_\alpha v) \times w + u + \tau_{dist}, \\ \dot{v} &= -J_\alpha^{-1}u, \end{aligned} \tag{1}$$

where J_{sc} is the mass moment of inertia of the spacecraft, $w = (w_1; w_2; w_3)$ is the absolute angular velocity of spacecraft body frame (expressed in the body frame), J_α is a diagonal matrix representing the mass moment of inertia for each wheel, u is the control torque from the wheels and τ_{dist} is the external disturbance torque.

3.2 Spacecraft and wheel parameters

The spacecraft is assumed to be cylindrical with a uniform density of $10kg/m^3$, the average density of Kepler telescope [6]. This way, the mass moment of inertia of a spacecraft can be calculated through a single variable: the mass of the spacecraft. The actual mass moment of inertia for a spacecraft is obviously different, and can only be determined through direct measurement or CAD.

The reaction wheel parameter is based on Sunspace's reaction wheel SUN-STAR, which has a moment of inertia of $1.5 \times 10^{-3} kgm^3$, a maximum torque of 0.05 Nm and a saturation speed of 4200 rpm (about 420 rad/s) in both directions. This is a relatively small reaction wheel. For comparison, Hubble's reaction wheels weigh 40kg [4], 20 times the mass of the Sunspace's wheels.

3.3 Control logic

To achieve a rest to rest maneuver from the origin quaternion [0, 0, 0, 1] to a final quaternion q_f, the following linear controller from [1] is used:

$$\begin{aligned} \mathbf{u} &= -sgn(q_4)\mathbf{K}\mathbf{q}_e - \mathbf{C}\mathbf{w}, \\ \mathbf{K} &= kJ_{sc}, \\ \mathbf{C} &= cJ_{sc}, \end{aligned} \tag{2}$$

where u is a vector of the control torque in each axis direction, k and c are controller gain coefficients, and $\mathbf{q}_e$ is the attitude error quaternion, which can be determined from the current attitude quaternion and q_f. The exact matrix come from page 403 of [1]. u is constrained by the maximum torque of 0.05 in either direction through a min and max function. The saturation speed of the reaction wheels is not taken into account in the control logic, but will be considered when analyzing the results.

3.4 Validation of model

To validate that the model simulate reality reasonably, the rotation angle, angular velocities, control torques, and reaction wheel speeds are plotted vs time in Figure.3 for a small rotation angle of 5 deg along a 1-1-1 rotation axis. The components of the angular velocity, control torque and wheel speed are fairly identical because of the symmetric axis. The eigen angle theta exhibits a damped linear system response. The other three variables behave as expected.

Figure.4 shows a larger rotation angle of 60 deg along 1-2-3 rotation axis. The components of the angular velocity, the control torque and the wheel speeds are not the same now because of the different components of the rotation axis. The wheel speeds v also exceeds the saturation limit of 420 rad/s. This means that for a spacecraft of similar or greater mass than 50kg, a relatively large angle can easily cause saturation, suggesting to limit the use of the particular Sunspace reaction wheels to small angles and smaller spacecraft.

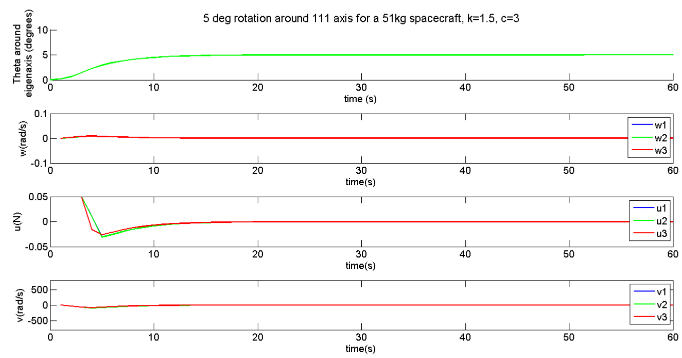

Figure 3: Plot of the eigen angle theta, the 3 components of the angular velocity(w), the control torque from the 3 wheels(u) and the speed of the three wheels(v).

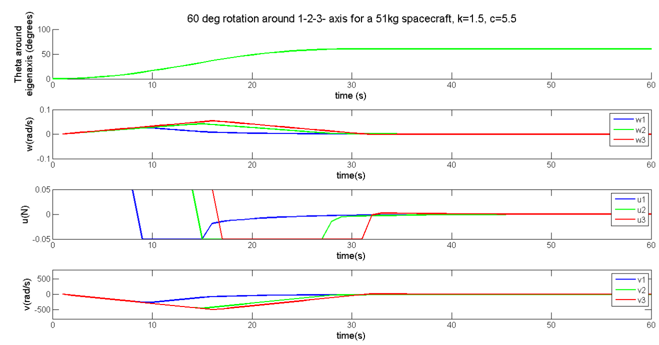

Figure 4: Same as Figure. 3

4 Results

One important output variable to look at is the time it takes to accomplish a rest-to-rest maneuver, represented as the settling time of the rotation angle. The effect of the spacecraft's mass on the best controller gain coefficients and settling time is investigated.

For a given maneuver, the best controller gain coefficients k and c are determined by looping through k values from 0.5 to 5 and c values of 0.5 to 8 in increments of 0.5. This is done for spacecraft masses of 1 to 101kg in increments of 10kg, as shown in Figure.5. The k and c corresponding to the minimum settling time is found for each mass from the matrix and fminsearch is used with that k and c as initial values. The results for a 5 deg, 30 deg, and 60 deg rotation are shown in Figure.6, Figure.7 and Figure.8.

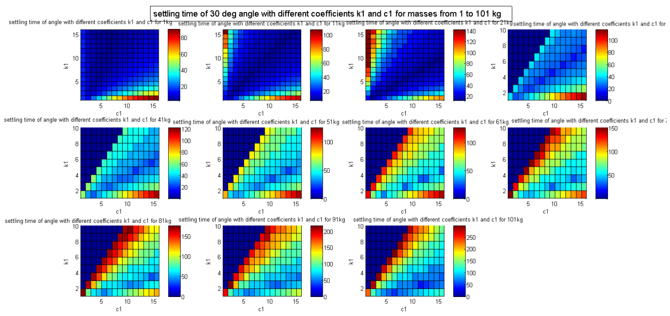

Figure 5: A collection of p-color graph of the settling time for different controller gain coefficients. The eigan axis of rotation is 1-0-0. The actual k and c value is half of the axis value. For masses greater than 30kg, only half of the matrix is calculated (the lower right half) because the minimum time is almost guaranteed to occur in that half.

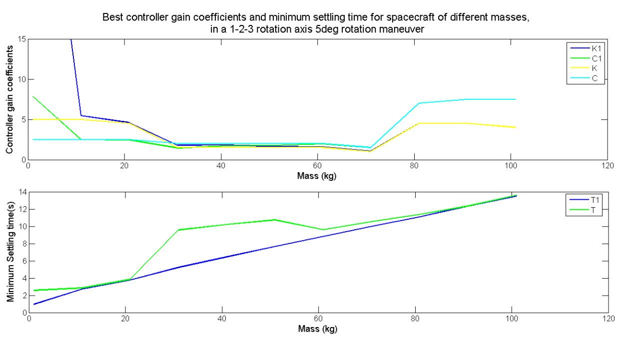

Figure 6: K and C are the best controller gain coefficients gathered from the matrix, while K1 and C1 are the best from fminsearch. Similarly, T corresponds to the settling time for K and C, and T1 corresponds to the settling time for K1 and C1.

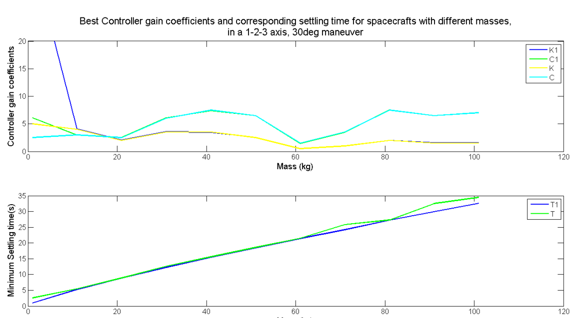

Figure 7: Same as Figure. 6

The settling time increases linearly as the mass of the space craft increases, and not surprisingly, it takes a longer time for a larger angle turn. The best control gain coefficients exhibit interesting patterns. The best k tend to decrease as mass increases, while c increases slightly in the beginning and varies afterwards. The jagged pattern for c in Figure.7 and 8 might be due to the discrete nature of the matrix search. The best values for k are quite similar for the 3 different angles, while and best values for c varies more. k and c have a great effect on the settling time, as the largest settling time with some coefficients can easily be 5 times greater than the lowest settling time with the best coefficients.

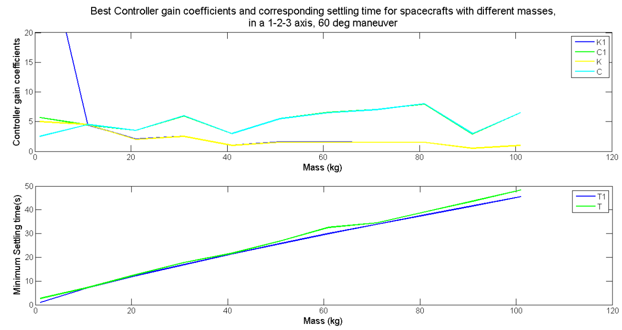

Figure 8: Same as Figure. 6

5 Animation and Visualization

An animation of the rotation of the body frame is implemented in matlab.

6 Improvement

There are a lot more that can be done with the model. First, the spacecraft's mass moment of inertia can be more accurate. However, it is more difficult to measure the mass moment of inertia for a spacecraft in order to use it in the simulation. Secondly, as mentioned in the validation section, the reaction wheels saturate for certain maneuvers. The rotation angle and axis that cause reaction wheels to saturate for different spacecrafts can be investigated quantitatively. This saturation can also be included in the control logic. Thirdly, there might be a better way to find the best controller gain coefficients that is more efficient or accurate than the one I used. Lastly, external torques on the spacecraft such as gravity gradient can be taken into account, as well as power consumption of the wheels as a constraint, as demonstrated in [3].

7 Conclusion

Reaction wheels can theoretically achieve any reorientation maneuver for small and large spacecraft if momentum damping is implemented so that the wheel speeds do not saturate. However, the time it takes for rest-to-rest maneuvers increases as the mass of the spacecraft and as the turn angle increase. There are of course power constraints on the feasibility of reaction wheels in actual spacecrafts, because reaction wheels are generally less power efficient than other control mechanisms such as control moment gyros [3].

The best gain coefficients of a linear state feedback controller that lead to the fastest maneuver can be found roughly through a Matlab simulation. The matlab model is powerful as it can accommodate any set of spacecraft and reaction wheels parameters, and ode45 is good for simulation of closed-loop control systems with a linear state feedback controller.

8 Diagnosis and Reflection

The project went pretty smoothly. The biggest challenge I faced was to define the question to investigate. The model of dynamics and control was fairly straight forward to create in Matlab, but deciding what to do with it is hard because I have to look at what other people care and write about and learn more about the topic. Once I decided on that, I had to write a lot more scripts to accomplish the goal.

While I originally used Euler angles, I realized halfway that quaternions are used more in spacecraft orientation, so I had to abandon my Euler angle script and switch to quaternions. Quaternions are convenient though, and I'm glad to have learnt them. I also learnt about linear state feedback controller and how reaction wheels work. I was able to find many good papers and textbooks and understand most of them.

9 Future Usage

This project is suitable for a future Dynamics class project because it is reasonably scoped (no more than 25 hours of work) and there are many resources available on line. A simple set of instructions for my project would be:

1. Learn about quaternions.

2. Derive equations of motion for a spacecraft with reaction wheels (either by yourself or get it from a paper. [2] is a good one for that).

3. Implement a model using the equations of motion in matlab.

4. Learn about state feedback control logic and implement it in the matlab model.

5. Test your model, debug, decide what you want to investigate with it.

6. Generate results and write the paper.

References

[1] Space Vehecle Dynamics and Control. Wie, Bong. Chapter 7: Rotational Maneuvers and Attitude Control. 1998.

[2] Inertia-Free Spacecraft Attitude Control with Reaction-Wheel Actuation. Avishai Weiss, Xuebo Yang, Ilya Kolmanovsky, and Dennis S. Bernstein. University of Michigan. 2010. http://deepblue.lib.umich.edu/bitstream/handle/2027.42/83656/AIAA-2010-8297-351.pdf?sequence=1

[3] Comparison of Control Moment Gyros and Reaction Wheels for Small Earth-Observing Satellites.Ronny Votel, Doug Sinclair. 26^{th} annual AIAA Conference on Small Satellites. http://deepblue.lib.umich.edu/bitstream/handle/2027.42/83656/AIAA-2010-8297-351.pdf?sequence=1

[4] Angular Momentum in the real world. http://spiff.rit.edu/classes/phys216/workshops/w11c/ang_mom_action.html

[5] Quaternions and spatial rotation. Wikipedia. https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation#Quaternion-derived_rotation_matrix

[6] Kepler(spacecraft). Wikipedia. https://en.wikipedia.org/wiki/Kepler_(spacecraft)

A Matlab code

This is the script that defines parameters, call ode45, plot and does animation:

```
function res=FP_2(m,k1,c1,time,graph)
%close all
%Space craft dimensions
%Isc=FP_Icube(10,.2,.2,.2); %m (kg),a,b,c calculate MOI for a cubic spacecraft
r=nthroot(m/40/pi,3);
Isc=FP_Icylinder(m,2*r,r); %m(kg), h, r calculate MOI for a cylindrical spacecraft
Ia=[1.5e-3,0,0;0,1.5e-3,0;0,0,1.5e-3];

%Control Gain matrices
wn=1; % natural frequency in Hz
zeta=0.05; % value between 0 and 0.05
%k1=2*wn^2;
%c1=2*zeta*wn;
K=k1*Isc;
C=c1*Isc;

%Initial values
Tq=[0;0;0]; %External torque experienced by spacecraft
v_0=[0;0;0];
q_0=[0,0,0,1];
turnangle=30; % desired turn angle in degrees
e=[1,2,3];
en=norm(e);
qc=[sind(turnangle/2)*e(1)/en;sind(turnangle/2)*e(2)/en;sind(turnangle/2)*e(3)/en;cosd(turnangle/2)]; % desired final attitude in quartenions

w_0=[0,0,0];
initial=[q_0,w_0,v_0']; %q1,q2,q3,q4,wb1, wb2, wb3 (of spacecraft), v(rotation speed of wheels)
%time=200; %run time in seconds


[T,Z]=ode45(@(t,y)FP_2_rates(t,y,Isc,Ia,K,C,qc),[0:time],initial);
Z(:,1:10);
Q4=Z(:,4);
THETA=2*acosd(Q4);

if graph==1
    figure
    subplot(4,1,1)
    hold on
    plot(T,THETA,'g','LineWidth',1.5)
    xlabel('time (s)','FontSize',14);
    ylabel('Rotation angle theta around eigenaxis (degrees)','FontSize',14);
    title([num2str(turnangle),' deg rotation around ', num2str(e(1)),num2str(e(2)),num2str(e(3)),' axis for a ', num2str(m), 'kg spacecraft , k=',num2str(k1),', c=',num2str(c1)],'FontSize',16);
Q=Z(:,1:4);
W=Z(:,5:7);

    for i=1:length(T)
    qe=[qc(4),qc(3),-qc(2),-qc(1);...
        -qc(3),qc(4),qc(1),-qc(2);...
        qc(2),-qc(1),qc(4),-qc(3)]*Q(i,:)';

     U(i,1:3)=min(0.05, max(-0.05, (-sign(Q(i,4))*K*qe-C*W(i,:)')'));
    end

    subplot(4,1,2)
    plot(Z(:,5),'LineWidth',1.5);
    hold on
    plot(Z(:,6),'g','LineWidth',1.5);
    plot(Z(:,7),'r','LineWidth',1.5);
    xlabel('time(s)','FontSize',14);
    ylabel('w(rad/s)','FontSize',14);
    legend('w1','w2','w3');
    axis([0 time -.1 .1]);

    subplot(4,1,3)
    plot(U(:,1),'LineWidth',1.5);
    hold on
    plot(U(:,2),'g','LineWidth',1.5);
    plot(U(:,3),'r','LineWidth',1.5);
    xlabel('time(s)','FontSize',14);
    ylabel('u(N)','FontSize',14);
    legend('u1','u2','u3');
    axis([0 time -.05 .05]);

    subplot(4,1,4)
    plot(Z(:,8),'LineWidth',1.5);
    hold on
    plot(Z(:,9),'g','LineWidth',1.5);
    plot(Z(:,10),'r','LineWidth',1.5);
    xlabel('time(s)','FontSize',14);
    ylabel('v(rad/s)','FontSize',14);
    legend('v1','v2','v3');
    axis([0 time -800 800]);
```

```

    %rotation matrix to translate quaternion to orientation

    for i=1:length(T)
        qi=Z(i,1);
        qj=Z(i,2);
        qk=Z(i,3);
        qr=Z(i,4);
        R=[1-2*qj^2-2*qk^2, 2*(qi*qj-qk*qr), 2*(qi*qk+qj*qr);...
           2*(qi*qj+qk*qr), 1-2*qi^2-2*qk^2, 2*(qj*qk-qi*qr);...
           2*(qi*qk-qj*qr), 2*(qj*qk+qi*qr), 1-2*qi^2-2*qj^2];
        xaxis(i,1:3)= (R*[1;0;0])';
        yaxis(i,1:3)= (R*[0;1;0])';
        zaxis(i,1:3)= (R*[0;0;1])';
%          xaxis(i,1:3)*yaxis(i,1:3)'
%          zaxis(i,1:3)*yaxis(i,1:3)'
%          xaxis(i,1:3)*zaxis(i,1:3)'
    end

    %animation
    figure
    for i=1:length(T)
        quiver3(0,0,0,xaxis(i,1),xaxis(i,2),xaxis(i,3),'g','LineWidth',2);
        hold on
        quiver3(0,0,0,yaxis(i,1),yaxis(i,2),yaxis(i,3),'c','LineWidth',2);
        quiver3(0,0,0,zaxis(i,1),zaxis(i,2),zaxis(i,3),'r','LineWidth',2);
        quiver3(0,0,0,1,0,0,'k','LineWidth',1);
        quiver3(0,0,0,0,1,0,'k','LineWidth',1);
        quiver3(0,0,0,0,0,1,'k','LineWidth',1);
        quiver3(0,0,0,e(1),e(2),e(3),'k','LineWidth',2)
        xlabel('x'); ylabel('y'); zlabel('z');
        drawnow
        hold off
    end

end


S=stepinfo(THETA,T);
res=S.SettlingTime;
end
```

This is the ode45 rate function that contains the equation of motions and control logic:

```
function res=FP_2_rates(t,z,Isc,Ia,K,C,qc)
        q=z(1:4);
        w=z(5:7);
        v=z(8:10);

        qe=[qc(4),qc(3),-qc(2),-qc(1);...
            -qc(3),qc(4),qc(1),-qc(2);...
            qc(2),-qc(1),qc(4),-qc(3)]*q;

        u=-sign(q(4))*K*qe-C*w;    % control input based on qe and omega
        u= min(0.05, max(-0.05, u)); % saturation limit based on max torque of 50mNm, 0.05 Nm

        dvdt=inv(Ia)*(-u);

        dqdt=1/2*[0 w(3) -w(2) w(1);...
                  -w(3) 0 w(1) w(2);...
                  w(2) -w(1) 0 w(3);...
                  -w(1) -w(2) -w(3) 0]*q;

        dwdt=inv(Isc)*(cross((Isc*w+Ia*v),w)+u);

        res=[dqdt;dwdt;dvdt];
end
```

This is script that calculates the mass moment of inertia of a cylindert:

```
function res=FP_Icylinder(m,h,r)
res=[m*(3*r^2+h^2)/12,0,0;...
    0,m*(3*r^2+h^2)/12,0;...
    0,0, m*r^2/2];
end
```