Algorithm for Trajectory Optimization of Magnetorquer-Based Underactuated Control

Andrew Gatherer, Zac Manchester
Stanford University
496 Lomita Mall, Stanford, CA 94305
gatherer@stanford.edu

Faculty Advisor: Zac Manchester
Stanford University

April 4, 2019

1 Introduction

In order to attack the problem of underactuated control for magnetorquer-only attitude stabilization and slewing of small satellites, a trajectory optimization method was used to come to an optimal trajectory. Once the optimal trajectory was arrived upon, the trajectory was simulated against expected noise through a time-varying LQR algorithm.

This method was found to provide reasonable and robust results for the dynamics of satellite attitude explored for the purpose of the research. The trajectory optimization method used included an inner loop with an iLQR solver and an outer loop that employed constrained lagrangians. This solver was developed by Brian Jackson and Taylor Howell, doctoral students in the Robotic Exploration Lab at Stanford[1].

Ultimately, the results from this simulation were seen to be strong enough to continue testing this algorithm on hardware, or what's known as a "flat-sat." Future research will encompass uploading this algorithm to actual CubeSat hardware, wherein the control of the system will be simulated. The algorithm will be run on-board the microprocessor, and the control input generated will be output to another computer, where the dynamics are simulated. The control input generated will be compared to the simulation to again improve the fidelity of the control method.

2 Initialization

2.1 Physical Parameters

The first order of business is to describe the physical characteristics of the satellite and the necessary constants that describe the Earth. Namely, the specific gravitational constant and the radius of the orbiting body (Earth) should be described. Additionally, the mass, the moments, the ballistic coeffieint, and products of inertia should be described. Then the time information is required, including the Modified Julian Day (for ECEF consideration) and the length of the simulation. The orbital parameters or starting location are then required to initiailzie the satellite's motion, which will be then propagated forward in a later section.

Most importantly, the number of knot points is chosen. There is a fine balance between increasing the number of knot points to ensure clean control and decreasing the number of knot points to allow for fast calculation. It was found experimentally through many trials that the knot points must occur at a frequency of greater than 2 Hz (one every 0.5 seconds) in order to model the dynamics appropriately, especially when the time varying linear quadratic regulator is attached to the system for simulation purposes.

2.2 Magnetic Field Generation

Rather than incorporate the magnetic field in the dynamics, significant computational complexity was avoided by calculating the motion of the satellite over the course of the time range and then creating a vector of the magnetic field over each knot point in consideration. Then, in future dynamics, the time was incorporated into the state such that the dynamics could reference the magnetic field vector given a specific time step. This provided significant dividends to helping the solver perform calculations at speed.

For the dynamics used for the propagation of the satellite, first the Greenwich Standard Mean Time is calculated and used to transform the known ECI position into an ECEF position, which is then incorporated into the NRLMSISE-00 model for the atmosphere in order to find the density at the satellite's location. Using this density, the known velocity of the satellite, and the rotation speed of the Earth, the acceleration from drag, f_{drag} is found. The J2 acceleration is also found through:

$$\begin{aligned} f_{J2} &= J2 * r_x/|r|^7 * (6 * r_z - 1.5 * (r_x^2 + r_y^2) && (1) \\ & \quad J2 * r_y/|r|^7 * (6 * r_z - 1.5 * (r_x^2 + r_y^2) && (2) \\ & \quad J2 * r_z/|r|^7 * (3 * r_z - 4.5 * (r_x^2 + r_y^2) && (3) \end{aligned}$$

Then the accelerations and velocities ($\dot{x}$) are returned to the solver. For the orbit propagation, a Runge-Kutta 4 solver was used, which showed acceptable errors over the relatively small time ranges(less than 10% of an orbit).

Now with the ECI positions at ever point of the orbit, the ECEF positions are again found using the GMST calculated earlier. Given the ECEF position, it is straightforward to calculate the geocentric latitude and longitudes of every knot point, which is critical to determining the magnetic field strength.

Again, rather than call the magnetic field model in the dynamics, a known magnetic field model was smoothed over. Using the IGRF function in the SatelliteToolbox Julia package, a selection of points was taken and a 2 dimensional spline of the magnetic field strength was calculated over the entire Earth. This smoothed out any patch discontinuities in the magnetic field that were observed (leading to unstable results), and made for a much simpler trajectory optimization problem.

With the latitudes and longitudes of the satellite's knot points in hand, the magnetic field at every timestep can be found by referencing the smoothed magnetic field. This is then stored in an array for use in the dynamics.

3 Trajectory Optimization

This project used the ALTRO trajectory optimization solver developed by Brian Jackson and Taylor Howell, and more information on the solver can be found through their publications or through their Julia package. This document will go into little detail about the specifics of their solver but rather how the interface was developed for specifically this purpose.

3.1 Expansion Function

One of the most critical issues with using the quaternion in an optimal control problem is assigning an appropriate error metric. Rather than simply use LQR ($J = x_f^t * Q_f * x_f + \Sigma(x^t * Q * x + u^t * R * u)$, a more complex method of dealing with the quaternions is required, since they are normalized to 1 by definition, and therefore cannot be minimized to 0 like normal linear states. This was done through Modified Rodrigues Parameters[**?**], which incorporates information about the quaternion and decreases the number of states in the quaternion from 4 to 3.

3.2 Power Efficiency

3.3 Computational Requirements

4 Conclusion

References

[1] Jackson, Brian and Howell, Taylor, *ALTRO: A Fast Solver for Constrained Optimization*, 2018.