Module 2, B-set 0: Attitude Control Breakdown

November 3, 2016

Goals:

By the end of this building block you should be able to:

1. Identify the key concepts that you need to learn and understand in order to design an attitude control system.
2. Clearly state quantitative questions which must be answered in order to control the attitude of a satellite.
3. Enumerate the key steps necessary in the attitude control process.
4. Record and process acceleration and orientation data from a smartphone.

Overview and Orientation:

This is a very short Building Block dedicated to doing a deconstruction of attitude control for a satellite. What are the big ideas that one needs to learn in order to understand and create an attitude control system? Most of your time during this block should be dedicated to finishing your final write-up of your model project from commmunications.

A short note: we are asking you in this breakdown to think about attitude control in a couple of different ways. Feel free to move back and forth between these two different exercises: your results from one may inform the other and vice versa.

Attitude Control Breakdown (1.5 hours)

Concept Map

One way to state the overarching question of this module is `how can we be sure our satellite is in the right place and is pointing in the right direction?`. You should individually tackle this question, breaking it down into sub-questions and researching each of those. Build a concept map from this.

As a reminder, the process of concept mapping goes as follows:

1. Write the key term at the top or in the center of a piece of butcher paper. Circle it, since you don't know it.

2. Research it, and identify terms that are immediately associated with it. Write them down and connect them with lines.
3. Label the lines to indicate what you understand the connection to be.
4. Circle new terms you don't understand, and break these down too.
5. Continue this process until you are down at a fine-grain level of detail: at this level of detail, you should be able to think about things in terms of quantitative questions associated with each topic: ie, what torque do we need to apply, and about which axis?

Once you've created your map, identify 8-10 key underlying concepts that you think you will need to understand in order to be able to make sense out of how satellite attitude control works. By "underlying concepts," we mean more fundamental ideas as opposed to stuff that is specific to satellites: for example, "angular momentum" as opposed to "reaction wheel". Write them down somewhere. You will discuss these next class with your table.

System Diagram

In addition to the concept map, we'd like you to make a SigSys Abstraction in block diagram form for an attitude control system. In other words: think about the exact sequence of events/actions/operations which need to happen to make an attitude control adjustment and represent this in block diagram form. Describe what the blocks do (you don't have to write out specific mathematical operations: it's ok if these are very general.) and clearly specify the inputs and outputs of each block.

Playing with IMU Data from a Smartphone (2.5 hours)

To help us think about the position and attitude (orientation in space) dynamics that you need to know to plan a cubesat mission, you'll record and process (linear) acceleration and orientation data from a smartphone's inertial measurement units (IMUs). You can do this with a partner if you don't have a suitable phone or are unable to access the required software.

Obtaining the Phone Support Package

MathWorks has a mobile app that can access and upload IMU data to a computer or online account. Instructions for setting it up follow. Feel free to use an alternative app if you prefer.

- In the MATLAB Command window, enter

```
supportPackageInstaller
```

- Select `Install from Internet`, click `Next`, and select `iOS sensors` or `Android` in the `Support for:` list. Only the MATLAB support package is necessary.
- Follow the instructions and default settings provided to complete the installation. You will have to create a Mathworks account if you don't already have one.

Phone Setup

- Download the MATLAB Mobile App through the Google Play or App Store.
- Connect your phone to the OLIN network by following the instructions at `wikis.olin.edu/it/doku.php`, `Mobile`, `Configure Olin Wifi on Android`. The instructions here work for both Android and iPhone.

Connect Phone and Computer

- On your computer, in your MATLAB Command window, input

```
connector on yourpassword
```

  where yourpassword is a password of your choice.
- Launch MATLAB Mobile on your phone. Select Connect to Your Computer. If you already had MATLAB Mobile before this assignment, you may need to go to the settings screen and select Add a Computer. Enter the IP address that was displayed on your computer when you turned the connector on.
- In your Command window on the computer enter

```
m= mobiledev
```

  The output should be

```
Connected: 1
```

Note that if you close your computer and come back to this activity, your IP address will have changed and you will have to redo these steps.

Recording and Sending Data

- Turn on the sensors for angular velocity, orientation and acceleration by

```
m.AngularVelocitySensorEnabled = 1;
m.OrientationSensorEnabled = 1;
m.AccelerationSensorEnabled = 1;
```

- To start logging, enter

```
m.Logging = 1;
```

Shake, twist, or throw your phone and then input

```
m.Logging = 0;
```

to stop logging. Note, in the iPhone version, there is a blue button in the top, right corner of the screen that starts and stops data logging.

Plotting Data

a Retrieve your logged sensor data using

```
[av, tav] = angvellog(m);
[o, to] = orientlog(m);
[a, t] = accellog(m);
```

b If you plot this data directly, what do you notice about the time axis? At the moment, the data is plotted against relative time. You can convert time to absolute time by

```
tInit = datetime(m.InitialTimestamp,...
'InputFormat', 'dd-MM-yyyy HH:mm:ss.SSS');
tAngVel = tInit + seconds(tav);
tOrient = tInit + seconds(to);
tAccel = tInit + seconds(t);
```

You can also convert the angular units from radians to degrees, *e.g.*, for angular velocity

```
yAngVelDeg = yAngVel * 180/pi;
```

1. *Playing with Acceleration*

(a) Rest your phone on top of a table. Why is the z-direction accelerometer displaying $\approx +9.8$ m/s^2 even though the phone is essentially stationary?

(b) Log acceleration data as you walk some distance with your phone in a straight line. Based on this data, estimate how far you walked. Remember that integration produces an integration constant. How does it affect calculations of velocity and displacement? How can a high-pass filter be of help processing the data?

2. *Playing with Orientation*

(a) Log data as you rotate the phone back-and-forth about a single axis (x, y, or z) of rotation. Numerically integrate angular velocity about one axis and compare it to the corresponding orientation angle data. Notice that there are significant differences. Why do think this is the case?

(b) Log data as you rotate the phone back-and-forth about a single axis that is not aligned with x, y, or z. Plot the orientation angle data. Explain how these plots describe how you were rotating the phone.