Modeling and Simulation of Quad Rotor and EDF-8 Ducted Fan Platforms

Undergraduate research fall 2014

Pachu chembukave

2014

**Executive Summary**

This semester I set out to simulate the behavior of both the Quad Rotor and the EDF-8 Ducted Fan systems. I believed that by simulating these system’s we could get better feedback as to how our control algorithms interacted with the mathematical models they were based on. For the Quad Rotor System I used the mathematical model from Bouabdallah’s paper (with some vehicle specific parameters changed to match our Quad Rotor platform) (1). For the ducted fan I used the truncated linear dynamics from Avid’s EDF-8 Linearized Dynamics (2). I chose MATLAB as my development environment for its rapid workflow, great graphing /UI tools, and extensibility to C++. After developing models for both the ducted fan and the quad rotor, I set out to make the simulation easy to use. I added a front end GUI with the ability to switch vehicles, set the simulation parameters, graph the state variables of the vehicles and save as well as load data. I also implemented some sample algorithms to check the behavior of the models and possibly stabilize them, but I was unable to see many positive results from these algorithms. Currently the system only works with MATLAB functions as control algorithms. I wished to extend this capability to accept C++ functions, but it was a goal I was unable to meet. This tool can save data sets for further analysis, allow for quick testing with different time or state settings, and visualize current or previous data sets. I believe it can be used to eventually develop a well-functioning and robust control algorithm. Full source code: <https://github.com/pachuc/QuadSim>

**Introduction and Background**

My project is based entirely around mathematically modeling both the Quad Rotor and the EDF-8 platforms. For these models I consulted two papers: Samir Bouabdallah’s Design and Control of Quadrotors with Application to Autonomous Flying (1), and AVID LLC’s EDF-8 Linearized Dynamics (2). Below I will detail how these two systems are represented in their respective models.

Both models are based around a vehicle state (composed of various vehicle parameters), a control input and a time step, used to calculate a change in state. For the Quad Rotor the state vector consisted of twelve items: X, Y and Z positions; Phi, Theta, and Psi angles; X, Y, and Z velocities; and finally Phi, Theta, and Psi angular velocities. The Quad Rotor model also assumed the following details about the system:

* The structure is supposed rigid.
* The structure is supposed symmetrical.
* The Center of Gravity and the body fixed frame origin are assumed to coincide.
* The propellers are supposed rigid.
* Thrust and drag are proportional to the square of the propeller’s speed.

These assumptions matched our actual Quad Rotor system quite well, and so the model presented in this paper seemed extremely pertinent. After researching different Quad Rotor simulations and models, I came upon a Simulink representation of the exact model detailed in Bouabdallah’s paper (3). Using this model as a basis in combination with the actual paper, I was able to derive a MATLAB function to compute the next state of the Quad Rotor System. The system boils down to equations that are able to compute linear accelerations for the Quad Rotor in X, Y, and Z, and angular accelerations in Phi, Theta, and Psi. These accelerations can then be integrated over the time step in order to produce the linear and angular positions as well as velocities for the next state.

For the EDF-8, I used the truncated linear dynamics from AVID’s paper. These truncated dynamics are designed only to track the orientation and body frame angular rates. This model ignores the translational velocities of the ducted fan as they are parameters that are hard to estimate given current sensor data. Thus, the truncated linear dynamics presents a state vector composed of six entries: Phi, Theta, Psi, P, Q and R. The model takes a state vector and a control vector, and produces a state change based around a reference state (the hover state of the EDF-8). This change can then be integrated over the time step to produce the next state of the ducted fan.

The control functions I developed to interact with the models take a state vector, desired state vector, and time step as parameters. They then produce an appropriate length control vector, which can be used in the vehicle model to compute state change. These two components together make up the crux of the simulation.

**Design Documentation**

The first step in creating the simulation was designing MATLAB classes to represent each vehicle. The classes both shared a few common variables: time step, run time, control function, starting state, and desired state. Both classes share common functions for setting these parameters in the simulation.

The state vector for the Quad Rotor is 12 elements long (A1) and the control vector its control function must produce is 4 elements long (A2). This control vector represents the inputs to the four rotors in RPM. The Quad Rotor also uses numerous constants, most of which are the same as in Bouabdallah’s paper (1), but some have been modified to match our Quad Rotor platform (A3).

The state vector for the ducted fan is 6 elements (B1) and the control vector is 5 elements (B2). The control vector for the ducted fan is the four vein deflections and the fan speed in RPM. The truncated dynamics (2) also use an A matrix (B3) a B matrix (B4) and a nominal state and control pair (B5) to produce the state change.

I used these two classes (QuadRotor.m and EDF8.m) as the backbone of my simulation. These backend classes are manipulated by the GUI to set the appropriate parameters and generate the appropriate simulation data. The GUI can then parse the data and generate graphs for each element of the vehicles state over time. The data sets can also be saved as .mat files and loaded into the GUI for viewing through the same format. The results matrix is NxM dimensions where N = the number of time steps in the simulation and M = the number of elements in the state vector of the vehicle. Thus, each row represents one variable’s value over time.

**Results**

While the models and simulation in general were implemented successfully, the results produced by the control algorithms were not satisfactory in the least. As can be seen from the included data sets in appendix D, the states of the vehicles were not able to stabilize beyond a few short seconds. For the Quad Rotor I used and LQR controller with a K matrix derived through the MATLAB LQR function. The Q, R, A and B matrices used for this generation are listed in appendix C. For the ducted fan I merely provided the hover state input, while starting the vehicle in hover state to see if the vehicle would be stable. Unfortunately, this was not the case.

**Conclusion**

In conclusion, I was able to produce a system to accurately measure and record the state changes in the model of each vehicle; however, I was unable to use it to produce a working control algorithm. I do believe, however, that this tool will be useful for analysis of both systems, and testing potential control solutions. While currently the system only accepts MATLAB functions as input, it can easily be extended for C++, as there are many functions out there for converting C++ functions to MATLAB or directly executing C++ code. The data sets can also be easily passed to other MATLAB functions or manipulated separately to be graphed and analyzed in different ways. This project helped me gain a better understanding of the mathematical models representing each system, and I believe it will help others to better understand the dynamics and responses of each system.

APPENDIX

**Appendix A: Quad Rotor System**

**A1. State Vector**

[X Y Z Phi Theta Psi X’ Y’ Z’ Phi’ Theta’ Psi’]

**A2. Control Vector**

[U1 U2 U3 U4]

**A3. Quad Rotor Constants**

|  |  |
| --- | --- |
| Variable | Value |
| Moment of Inertia about the X axis | 2.297e-2 |
| Moment of Inertia about the Y axis | 2.297e-2 |
| Moment of Inertia about the Z axis | 4.935e-2 |
| Total Rotational Moment of Inertia around the propeller axis | 6.5e-5 |
| Mass in Kg | 1.89 |
| Thrust Factor | 1 |
| Drag Factor | 0.3048 |
| Length of Arms (Distance from Rotor to center) | 0.23 |

**Appendix B: Ducted Fan System**

**B1. The State Vector**

[Phi Theta Psi P Q R]

**B2. Control Vector**

[d1 d2 d3 d4 dt]

**B3. The A matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | -1.3103e-16 | 2.4495e-08 |
| 0 | 0 | 0 | 0 | 1 | 5.3493e-09 |
| 0 | 0 | 0 | 0 | -5.3493e-09 | 1 |
| 0 | 0 | 0 | 0 | 7.1782 | 0 |
| 0 | 0 | 0 | -6.7417 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1.7951e-16 | 0 |

**B4. The B matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| -0.43279 | -0.43279 | -6.2763e-18 | 6.2763e-18 | -7.3552e-12 |
| 0 | 0 | -0.40647 | -0.40647 | -1.7118e-10 |
| 0.18763 | -0.18763 | -0.18763 | 0.18763 | -5.7012e-05 |

**B5. Sate and Input Values for Hover State**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phi | Theta | Psi | P | Q | R | D1 | D2 | D3 | D4 | dt |
| 1.23e-7 | 1.95e-7 | -0.108 | 0 | 0 | 0 | 0.32 | 0.32 | 0.32 | 0.32 | 8301.46 |

**Appendix C: Control Matrices**

**C1. The Q Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |

**C2. The R Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| 100000 | 0 | 0 | 0 |
| 0 | 100000 | 0 | 0 |
| 0 | 0 | 100000 | 0 |
| 0 | 0 | 0 | 100000 |

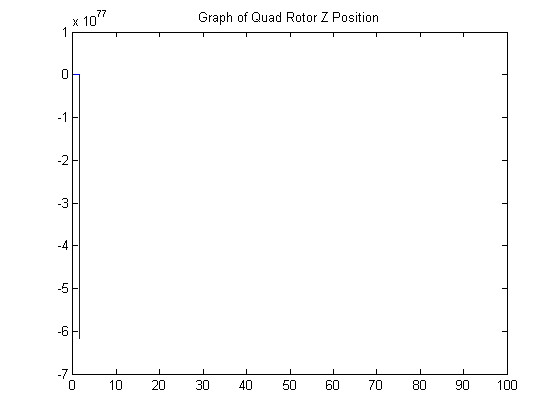
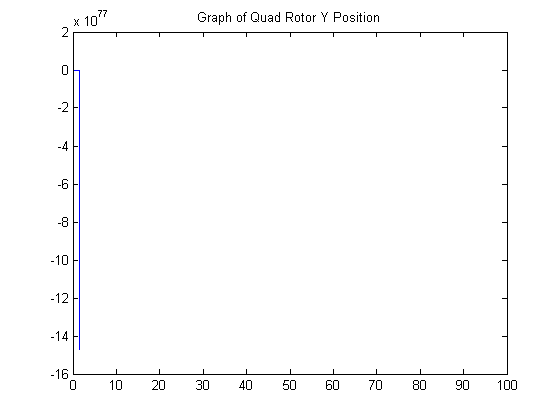
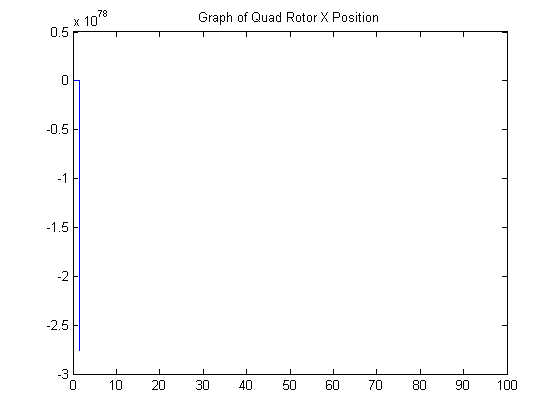
**C3. The A Matrix**

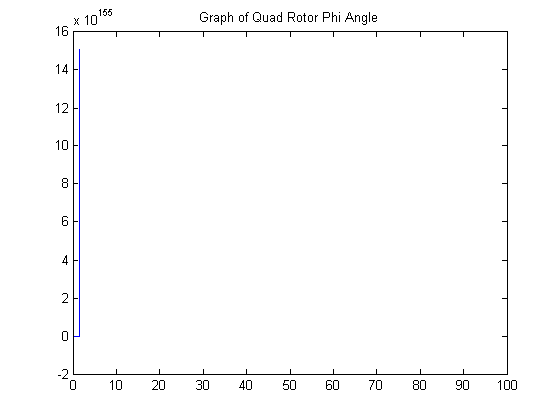
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | -9.81 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 9.81 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

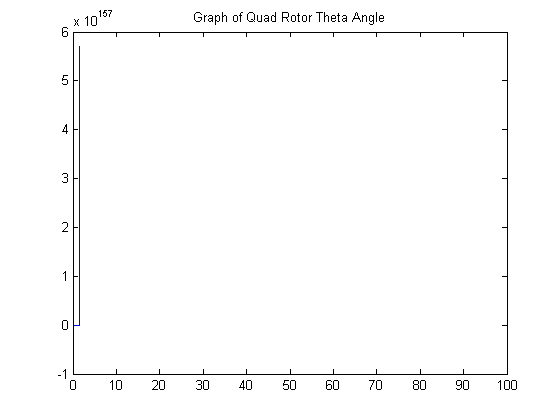
**C4. The B Matrix**

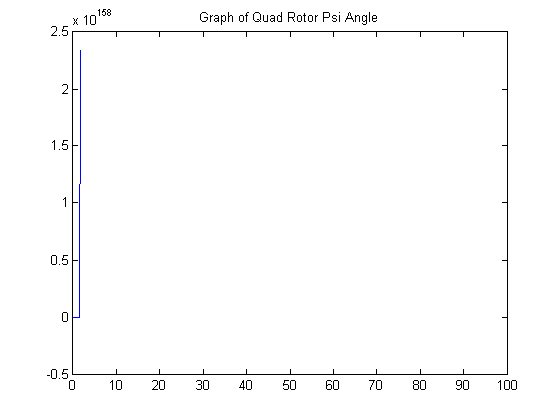
|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| -0.5291 | -0.5291 | -0.5291 | -0.5291 |
| 0 | -13.269 | 0 | 13.269 |
| 13.269 | 0 | -13.269 | 0 |
| 0.0013171 | -0.0013171 | 0.0013171 | -0.0013171 |

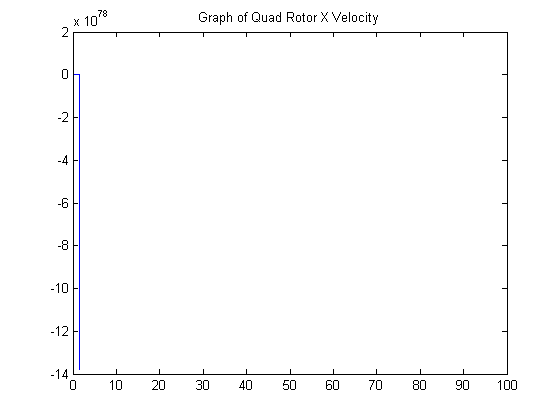
**Appendix D: Graphs**

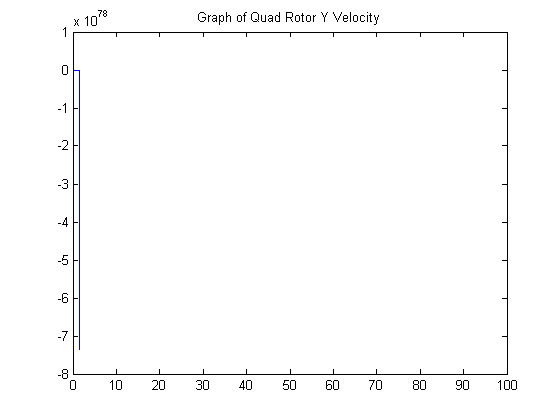


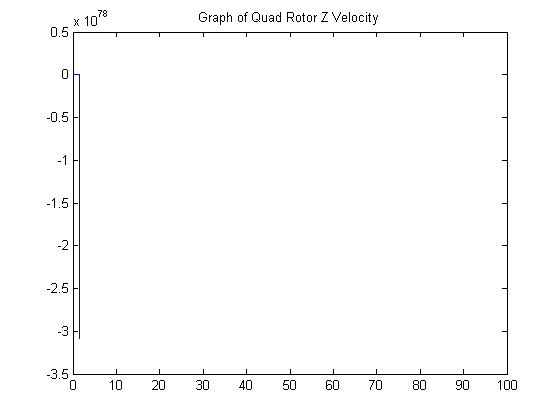


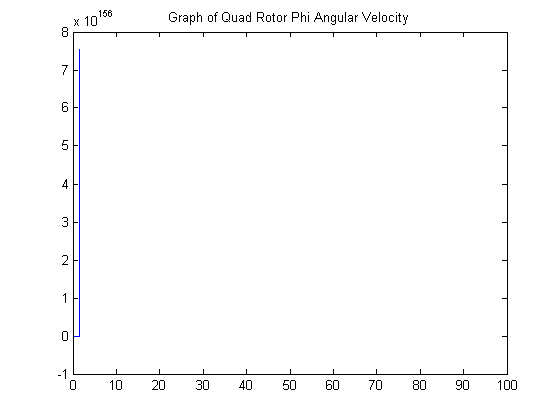


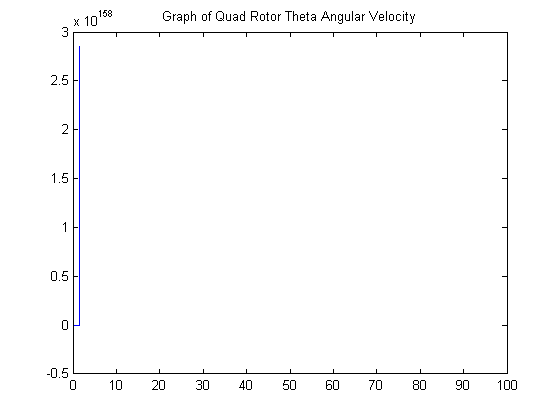


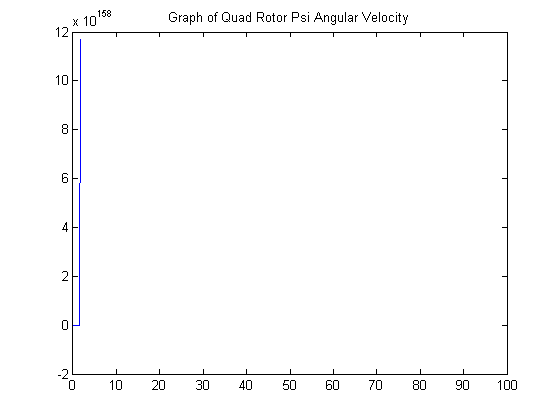


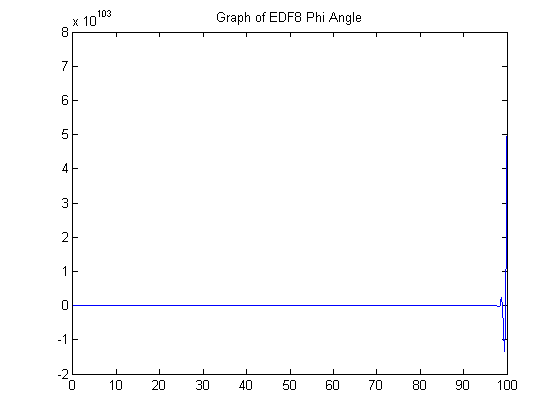


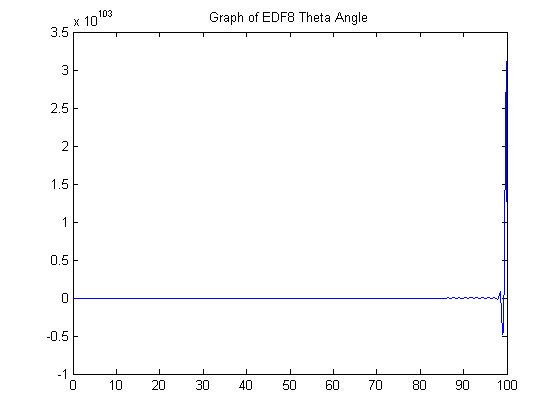


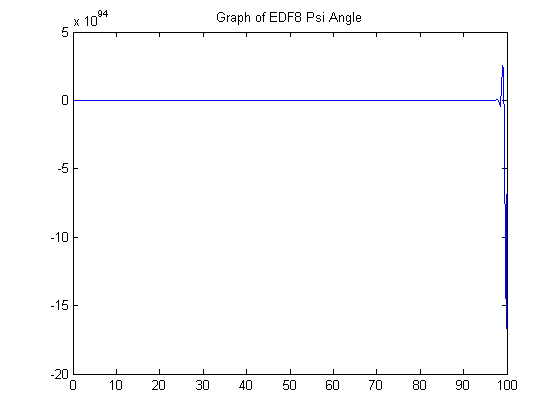


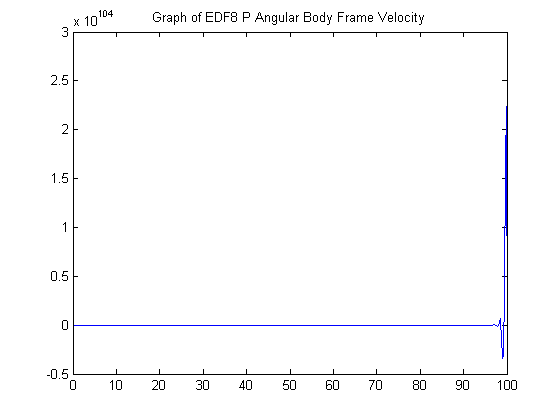


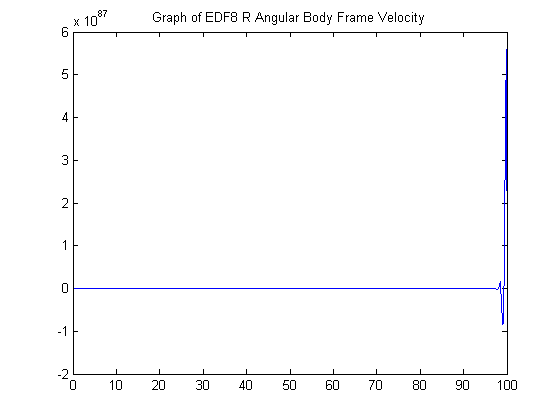
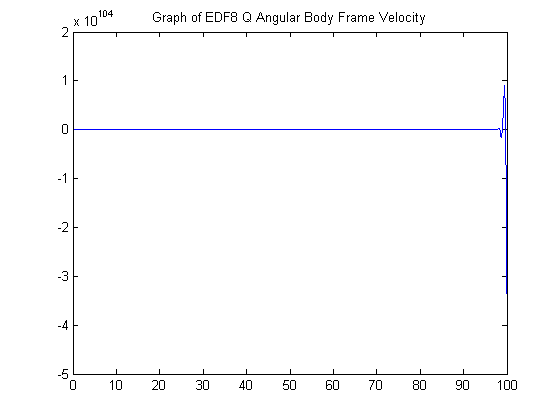












Works Cited

1. Bouabdallah, Samir. "Design and Control of Quadrotors with Application to Autonomous Flying." (n.d.): n. pag. Print.
2. *EDF-8 Linearized Dynamics*. Tech. Blacksburg: AVID LLC, 2013. Print.
3. Razzak, Abdel. *PD Control Quadrotor - Simulink*. Computer software. *Matlab Centeral File Exchange*. Mathworks, 05 Apr. 2013. Web. 16 Dec. 2014.