Model Reference Adaptive Control for Satellite Attitude, or How To Control a Satellite When it Gives You Attitude

J.Bryan Figueroa

December 2019

1 Abstract

Presented in this term paper is a derivation of state-space representation for attitude control of a satellite in low Earth orbit. We begin with simple pitch consideration, and then we consider the scenario where the inertial terms makes the dynamics non-homogenious.

2 Table of Contents

3 Introduction

Artificial satellites are distinguished from natural satellites in that they have been intentionally placed in orbit of something. It could be an asteroid such as 433 Eros or our very own planet. With a satellite, two types of position controllers could exist. Altitude control, which is the height of the satellite to what it's orbiting, and attitude control, which is the satellite's predisposition to that which it orbits. Here, we'll look at attitude control of three states of rotation by the Euler angles. Those three states as pitch, yaw, and roll (ϕ, θ, ψ). Given a circular orbit, the linearized attitude dynamics can be represented by:

$$I_x\ddot{\phi} + 4(I_y - I_z)\omega_0^2\phi + (I_y - I_z - I_x)\omega_0\dot{\phi} = u_i \tag{1}$$

$$I_y\ddot{\theta} + 3\omega_0^2(I_x - I_z)\theta = u_2 \tag{2}$$

$$I_z\ddot{\psi} + (I_y - I_x)\omega_0^2\psi + (I_x + I_z - I_y)\omega_0\dot{\phi} = u_3 \tag{3}$$

The observation here is that the pitch angle θ is decoupled from the rest of the system. A similar satellite orbiting 344 Eros would have altered dynamics given by

Attitude control is important because the instrumentation carried by a satellite usually works if it's being pointed at a nice location. Ideally, one would have their instruments pointing perpendicular to the surface which they are tracking or receiving commands from.

Solutions to these problems exist, but we will attempt to solve them for systems where mass and moments of inertia are varying. Furthermore, because pragmatic solutions exist only in the digital world, we will also attempt to do this by discretization of our system.

For satellite orbit of an asteroid, work has been done by K. D. Kumar. Their attitude motion of a satellite is modeled in section 4.2 of this report.

4 Mathematical Modeling

4.1 System Parameters

Take equations 1, 2, and 3 above. The satellite moments of inertia are $I_y, I_x, I_z = (20, 18, 15)[kg.m^2]$, and the trust inputs are $u = (u_1, u_2, u_3)$. Low Earth orbit or LOW cange range from 160 km up to 1000 km above the Earth surface. Satellites in a circular orbit can travel at a speed of around 7.8 km per second. A period of about 90 minutes exists at this velocity. Assuming the altitude above earth is $R_h = 400[km]$, and that the radius of a circular Earth is $R_E = 6,371[km]$, the orbital rate ω_0 is calculated as

$$\omega_0 = \sqrt{\frac{\mu}{R_h + R_E}}(4)$$

where $\mu = GM_e$, G = universal gravitational constant, and M_e is the mass of the Earth. Thus μ can be taken to be

$$\mu = GM_e = 3.986004418x10^5[km^3/sec^2] \tag{5}$$

Here, μ is in units of kilometers and seconds. Had it been in units of meters and seconds, we would see the more common

$$\mu = 3.986004418x10^{14}[m^3/sec^2] \tag{6}$$

4.2 Pitch Dynamics

Consider equation 2 above - rewritten here for the reader's comfort

$$I_y\ddot{\theta} + 3\omega_0^2(I_x - I_z)\theta = u_2 \tag{7}$$

Note that the only rotational term here is θ, the pitch state of our satellite. Thus, it is implied that the pitch dynamics will be decoupled from the yaw and roll dynmaics of the satellite. Using subsitution

$$x_1 = \theta, \dot{x_1} = \dot{\theta}, x_2 = \dot{x_1}, \dot{x_2} = \ddot{\theta} \tag{8}$$

Now equation 7 can be rewritten as

$$I_y \dot{x_2} + 3\omega_0^2(I_x - I_z)x_1 = u_2 \tag{9}$$

isolating for $\dot{x_2}$ yields

$$\dot{x_2} = \frac{-3\omega_0^2(I_x + I_z)}{I_y} x_1 + \frac{1}{I_y} u_2 \tag{10}$$

From equation 10 along with the variable subtiutions created above, we can convert this into a state-space representation

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-3\omega_0^2(I_x+I_z)}{I_y} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -\frac{1}{I_y} \end{bmatrix} \begin{bmatrix} u_2 \end{bmatrix} \tag{11}$$

The form of this equation matches that of

$$\dot{X} = AX + BU \tag{12}$$

where, $A \in R^2, B \in R^2, X \in R^2$, and $U \in R^1$. Thus, U begin in a 1x1 matrix is a scalar quanity. The cross product of B and U create a 2x1 matrix which matches the dimensions of the other term in equation 11. The x subtitute variable used here has no relation to subscripts used in the inertia parameters.

4.2.1 Pitch Calculations

With the values stated about the system parameters in section 4.1 placed into the dynamics in equatoin 11, we get the systems

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1.000000000000000 \\ -0.000001070035259 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1.000000000000000 \\ -0.055555555555556 \end{bmatrix} \begin{bmatrix} u_2 \end{bmatrix} \tag{13}$$

The .... of this system are shown in the appendeix in the script pitchConsiderations.m The parameters within the A matrix differ by several orders of magnitude, and we question the stiffness of this matrix. The numerical solver ode45() will be tried.

4.3 Controllability and Observability

X theorem sates that the rank of the controllability matrix determines ...

The Y observaility matrix given by ...

5 Full State Dynamics

Following the same process we used in section 4.2 for the pitch dynamics, we can isolate the roll and yaw states into canoical form. Because these are second derivative equations, we'll need two states for each additional ODE for a total of six states x_1 through x_2

Looking back to equations 1 through 3 of our original system, let's make the following subtitutions

$$\begin{aligned} x_1 = \psi, \dot{x_1} = \dot{\psi} \\ x_2 = \dot{x_1}, \dot{x_2} = \ddot{x_1} = \ddot{\psi} \\ x_3 = \theta, \dot{x_3} = \dot{\theta} \\ x_4 = \dot{x_2}, \dot{x_4} = \ddot{x_3} = \ddot{\theta} \\ x_5 = \phi, \dot{x_5} = \dot{\phi} \\ x_6 = \dot{x_5}, \dot{x_6} = \ddot{x_5} = \ddot{\phi} \end{aligned} \tag{14}$$

Paralleling the derivations we did in section 4.2, come up with

$$\dot{x_2} = \frac{-4(I_y - I_z)\omega_o^2}{I_x} x_1 - \frac{(I_y - I_z - I_x)\omega_o}{I_x} x_6 + \frac{1}{I_x} u_1 \tag{15}$$

$$\dot{x_4} = \frac{-3\omega_0^2(I_x + I_z)}{I_y} x_3 + \frac{1}{I_y} u_2 \tag{16}$$

$$\dot{x_6} = \frac{-(I_y - I_x)\omega^2}{I_z} x_5 - \frac{(I_x + I_z - I_y)\omega_o}{I_z} x_2 + \frac{1}{I_z} u_3 \tag{17}$$

Note that equation 16 represents equation 10, but with a change of subscripts. With the information from equations 14 through 17 we can now build the state-space representation. Pay particular attention to the B matrix.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-4(I_y - I_z)}{I_x}\omega_o^2 & 0 & 0 & 0 & 0 & \frac{-(I_y - I_z - I_x)}{I_x}\omega_o \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-3\omega_0^2(I_x + I_z)}{I_y} & 0 & 0 & 0 \\ i0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0\frac{-(I_x + I_z - I_y)}{I_z}\omega_o & 0 & 0 & \frac{-(I_y - I_x)}{I_z}\omega_o^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{I_y} \\ 0 \\ \frac{1}{I_x} \\ 0 \\ \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} 0 \\ u_1 \\ 0 \\ u_2 \\ 0 \\ u_3 \end{bmatrix}^T \tag{18}$$

Where the U matrix is transposed

5.0.1 Controllability and Observability

6 Results

6.1 Data

6.2 graphs

7 Conclusion

7.1 limitations

8 Appendix

8.1 adaptiveControl.py

```
// adaptiveControl.py
# Simulations for attitude control problem
# check requirements.txt

# Julio B. Figueroa
# December, 2019

##### Acknowledgements #####
# this code is heavily inspired by the work of Dhruv Laad
# via https://github.com/Thalaivar/control_practice

# test run
import matplotlib.pyplot as plt
from scipy.integrate import ode
import numpy as np
from numpy import sin

# adaptive gain
gamma = 2

def model(t, X, params):
    y, ym, ar_hat, ay_hat = X

    if params == 1:
        r = 4*sin(3*t)
    else:
        r = 4
    e = y - ym
    u = ar_hat*r + ay_hat*y

    ydot = y + 3*u
    ymdot = -4*ym + 4*r
```

```
    ar_hat_dot = -gamma*e*r
    ay_hat_dot = -gamma*e*y

    return [ydot, ymdot, ar_hat_dot, ay_hat_dot]

def main(option):
    # start with initital conditions
    X0 = [0, 0, 0, 0]
    t0 = 0
    t1 = 8                  # change this freely now!
    dt = 0.01               # this too..
    steps = int(t1 / dt) + 1 # keep this

    r = ode(model).set_integrator('vode', method='bdf')
    r.set_initial_value(X0, t0).set_f_params(option)

    x = np.zeros((steps,4))
    t = []
    i = 0
    while r.successful() and r.t < t1: # checks if integration was
        successful
        r.integrate(r.t + dt)        # returns the integrated value at
            input
        x[i] = r.y
        t.append(r.t)
        i += 1
    return x,t

# really, this is the plotting section
# we're plotting the same model with two different inputs
# then we can compare them side by side, or top by down... top by
    bottom...
if __name__ == '__main__':

    # main(1) has the in control input, main(0) was the constant control
        input
    x1, t1 = main(1)         # rich
    x2, t2 = main(0)         # less rich

    steps = len(t1)          # "should" be 501
    # we're hoping len(t1) = len(t2) for now, if it changes then add a
        steps2
    # steps = int(t1 / dt) + 1 # +1 because python, see main(option)
    # basically, I just don't want to hardcode it

    # data for the first plot
    y1     = np.reshape(x1[:, 0], (steps,))
    ym1    = np.reshape(x1[:, 1], (steps,))
    ar_hat1 = np.reshape(x1[:, 2], (steps,))
```

```
ay_hat1 = np.reshape(x1[:, 3], (steps,))

# to compare against the reference, so... this is for adaptive
    regulation?
ar_ideal1 = (4/3)*np.ones((steps,))
ay_ideal1 = -(5/3)*np.ones((steps,))

# data for the second plot
y2     = np.reshape(x2[:, 0], (steps,))
ym2    = np.reshape(x2[:, 1], (steps,))
ar_hat2 = np.reshape(x2[:, 2], (steps,))
ay_hat2 = np.reshape(x2[:, 3], (steps,))

# now draw
plt.subplot(211)
plt.plot(t1, y1, 'r', t1, ym1, 'k', t2, y2, 'b', t2, ym2, 'g')
plt.ylabel('tracking performance')
plt.grid()

plt.subplot(212)
plt.plot(t1, ar_hat1, 'b', t1, ay_hat1, 'k', t1, ar_ideal1, t1,
    ay_ideal1)
plt.ylabel('parameter estimation')
plt.grid()

plt.show()
```

9 References

[**ruiter**] test [Dir81] test [F L14] test [Ein05] test [Knu73] test

References

[Ein05] Albert Einstein. "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]". In: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: http://dx.doi.org/10.1002/andp.19053221004.

[Knu73] Donald E. Knuth. "Fundamental Algorithms". In: Addison-Wesley, 1973. Chap. 1.2.

[Dir81] Paul Adrien Maurice Dirac. *The Principles of Quantum Mechanics*. International series of monographs on physics. Clarendon Press, 1981. ISBN: 9780198520115.

[F L14] John L. Crassidis F. Landis Markley. *Fundamentals of Spaceraft Attitude Determination and Control*. Space Technology Library. Springer, 2014. ISBN: 9781493908011.