4-25-2021

# CubeSat Reaction Wheel Simulation

Joshua Conway

Barkley Hunter

Jacob Kincheloe

# CubeSat Reaction Wheel Simulation

**By**

**Joshua Conway Barkley Hunter Jacob Kincheloe**

**An Honors Capstone**

**submitted in partial fulfillment of the requirements**

**for the Honors Diploma**

**to**

**The Honors College**

**of**

**The University of Alabama in Huntsville**

**April 25, 2021**

**Honors Capstone Director:**

**Dr. Matthew Turner**

Research Engineer

Joshua Conway 4/22/2021
Student Date

Jacob Kincheloe 4/22/21
Student Date

Barkley Hunter 4/22/2021
Student Date

Matthew W. Turner — Digitally signed by Matthew W. Turner Date: 2021.04.22 22:15:34 -05'00'
Director Date

D. Keith Hollingsworth — Digitally signed by D. Keith Hollingsworth Date: 2021.04.23 17:05:25 -05'00'
Department Chair Date

William Wilkerson — Digitally signed by William Wilkerson Date: 2021.04.25 14:53:00 -05'00'
Honors College Dean Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

**Honors Thesis Copyright Permission**

**This form must be signed by the student and submitted as a bound part of the thesis.**

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Joshua Conway

Student Name (printed)

Student Signature

4/22/2021

Date

Jacob Kincheloe

Student Name (printed)

Student Signature

4/22/2021

Date

Barkley Hunter

Student Name (printed)

Student Signature

4/22/2021

Date

# Table of Contents

# Abstract

The goal of this project was to develop an Attitude Determination and Control System (ADACS) software model that could simulate how a CubeSat system would behave and operate while preparing for a MEMS Digital Thruster (MDT) test, which was the science objective and primary focus of this group's senior design project. This model was developed by analyzing the characteristics, properties, and operational limitations of several existing reaction wheel systems. These characteristics were then used to develop a computer program that would calculate the approximate spin-down time for the reaction wheel system. In the end, this model gives a basic first order calculation of an ADACS system's operating speed which will help CubeSat designers make decisions about what system would be most suitable for their mission. A second round of research was then carried out with the goal of determining how the program could continue developing in the future in order to increase its usefulness and performance.

# Introduction

Before discussing the development of the software model and the equations and calculations used therein, it is important to establish what exactly the ADACS does and why its functions are so important to a mission system as a whole. The term ADACS refers to a spacecraft's Attitude Determination and Control System and, as the name suggests, this subsystem determines the position and orientation of the spacecraft and then implements the use of control systems in order to reposition and reorient the spacecraft as necessary. Regardless of what manner of mission you are designing, the positioning of the mission's spacecraft is undoubtedly going to be one of if not the most important factor since orbital crafts experience many factors that impart torques and generate motion in the system, such as gravity gradients, aerodynamic drag, magnetic torques, solar radiation, mass expulsion, and even internal disturbances from on-board equipment that may negatively affect the spacecraft or the mission. The ability to control the spacecraft orientation is also crucial for accomplishing science objectives since most scientific instruments used in objective analysis can only operate at their intended capacity if they are oriented towards the object that they intend to measure or analyze. To perform these functions the ADACS can utilize various types of equipment. For attitude determination there are a wide variety of sensor types including GPS, star trackers, limb sensors, rate gyros, and inertial measurement units. The sensor type being employed depends on the type of orbit being used and the nature of the mission, though star trackers are a very common selection. For attitude control spacecraft have the option of using any combination of reaction wheels, control moment gyros, magnetic torque rods, and thrusters. However, for small satellites like the CubeSats in this analysis, reaction wheel systems are the most commonly used with magnetic torque rods sometimes being used in tandem with reaction wheel systems for desaturation purposes.

For the purposes of this project, a general model for analyzing the performance of reaction wheel subsystems was developed since they are the most common attitude control component utilized in CubeSat systems. The model was developed with the intention of providing a first order analysis of reaction wheel systems' performance, primarily their spin-down time, for the purpose of system selection in mission design. In addition to this model, research has been performed and discussion provided for ways that the model can be improved upon in order to eventually account for more complex mechanics that affect the spacecraft as a whole rather than just the reaction wheel subsystem. In order to accomplish these goals in an efficient and timely manner, the three team members divided project tasks into three major project roles. Role one was to gather data on current state-of-the-art CubeSat ADACS systems in order to determine the characteristics and properties for simulation and the limits to their operation. Role two was to research methods for ADACS simulation including the numerical methods used, the limitations of the simulations, the coordinate systems used, and the methods for dynamics simulation. Role three was to take the data gathered on existing systems and simulations and develop a model for analyzing reaction wheel performance.

# Developing The Model

In order to practically apply the team's research into ADACS, a MATLAB program was developed to produce a first order approximation of the spin-down time of any disc-shaped reaction wheel system using inputs easily obtained from most CubeSat reaction wheel datasheets. The spin-down time of a reaction wheel is the length of time required for the wheel to slow to a stop, thus releasing its stored momentum and returning it into the body of the spacecraft. Such a maneuver would be necessary in situations where the ADACS needs to be shut down in order to conserve power or allow the spacecraft to freely reorient itself without interference. The latter scenario was central to the team's senior design project, the MEMS Digital Thruster Technology Demonstration Mission (MDT TDM) CubeSat, as the 3U satellite's ADACS system needed to be spun down and deactivated prior to each MDT test in order to allow the experimental thrusters to impart a measurable moment to the spacecraft.

## Model Mathematics

The mathematics of the first order spin-down time model are fairly straightforward. The ability of a reaction wheel to convert the angular velocity of a vehicle into angular velocity "stored" within the wheel itself is commonly expressed in terms of an angular momentum capacity. Angular momentum, $L$, and angular velocity, $\omega$, are linearly related to one another by the moment of inertia $I$ of the object to which the momentum and velocity belong, as seen in Equation 1 below (9).

$$L = I \cdot \omega \qquad \text{(Eq. 1)}$$

The moment of inertia of an object, or its resistance to changes in angular velocity, is calculated differently depending on the physical shape of the object. Satellite attitude control systems can exist in multiple shapes, most notably circular disks that rotate around their center in one axis and spheres that rotate around their center in three axes. This tool assumes any reaction wheel it considers to be a flat disc rotating in a single perpendicular axis about its center, which means that the moment of inertia of any wheel can be found from the wheel's mass *m* and radius *r* according to Equation 2 (10,11).

$$I = \frac{1}{2} \cdot m \cdot r^2 \qquad \text{(Eq. 2)}$$

With the ability to find the angular velocity of a disc-shaped reaction wheel from its physical dimensions and the amount of angular momentum stored by the wheel, the only additional characteristic needed to estimate the wheel's spin-down time is the rate at which the wheel decelerates. The acceleration and deceleration of real-world reaction wheels are non-linear due to the effects of multiple types of friction that vary with different aspects of the reaction wheel's movement. However, thoroughly modelling these frictional forces and their interactions requires a much more complex physical simulation of the entire reaction wheel system. For this reason, the first order program assumes that the deceleration of the reaction wheel is constant at the value produced by the initial torque produced at the start of deceleration, since this torque value is commonly available on reaction wheel datasheets. Fortunately, just as angular momentum can be converted to angular velocity using moment of inertia, the quantity of rotational force known as torque $T$ can be converted to angular acceleration $\alpha$ using a linear relationship with moment of inertia as well, as Equation 3 shows (12).

$$T = I \cdot \alpha \qquad \text{(Eq. 3)}$$

Once the angular velocity and deceleration have been determined, the spin-down time to slow the reaction wheel to a stop can be easily determined using Equation 4 below.

$$t = \omega / \alpha \qquad \text{(Eq. 4)}$$

## Model Code

This first order spin-down time model was programmed in MATLAB R2020a. The first section of the code, seen in Figure 1 below, prompts the user for the four inputs needed to compute all necessary values.

```
% Inputs from User (Common Values from Reaction Wheel Datasheets)
L = input('Angular Momentum Stored in Reaction Wheel (mN*m*s):\n');
T = input('Max Torque Exerted by Reaction Wheel (mN*m):\n');
r = input('Radius of Reaction Wheel (mm):\n');
m = input('Mass of Reaction Wheel (g):\n');
```

**Figure 1: Input Prompts**

The units requested for each input value were chosen because they are the most common units seen on the datasheets of CubeSat-scale reaction wheels. All necessary unit conversions are then handled automatically within the next block of code, the actual first order reaction wheel simulation math. Figure 2 shows these calculations, which use the equations and symbols discussed previously.

```
% Calculate Estimated Spin-Down Time from Inputs
% Constant torque/deceleration is assumed (ideal)
I = 0.5*(m/1000)*(r/1000)^2; % reaction wheel moment of inertia (kg/m^2)
alpha = (T/1000)/I; % reaction wheel angular acceleration (deceleration) (rad/s^2)
omega = (L/1000)/I; % reaction wheel initial angular velocity (rad/s)
t = omega/alpha; % reaction wheel spindown time (s)
```

**Figure 2: Performance Calculations**

Once all calculations are complete, the program outputs its results in two ways. First, it generates a list in the MATLAB Command Window showing the calculated moment of inertia of the reaction wheel as well as its initial angular velocity, rate of angular deceleration, and estimated spindown time. Second, it also generates a plot showing the linear decrease in angular velocity over time as the wheel slows to a stop. Figure 3 shows this portion of the code.

```
% Display Results & Plot Angular Velocity vs. Time
fprintf('======== Reaction Wheel Characteristics ========\n')
fprintf('Moment of Inertia:\t\t%.5f kg*m^2\n',I)
fprintf('Angular Acceleration:\t-%.2f rad/s^2\n',alpha)
fprintf('Init Angular Velocity:\t%.2f rad/s\n',omega)
fprintf('Spin-Down Time:\t\t\t%.2f sec\n',t)
tvector = linspace(0,t); % time vector for plotting
ovector = omega-alpha*tvector; % angular velocity vector for plotting
figure
plot(tvector,ovector)
title('Reaction Wheel Spin-Down Profile')
xlabel('Time (sec)'), ylabel('Angular Velocity (rad/s)')
```

**Figure 3: Results Listing and Plot Generation**

## Model Results

The first order spin-down time program was applied to around thirty off-the-shelf reaction wheels to observe its results. The overwhelming majority of the products tested were CubeSat reaction wheels, though a few larger reaction wheels were also tested. The average spin-down time among all of the tested products was 18 seconds with a standard deviation for the population of 16.5 seconds. No strong correlations were observed between reaction wheel radius, mass, momentum capacity, or torque capability and the spindown time predicted by the program. Table 1 summarizes the key characteristics of each reaction wheel tested and each spin-down time predicted by the program. Unfortunately, spin-down time is not a routinely provided value for off-the-shelf CubeSats, and therefore the exact accuracy level of the program has not yet been confirmed.

**Table 1: Sample Reaction Wheels and Predicted Spin-Down Times (Fastest to Slowest)**

| Product | Max Momentum (mN*m*s) | Max Torque (mN*m) | Radius (mm) | Mass (g) | Spindown Time (sec) |
|---|---|---|---|---|---|
| CubeSpace CubeWheel Small+ | 3.6 | 2.3 | 33.4 | 90 | 1.57 |
| Sinclair Rad-Hard Wheel Light | 200 | 100 | 110 | 600 | 2 |
| Blue Canyon RWP015 | 15 | 4 | 42 | 130 | 3.75 |
| Sinclair Rad-Hard Wheel Heavy | 400 | 100 | 110 | 770 | 4 |
| Hyperion RW400 | 50 | 12 | 50 | 375 | 4.17 |
| Sinclair 3mNms Picosat Wheel | 5 | 1 | 33.5 | 50 | 5 |
| NewSpace NRWA-T005 | 50 | 10 | 65 | 500 | 5 |
| NanoAvionics Reaction Wheel | 20 | 3.2 | 43.5 | 137 | 6.25 |
| Blue Canyon RWP050 | 50 | 7 | 58 | 240 | 7.14 |
| CubeSpace CubeWheel Small | 1.77 | 0.23 | 28 | 60 | 7.7 |
| Sinclair 60mNms Microsat Wheel | 180 | 20 | 65 | 226 | 9 |
| Sinclair 1Nms GEO Wheel | 1000 | 100 | 146 | 1380 | 10 |
| Comat RW20 | 20 | 2 | 35 | 150 | 10 |
| Comat RW60 | 80 | 8 | 72 | 300 | 10 |
| CubeSpace CubeWheel Medium | 10.82 | 1 | 46 | 150 | 10.82 |

| | | | | | |
|---|---|---|---|---|---|
| CubeSpace CubeWheel Large | 30.61 | 2.3 | 57 | 225 | 13.31 |
| Blue Canyon RWP100 | 100 | 7 | 70 | 330 | 14.29 |
| Blue Canyon RW1 | 1000 | 70 | 110 | 950 | 14.29 |
| Comat RW40 | 60 | 4 | 65 | 260 | 15 |
| Blue Canyon RW4 | 4000 | 250 | 170 | 3200 | 16 |
| Sinclair 10mNms Picosat Wheel | 18 | 1 | 50 | 120 | 18 |
| Sinclair 1Nms Microsat Wheel | 1000 | 50 | 140 | 970 | 20 |
| Blue Canyon RWP500 | 500 | 25 | 110 | 750 | 20 |
| ASTROFEIN RW35 | 100 | 5 | 95 | 500 | 20 |
| NewSpace NRWA-T2 | 2000 | 90 | 150 | 2200 | 22.22 |
| ASTROFEIN RW90 | 340 | 15 | 101 | 900 | 22.67 |
| ASTROFEIN RW1 | 0.58 | 0.023 | 21 | 20 | 25.22 |
| Blue Canyon RW8 | 8000 | 250 | 190 | 4400 | 32 |
| NewSpace NRWA-T065 | 650 | 20 | 102 | 1550 | 32.5 |
| ASTROFEIN RW150 | 1000 | 30 | 150 | 1500 | 33.33 |
| ASTROFEIN RW250 | 4000 | 100 | 197 | 2700 | 40 |
| Hyperion RW210 | 6 | 0.1 | 25 | 48 | 60 |
| Sinclair 30mNms Microsat Wheel | 40 | 0.5 | 50 | 185 | 80 |

## Example Output

Figures 4 and 5 below depict the input interface and output products from a typical run of the first order spin-down time estimation program. This particular run used the specifications of the Blue Canyon RWP100 CubeSat reaction wheel.

```
Angular Momentum Stored in Reaction Wheel (mN*m*s):
100
Max Torque Exerted by Reaction Wheel (mN*m):
7
Radius of Reaction Wheel (mm):
70
Mass of Reaction Wheel (g):
330
======== Reaction Wheel Characteristics ========
Moment of Inertia:       0.00081 kg*m^2
Angular Acceleration:    -8.66 rad/s^2
Init Angular Velocity:   123.69 rad/s
Spin-Down Time:          14.29 sec
```

**Figure 4: Program Inputs and Outputs**

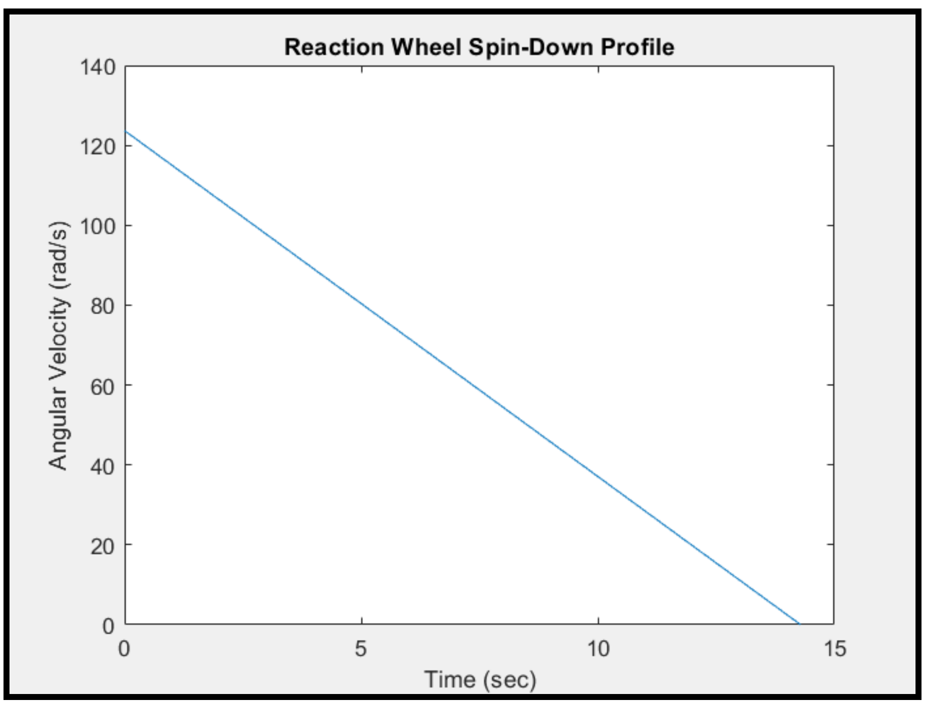


**Figure 5: First Order Spin-Down Velocity Profile Plot**

# Next Steps

One of the main limitations of the current ADACS model's approach is the limited accuracy of a first order simulation. First order dynamics produce a good estimation of reaction dynamics that are produced by a reaction wheel ADACS system, but can miss out on physical phenomena produced by higher order components of the system in order to prioritize model simplicity. The current code predicts a linear response that is roughly similar to reality without taking into account these effects, which leaves hardware testing as the most straightforward path forward to more accurate predictions.

There are a handful of ways to iterate on the current code to produce a more versatile codebase. The use of tensor values for inertia and incorporation of operations done in multiple coordinate systems through vectors could also allow for more flexibility in the kinds of ADACS units being modeled. These changes would be beneficial due to the possibility of handling angular setups where components are not lined up in the way assumed by the current model, as well as the ability to analyze compound ADAC systems with multiple reaction wheels working in tandem instead of a single unit. This ability to introduce complex inputs to the model would naturally bring with it more verification and validation requirements in order to prove its usefulness in these new applications.

Transferring the model to Simulink, MATLAB's block-based simulation program, could be another viable path forward, similar to the work described in Corey Whitcomb Crowell's thesis on the subject (13). This would allow the addition of blocks for modelling other effects such as air bearings and extended Kalman filters to produce even higher prediction accuracy compared to real-world testing. These effects would make the spin-down time output of the code produce a more tailored result than with standard effects as modeled currently. The ability to further configure and specialize the model to each individual ADACS input is a key additional goal.

Once updated to more closely match actual physical behaviors, the simulation can be made to produce response plots for comparison with the measurable responses of physical testing of a given input ADACS system. Matching responses between the model and the physical testbed would provide supporting evidence that the physical response is understood and an expected response based on vehicle dynamics.

Consideration of rejecting the use of MATLAB in favor of lower level languages that can run more quickly is also an option for improvements. Other languages, such as Python, can be more easily made available to those without academic or commercial licenses to the proprietary MATLAB software.

However, this would require a large amount of investment into creating functions and data structures to accommodate features lost by leaving the MATLAB/Simulink ecosystem. Many of these features can be retained in a nearly syntax-equivalent language called Octave. Octave is open source and widely available across computing platforms.

# Conclusion

The original intent of this project was to learn more about CubeSat Attitude Determination and Control Systems, or ADACS, of CubeSat-scale satellites, and to model their behavior as it related to the use of a CubeSat as a platform for the testing of experimental microthrusters known as MEMS Digital Thrusters (MDTs). With this in mind, research was performed and concluded that reaction wheels are the most common and widely available form of CubeSat ADACS, and that the spin-down time of an MDT-testing CubeSat would be essential to its success as a result of the need to measure an MDT firing's effect on the orientation and movement of the spacecraft. Since the spin-down time of a reaction wheel is not a value commonly provided by component manufacturers and vendors, a program was developed with the goal of estimating this key value using data that is readily available on the datasheets of CubeSat reaction wheel products. The resulting program utilizes a simplified linear model of reaction wheel torque in order to give the user a general idea of how quickly a given reaction wheel could decelerate itself to a complete stop.

While this initial tool is rooted in real reaction wheel physics, it is not able to address the full complexity of the spin-down process, which in reality is nonlinear and impacted by an array of different factors both within the reaction wheel itself and the CubeSat as a whole. With these limitations in mind, additional research was done into two possible paths for further development of the tool. The model could continue expanding to account for a greater range of physical factors and ported into Simulink to gain access to a higher degree of accuracy and the ability to simulate reaction wheel behavior in real time. Alternatively, the basic program could be further streamlined and ported to a more accessible programming language to maximize its usefulness as a rapid prototyping and planning tool more suited for the early stages of CubeSat design.

# References


1. Sinclair Interplanetary. *Reaction wheels.*

    http://www.sinclairinterplanetary.com/reactionwheels

2. NewSpace Systems. 2021 *Reaction Wheel.*

    https://www.newspacesystems.com/wp-content/uploads/2021/02/NewSpace-Reaction-Wheel_2021-10b.pdf

3. Nano avionics. *CubeSat Reaction Wheels Control System SatBus 4RWO.*

    https://nanoavionics.com/cubesat-components/cubesat-reaction-wheels-control-system-satbus-4rw/

4. Hyperion Technologies. *Products: Attitude Control.*

    https://hyperiontechnologies.nl/product-categorie/adc/

5. Cube Space. *CubeWheel: Small Satellite Reaction Wheels.*

    https://www.cubespace.co.za/products/adcs-components/cubewheel/#cubewheel-specifications

6. Comat. *Reaction Wheel Technology: For small satellites.*

    https://comat-agora.com/products/reaction-wheels-technology

7. Blue Canyon Technologies. *Components: Reaction Wheels and CMGs.*

    https://www.bluecanyontech.com/components

8. Astrofein. *Products: Position control - Reaction wheels.*

    https://www.astrofein.com/astro-und-feinwerktechnik-adlershof/produkte/

9. Lumen Learning. *Conservation of Angular Momentum.*

https://courses.lumenlearning.com/boundless-physics/chapter/conservation-of-angular-momentum/#:~:text=Angular%20momentum%20is%20defined%2C%20mathematically,all%20directions%20after%20a%20collision.

10. Brittanica. *Moment of Inertia.* https://www.britannica.com/science/moment-of-inertia
11. Hyperphysics. *Moment of Inertia: Thin Disk.* http://hyperphysics.phy-astr.gsu.edu/hbase/tdisc.html
12. Lumen Learning. *Torque and Angular Acceleration.* https://courses.lumenlearning.com/boundless-physics/chapter/torque-and-angular-acceleration/
13. Corey Whitcomb Crowell. *Development and Analysis of a Small Satellite Attitude Determination and Control System Testbed*

# Appendix

**Full Text of First Order Spin-Down Time Estimation Program (MATLAB R2020a)**

```
%% Honors Capstone Spring 2021
% Barkley Hunter, Joshua Conway, Jacob Kincheloe
% Project Director: Matt Turner

clear,clc

%% First-Order Estimation of CubeSat ADACS Spin-Down Time
% Inputs from User (Common Values from Reaction Wheel Datasheets)
L = input('Angular Momentum Stored in Reaction Wheel (mN*m*s):\n');
T = input('Max Torque Exerted by Reaction Wheel (mN*m):\n');
r = input('Radius of Reaction Wheel (mm):\n');
m = input('Mass of Reaction Wheel (g):\n');

% Calculate Estimated Spin-Down Time from Inputs
% Constant torque/deceleration is assumed (ideal)
I = 0.5*(m/1000)*(r/1000)^2; % reaction wheel moment of inertia (kg/m^2)
alpha = (T/1000)/I; % reaction wheel angular acceleration (deceleration) (rad/s^2)
omega = (L/1000)/I; % reaction wheel initial angular velocity (rad/s)
t = omega/alpha; % reaction wheel spindown time (s)

% Display Results & Plot Angular Velocity vs. Time
```

```
fprintf('======== Reaction Wheel Characteristics ========\n')
fprintf('Moment of Inertia:\t\t%.5f kg*m^2\n',I)
fprintf('Angular Acceleration:\t-%.2f rad/s^2\n',alpha)
fprintf('Init Angular Velocity:\t%.2f rad/s\n',omega)
fprintf('Spin-Down Time:\t\t\t%.2f sec\n',t)
tvector = linspace(0,t); % time vector for plotting
ovector = omega-alpha*tvector; % angular velocity vector for plotting
Figure
plot(tvector,ovector)
title('Reaction Wheel Spin-Down Profile')
xlabel('Time (sec)'), ylabel('Angular Velocity (rad/s)')
```