MAV SIMULATION IN SCILAB FOR HARDWARE-IN-LOOP TESTING

AE 497 B.Tech. Project Stage I

By

Saurav Agarwal

06001011

Under the guidance of

Dr. Hemendra Arya

Department of Aerospace Engineering,

Indian Institute of Technology, Bombay

November, 2009

Declaration of Academic Integrity

I declare that this report represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my report. I understand that any violation of the above will be cause for disciplinary action as per the rules and regulations of the institute.

Saurav Agarwal

(06001011)

Certificate

Certified that this B.Tech. Project titled "MAV Simulation in Scilab for Haridware-in-Loop Testing" by "Saurav Agarwal" is approved by me for submission. Certified further that, to the best of my knowledge, the report represents work carried out by the student.

Date: Signature and Name of Guide

Abstract

Building a Micro Aerial Vehicle (MAV) requires the testing of embedded systems before the aircraft can be flown in the real environment. Thus, simulators are implemented to help in design and testing of navigation, guidance and control laws, design and development of various interfaces e.g. GPS interface, filters etc. and identifying faults in the system. Using Hardware in Loop Simulations, an on-board computer can be directly switched to an MAV, without any modification in hardware and software. Thus, this project aims at building requisite software and hardware capabilities to enable hardware-in-loop simulations of Micro Aerial Vehicles. This report discusses the simulation software that has been built using Scilab, and the future course of this project.

Table of Contents

List of Figures

Nomenclature

MAV	Micro Aerial Vehicle
UAV	Unmanned Aerial Vehicle
GPS	Global Positioning System
DOF	Degree of Freedom
HILS	Hardware in Loop Simulation
I	Moment of inertia
u	Velocity in body x axis
v	Velocity in body y axis
w	Velocity in body z axis
p	Angular velocity about body x axis
q	Angular velocity about body y axis
r	Angular velocity about body z axis
φ,θ,ψ	Euler angles
x_e	Displacement along Earth fixed x axis
y_e	Displacement along Earth fixed y axis
H	Altitude
RK	Range-Kutta

Chapter 1

Introduction

The term **micro air vehicle** (**MAV**) refers to a type of unmanned air vehicle (UAV) that is remotely controlled. Today's MAVs are significantly smaller than those previously developed, with target dimensions reaching a maximum of approximately 15 centimetres (six inches). Development of insect-size aircraft is reportedly expected in the near future. Potential military use is one of the driving factors of development, although MAVs are also being used commercially and in scientific, police and mapping applications. Another promising area is remote observation of hazardous environments that are inaccessible to ground vehicles. Because these aircraft are often in the same size range as radio-controlled models, they are increasingly within the reach of amateurs, who are making their own MAVs for aerial robotics contests and aerial photography.

1.1 MAV Configurations

Three types of MAVs are under investigation: airplane-like fixed wing models, bird- or insect- like ornithopter (flapping wing) models, and helicopter-like rotary wing models. Each type has different advantages and disadvantages, different scenarios may call for different types of MAV. Fixed-wing MAVs can currently achieve higher efficiency and longer flight times, so are well suited to tasks that require extended loitering times, but are generally unable to enter buildings, as they cannot hover or make the tight turns required. Rotary-wings allow hovering and movement in any direction, at the cost of shorter flight time. Flapping wings offer the most potential for miniaturization and manoeuvrability, but are currently far inferior to fixed and rotary wing MAVs.

1.2 Uses

The largest use of MAVs is in military applications. Currently, military MAVs perform reconnaissance missions. They can be conveniently carried by soldiers in backpacks and can be quickly deployed for battlefield surveillance. An MAV can either be remotely piloted or can be completely autonomous depending on mission requirements as they can easily be programmed to follow a set path using GPS waypoints.

The RQ-11 is a perfect example of a military MAV.

- **RQ-11 Raven**: The **AeroVironment** RQ-11 Raven is a remote-controlled miniature unmanned aerial vehicle (or MUAV) used by the U.S. military and its allies. The craft is launched by hand and powered by an electric motor. The plane can fly up to 6.2 miles (10 km) up to altitudes of 1,000 feet (305 m) above ground level (AGL), and 15,000 feet mean sea level (MSL), at flying speed of 28-60 mph (45-97 km/h). The Raven can be either remotely controlled from the ground station or fly completely autonomous missions using GPS waypoint navigation. The UAV can be ordered to immediately return to its launch point simply by pressing a single command button. Standard mission payloads include charge-coupled device colour video and an infrared night vision camera.

Figure 1.1: RQ-11 being hand launched [1]

Figure 1.2: RQ-11 Raven [1]

MAVs are also used in a small but growing number of commercial applications, such as police surveillance, border patrol, agriculture, fire fighting, search and rescue operations etc.

1.3 Designing an MAV

Designing an MAV requires the capability to simulate the performance of an embedded control system using hardware in loop simulation techniques. This enables us to save a lot of money because an aircraft need not be flown which eliminates the risk of crashing and loss of expensive on-board equipment. Hence, computer modelling requires the programming of flight mechanics equations and the implementation of numerical methods for computing. Further, to test the response of our hardware, we need a test bed for HILS for real time testing. However, currently there is no fast and simple setup available in our lab to carry out direct integration of hardware and software for carrying out simulations.

1.3.1 Current Modelling Tools Available

1. [2]**AeroSim**: The AeroSim blockset is a Matlab/Simulink block library which provides components for rapid development of nonlinear 6-DOF aircraft dynamic models. In addition to aircraft dynamics the blockset also includes environment models such as standard atmosphere, background wind, turbulence, and Earth Models (geoid reference, gravity and magnetic field). Its key features are:

    - Full 6-DOF simulation of nonlinear aircraft dynamics
    - Visual output to Microsoft Flight Simulator and FlightGear Flight Simulator
    - Complete aircraft models that can be customized via parameter files
    - Ability to automatically generate C code from Simulink aircraft models using Real-Time Workshop.

2. [3]**Flight Gear Flight Simulator:** FlightGear is an open-source project. It is an interactive flight simulation tools which gives the opportunity to train pilots in simulated aircrafts. It can also be used to model the performance of aircrafts. It offers a variety of flight dynamics models.

    - **JSBSim**: JSBSim is a generic, 6DoF flight dynamics model for simulating the motion of flight vehicles. It is written in C++. JSBSim can be run in a standalone mode for batch runs, or it can be the driver for a larger simulation program that includes a visuals subsystem (such as FlightGear.) In both cases, aircraft are modeled in an XML configuration file, where the mass properties, aerodynamic and flight control properties are all defined.

- **YASim**: This FDM is an integrated part of FlightGear and uses a different approach than JSBSim by simulating the effect of the airflow on the different parts of an aircraft. The advantage of this approach is that it is possible to perform the simulation based on geometry and mass information combined with more commonly available performance numbers for an aircraft. This allows for quickly constructing a plausibly behaving aircraft that matches published performance numbers without requiring all the traditional aerodynamic test data.

- **UIUC**: This model is based on LaRCsim originally written by the NASA. UIUC extends the code by allowing aircraft configuration files instead and by adding code for simulation of aircraft under icing conditions. UIUC (like JSBSim) uses lookup tables to retrieve the component aerodynamic force and moment coefficients for an aircraft and then uses these coefficients to calculate the sum of the forces and moments acting on the aircraft.

1.3.2 Drawbacks

The major drawback of using Matlab based tools is the exorbitant cost involved in procuring Matlab. Also, the Aerosim block set is available free only for academic purposes, hence this project has implications beyond the scope of our department. On the other hand Flight Gear is an open source tool but it does not provide a method to integrate the flight dynamics model directly with embedded systems for HILS. Hence the need arises to setup a facility where we can smoothly integrate external hardware with PCs for simulations using free/low cost solutions such as open source software. This would allow the user to conduct experiments using a simple code written in an intuitive programming language such as Scilab instead of complex coding in C. Hence, we switch to Scilab as it removes these basic drawbacks.

1.4 [4]Scilab

Scilab is an open source scientific software package similar in ways to Matlab for numerical computations providing a powerful open computing tool for engineers and scientists. Its key features are:

- 2-D and 3-D graphics, animation
- Linear algebra, sparse matrices
- Polynomials and rational functions
- Interpolation, approximation
- Simulation: ODE solver and DAE solver
- Scicos: a hybrid dynamic systems modeler and simulator
- Classic and robust control, LMI optimization
- Differentiable and non-differentiable optimization
- Signal processing
- Metanet: graphs and networks
- Parallel Scilab
- Statistics
- Interface with Computer Algebra: Maple package for Scilab code generation
- Interface with Fortran, Tcl/Tk, C, C++, Java, LabVIEW

Scilab has another key feature; it allows direct integration with National Instruments Hardware, eliminating the need for writing complex control code for the microcontroller.

Aim of Report

The aim of this report is to describe the non linear mathematical model used for simulating the 6 Degree of Freedom motion of an aircraft and the code used to implement this. Further, the report show will discuss a few simulation results without any hardware in the loop done in Scilab.

Report Layout

Chapter 2 deals in-depth with the theoretical foundation of the mathematical model. Chapter 3 explains the code and methodology that has been implemented and Chapter 4 discusses the results of a few simulations. Chapter 5 will describe the future work that needs to be done.

Chapter 2

The Non-Linear Aircraft Model

2.1 General Equations of Motion

The aircraft equations of motion are derived from basic Newtonian mechanics. The general force and moment equations for a rigid body are:

$$\mathrm{F} = \mathrm{m}\left(\frac{\partial \mathrm{V}}{\partial \mathrm{t}} + \Omega \text{ X } \mathrm{V}\right) \tag{2.1}$$

$$\mathrm{M} = \frac{\partial (\mathrm{I} \cdot \Omega)}{\partial \mathrm{t}} + \Omega \text{ X } (\mathrm{I} \cdot \Omega) \tag{2.2}$$

These equations express the motions of a rigid body relatively to an inertial reference frame (see appendix B for the derivation of these rigid body equations). $\mathbf{V} = [u\ v\ w]^T$ is the velocity vector at the center of gravity, $\mathbf{\Omega} = [\ p\ q\ r\ ]^T$ is the angular velocity vector about the c.g. $\mathbf{F} = [\ Fx\ Fy\ Fz\ ]^T$ is the *total* external force vector, and $\mathbf{M} = [L\ M\ N\ ]^T$ is the *total* external moment vector. **I** is the inertia tensor of the rigid body, which is defined as:

$$\mathrm{I} = \begin{bmatrix} \mathrm{Ixx} & -\mathrm{Jxy} & -\mathrm{Jxz} \\ -\mathrm{Jyx} & \mathrm{Iyy} & -\mathrm{Jyz} \\ -\mathrm{Jzx} & -\mathrm{Jzy} & \mathrm{Izz} \end{bmatrix} \tag{2.3}$$

The coefficients from this tensor are the moments and products of inertia of the rigid body. If the frame of reference is fixed to the vehicle these values are constant, regardless of the attitude of the vehicle. In order to make equations (3.1) and (3.2) usable for control system design and analysis, simulation purposes, system identification, etc., these equations need to be re-written in non-linear *state-space* format. Moving the time-derivatives of the linear and angular velocities to the left hand side of the equations yields:

$$\frac{\partial V}{\partial t} = \frac{F}{m} - \Omega \times V$$

(2.4)

$$\frac{\partial \Omega}{\partial t} = \frac{1}{I}(M - \Omega \times I \cdot \Omega)$$

(2.5)

Together these dynamic equations form a state-space system which is valid for any rigid body, e.g. aircraft, spacecraft, road-vehicles, or ships. These equations obviously form the core of the simulation model. The body-axes components of linear and rotational velocities can be regarded as the *state* variables from this model, while the body-axes components of the external forces and moments are the *input* variables of these equations.

This clear picture is complicated by the fact that the external forces and moments themselves depend upon the motion variables of the aircraft. In other words: the state variables themselves must be coupled back to the force and moment equations. Although this makes the equations more complex, it is still possible to combine these equations in a non-linear state space system.

$$\dot{\mathbf{x}} = \mathbf{f}\,(\mathbf{x},\mathbf{F}_{tot}(t),\mathbf{M}_{tot}(t)) \tag{2.6}$$

with,

$$\mathbf{F}_{tot} = \mathbf{g}1\,(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t), t) \tag{2.7}$$

$$\mathbf{M}_{tot} = \mathbf{g}2\,(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t), t) \tag{2.8}$$

The state vector **x** obviously contains linear and angular velocity components, i.e. the elements from **V** and **Ω**. In addition to these variables, information about the spatial orientation of the aircraft is needed for finding the gravitational force contributions. Furthermore, the altitude of the aircraft is needed for the computation of aerodynamic and engine forces which are both affected by changes in air density that depend upon the altitude of the aircraft. The coordinates of the aircraft with respect to the Earth are not needed for solving the equations of motion, but they are useful for other purposes, such as the assessment of the flight-path for certain maneuvers. Therefore, the complete state vector **x** will consist of twelve elements: three linear velocities, three angular velocities, three Euler angles which define the attitude of the aircraft relatively to the Earth, two coordinates and the altitude which define the position of the aircraft, relatively to the Earth.

$$\mathbf{x} = [u\ v\ w\ p\ q\ r\ \phi\ \theta\ \psi\ x_e\ y_e\ H\ ]^T \quad (2.9)$$

2.2 Aerodynamics Forces and Moments

The aerodynamic forces are modelled as functions of total velocity and geometric parameters by finding the aerodynamic co-efficients of forces and moments. The co-efficients are taken as sum of individual components.

Cx, is the total aerodynamic co-efficient in the body x axis

Cy, is the total aerodynamic co-efficient in the body x axis

Cz, is the total aerodynamic co-efficient in the body x axis

Cl, is the total aerodynamic co-efficient in the body x axis

Cm, is the total aerodynamic co-efficient in the body x axis

Cn, is the total aerodynamic co-efficient in the body x axis

Xbar , Total aerodynamic force acting in body x axis = Cx qbar S_{ref}

Ybar , Total aerodynamic force acting in body y axis = Cy qbar S_{ref}

Zbar , Total aerodynamic force acting in body z axis = Cz qbar S_{ref}

Lbar , Total aerodynamic moment acting about body x axis = Cl qbar S_{ref} b_{ref}

M , Total aerodynamic moment acting about body y axis = Cm qbar S_{ref} c_{ref}

N, Total aerodynamic moment acting about body z axis = Cn qbar S_{ref} b_{ref}

Where,

qbar = $\frac{1}{2} \rho V^2_T$

ρ = local atmospheric density

S_{ref} = Wing reference surface area

b_{ref} = Wing reference span

c_{ref} = Wing reference chord

2.3 The Flat Earth, Body Axes 6-DOF Equations [5]

2.3.1 Force Equations

$$\frac{du}{dt} = rv - qw - g\sin(\theta) + \frac{(Thrust - Xbar)}{m} \tag{2.10}$$

$$\frac{dv}{dt} = -ru + pw + g\sin(\varphi)\cos(\theta) - \frac{Ybar}{m} \tag{2.11}$$

$$\frac{dw}{dt} = qu - pv + g\cos\varphi\cos\theta - \frac{Zbar}{m} \tag{2.12}$$

2.3.2 Kinematic Equations

$$\frac{d\Phi}{dt} = p + \tan\theta\,(q\sin\varphi + r\cos\varphi) \tag{2.13}$$

$$\frac{d\theta}{dt} = q\cos\varphi - r\sin\varphi \tag{2.14}$$

$$\frac{d\psi}{dt} = \frac{(q\sin\varphi + r\cos\varphi)}{\cos\theta} \tag{2.15}$$

2.3.3 Moment Equations

$$\frac{dp}{dt} = (C_1 r + C_2 p)\,q + C_3\,Lbar + C_4 N \tag{2.16}$$

$$\frac{dq}{dt} = C_5 pr - C_6\,(p^2 - r^2) + C_7 M \tag{2.17}$$

$$\frac{dr}{dt} = (C_8 p - C_2 r)\,q + C_4 Lbar + C_9 N \tag{2.18}$$

The moments of inertia have been used by converting to simpler constants,

Gamma = $I_{xx}\ I_{zz}$ - $(I_{xz}\ I_{xz})$

$C_1 = ((I_{yy} - I_{zz})\ I_{zz} - (I_{xz} * I_{xz}))$ / Gamma

$C_2 = ((I_{xx} - I_{yy} + I_{zz})\ I_{xz})$ / Gamma

$C_3 = I_{zz}$ / Gamma

$C_4 = I_{xz}$ / Gamma

$C_5 = (I_{zz} - I_{xx}) / I_{yy}$

$C_6 = I_{xz} / I_{yy}$

$C_7 = 1 / I_{yy}$

$C_8 = (I_{xx}\ (I_{xx} - I_{yy}) + I_{xz} * I_{xz})$ / Gamma

$C_9 = I_{xx}$ / Gamma

2.3.4 Navigation Equations

$$\frac{dxe}{dt} = u \cos\theta \cos\psi + v(-\cos\varphi \sin\psi + \sin\varphi \sin\theta \cos\psi) + w(\sin\varphi \sin\psi + \cos\varphi \sin\theta \cos\psi) \quad (2.19)$$

$$\frac{dye}{dt} = u \cos\theta \sin\psi + v(\cos\phi \cos\psi + \sin\phi \sin\theta \sin\psi) + w(-\sin\phi \cos\psi + \cos\Phi \sin\theta \sin\psi) \quad (2.20)$$

$$\frac{dH}{dt} = u \sin\theta - v \sin\varphi \cos\theta - w \cos\varphi \cos\theta \quad (2.21)$$

2.3.5 Total Body Components

Total Velocity $V_{T} = \sqrt{(u^2 + v^2 + w^2)}$ (2.22)

Sideslip $\beta = \frac{v}{VT}$ (2.23)

Angle of Attack $= \frac{u}{w}$ (2.24)

Chapter 3

Implementation of the Model

3.1 Functions

The code utilises some in-built and user some defined functions to calculate various parameters. They are briefly described here.

3.1.1 Atmosphere Model

The atmosphere model uses the total velocity and altitude as arguments to calculate the local density and thus qbar. It makes use of the ideal gas equation and a constant tempereature lapse rate to carry out its calculation.

Equations:

$$T = To - (H*Lapserate) \tag{3.1}$$

$$P = Po\left(\frac{To}{T}\right)^{\frac{g}{Lapserate*287}} \tag{3.2}$$

$$\rho = \frac{P}{287*T} \tag{3.3}$$

$$qbar = 0.5\,\rho\,V^2_T \tag{3.4}$$

Where,

H = Altitude

T = Local temperature

To = Temperature at sea level

P = Local atmospheric pressure

Po = Pressure at sea level

3.1.2 Engine Model

The engine model takes the throttle, total velocity and local density as arguments to calculate the engine thrust. This function has been designed in the controls lab to suit our specific need. It does not represent an accurate model of the actual engine.

Equations:

$$J = V_T / (0.3\ dth) \tag{3.5}$$

$$\text{Thrust} = 33\ (\frac{4}{\pi^2})\ \rho\ (dth)^2\ (0.3)^4\ (-0.0948\ J^2 + 0.058\ J + 0.0761\ ) \tag{3.6}$$

Where,

dth = throttle setting (Rotations per second)

ρ = atmospheric density

V_T = total velocity

3.1.3 Aerodynamic Lookup Tables

Since this model makes use of non-linear aerodynamic data, the aerodynamic co-efficients have to be taken from lookup tables. The data for these tables was taken from the [6]University of Illinois website for the **Pioneer** UAV. The program makes use of inbuilt spline interpolation functions because of their accuracy to interpolate from available values and obtain the co-efficient values at particular angles. The two inbuilt functions that have been used are "**interp**" and "**interp2d**".

For each aerodynamic co-efficient separate evaluation functions have been written.

3.1.4 Range-Kutta Algorithm Implementation

To integrate the derivate equations and obtain state values at each time step, the RK-4 numerical integration scheme has been implemented.

Method:

$$x' = f\,(t, x) \tag{3.7}$$

With initial condition $x(0) = x_0$.

Suppose that x_n is the value of the variable at time t_n. The Runge-Kutta formula takes x_n and t_n and calculates an approximation for x_{n+1} at a brief time later, t_n+h. It uses a weighted average of approximated values of $f(t, x)$ at several times within the interval (t_n, t_n+h).

The formula is given by,

$$x_{n+1} = x_n + {}^{h}\!/_{6}\,(a + 2\,b + 2\,c + d) \text{ where } a = f(t_n, x_n) \tag{3.8}$$

$$b = f(t_n + {}^{h}\!/_{2},\, x_n + {}^{h}\!/_{2}\, a) \tag{3.9}$$

$$c = f(t_n + {}^{h}\!/_{2},\, x_n + {}^{h}\!/_{2}\, b) \tag{3.10}$$

$$d = f(t_n + h,\, x_n + h\, c) \tag{3.11}$$

3.1.5 State Derivate Evaluation

The purpose of this function is to solve and compute the state derivatives at each time step using the Flat Earth equations described in the previous chapter.

3.2 Methodology for Implementation

The code utilises the above described functions and pre-set parameters taken for the **Pioneer** UAV such as,

- Mass = 190 kg
- I_{xx} = 47.22 kg-m^2
- I_{yy} = 90.84 kg-m^2
- I_{zz} = 111.48 kg-m^2
- I_{xz} = -6.64 kg-m^2
- S_{ref} = 2.826 m^2
- b_{ref} = 5.15 m
- c_{ref} = 0.55 m
- x_{cg} = 1 m, CG location relative to wing leading edge, expressed as a fraction of aerodynamic chord length

The user can set the control inputs i.e. Throttle, Control surface deflections (rudder, aileron, elevator) and the initial state of the aircraft. The code then calculates the state at each time step using by using a user set time step and total time for simulation.

Steps:

- The loop runs for the desired time and at each iteration calls the RK-4 evaluation function
- The RK-4 evaluator calls the state evaluation function and uses it to integrate the initial state to the next
- The state evaluator calls the aerodynamic lookup tables, engine model and atmosphere to calculate the aerodynamic co-efficients, thrust, qbar and density respectively
- The state variables at each step are stored in arrays for plotting

Chapter 4

Simulation Results

*Resolution of simulation images have slightly been compromised despite best efforts

4.1 Trim Conditions

After a few simulations and by trial and error method, trim condition was seen to be achieved with the following initial conditions:

- Altitude: 997
- Throttle: 188
- Elevator deflection: 1.7 degrees
- u = 34.4799 m/s
- w = 3.265 m/s
- θ = 0.09 rad
- Rest all parameters were set to zero

We can see from the figures that the body angular rates are settling down to zero. Moreover, total velocity as well as altitude is nearly constant.

Shown here are the simulation results:

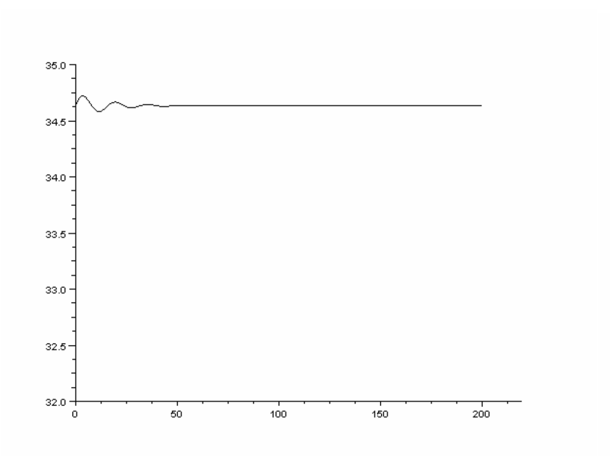


Figure 4.1: Total Velocity vs. Time

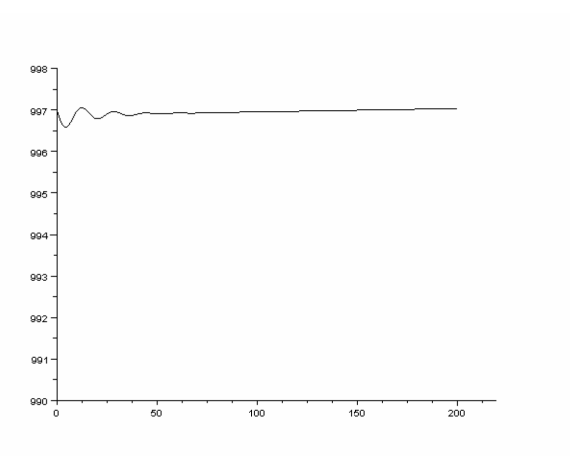


Figure 4.2: Altitude vs. Time

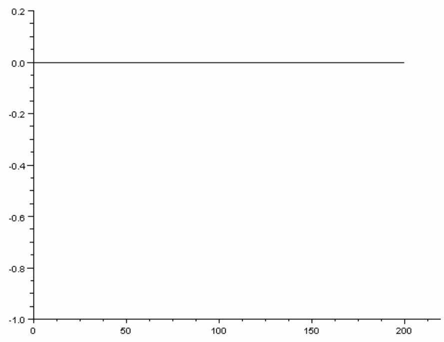


Figure 4.3: p vs. Time

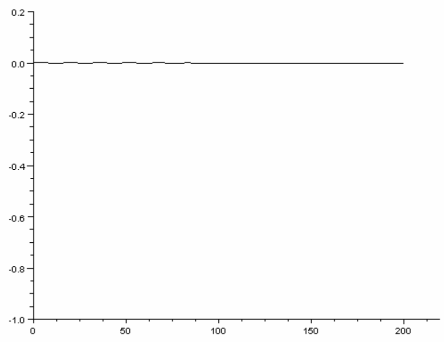


Figure 4.4: q vs. Time

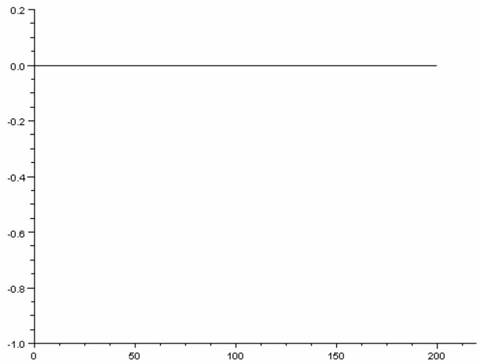


Figure 4.5: r vs. time

4.2 Elevator Impulse Input

Starting from trim conditions, the elevator deflection was increased to +5 degree for 2 seconds at t = 5 s and then brought back to 1.7 at t = 7 s. We see that aircraft is disturbed but tends to settle down to a steady state trim value. This shows that the aircraft has pitch stability.

Shown here are the simulation results:

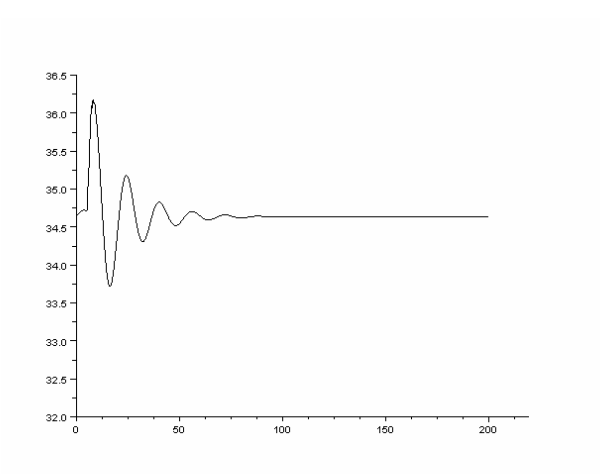


Figure 4.6: Total Velocity vs. Time

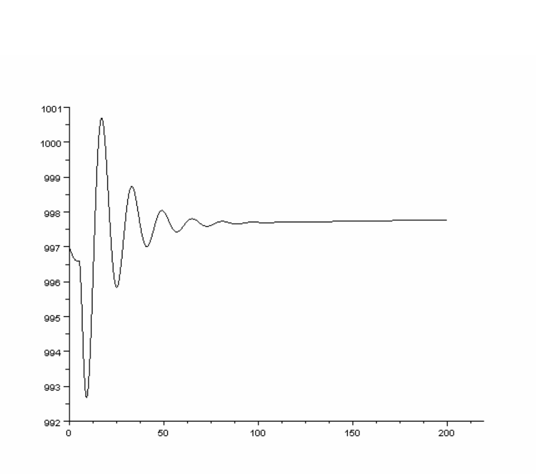


Figure 4.7: Altitude vs. Time

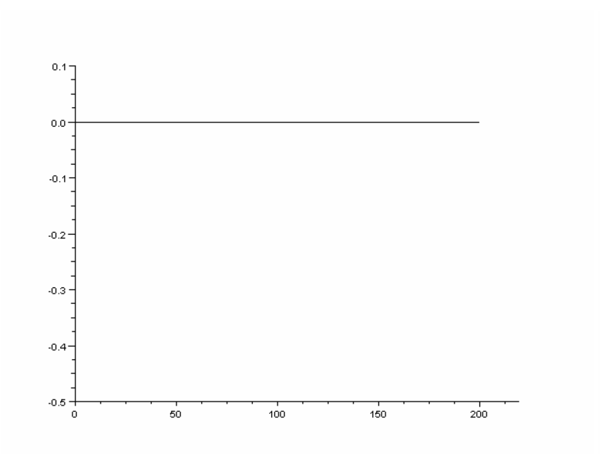


Figure 4.8: p vs. Time

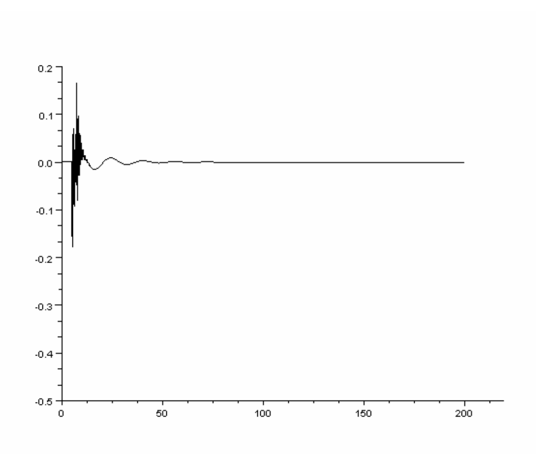


Figure 4.9: q vs. Time

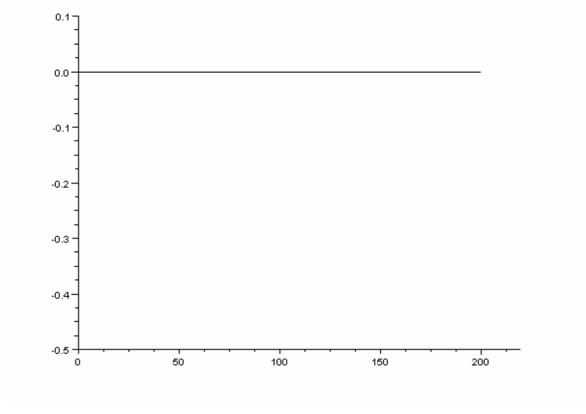


Figure 4.10: r vs. Time

4.3 Aileron Impulse Input

Starting from trim conditions, the aileron deflection was made +2 degree for 2 seconds at t = 5s and then brought back to 0 at t = 7 s. We see that aircraft is disturbed but tends to return to zero, however it starts diverging around t =50 s which is unexpected. It is expected that the aircraft will settle down to a zero roll rate, hence this anomaly remains to be verified. Whether it is an error in the aero data or implementation.

Shown here are the simulation results:

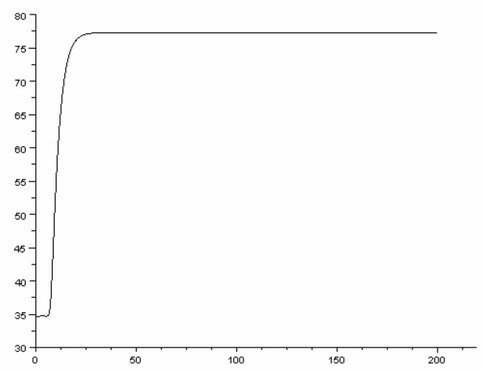


Figure 4.11: Total Velocity vs. Time

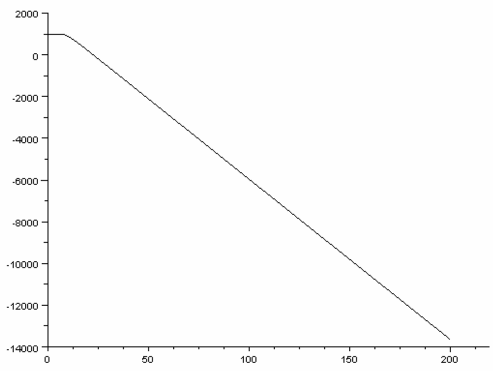


Figure 4.12: Altitude vs. Time

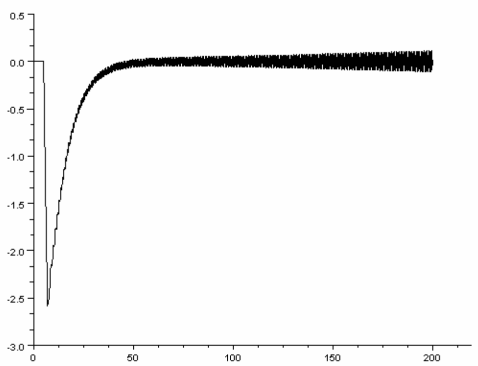


Figure 4.13: p vs. Time

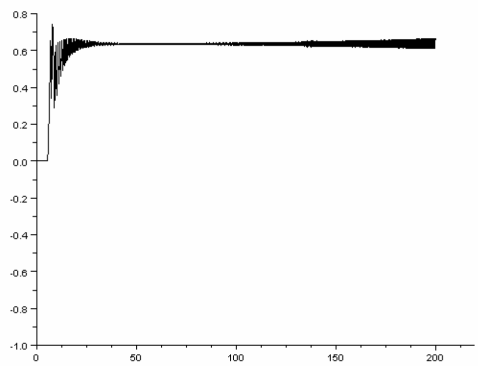


Figure 4.14: q vs. Time

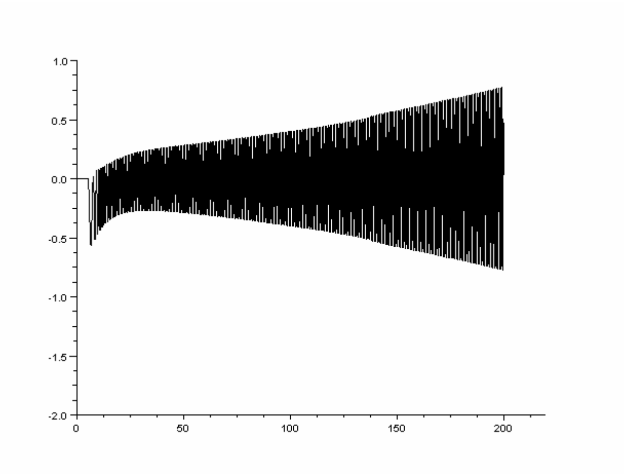


Figure 4.15: r vs. Time

4.4 Rudder Impulse Input

Starting from trim conditions, the rudder deflection was made +5 degree for 2 seconds at t = 5s and then brought back to 0 at t = 7 s. We see that aircraft is disturbed but tends to settle down to original state. This aircraft displays yaw stability, by coming back to nearly its original orientation.

Shown here are the simulation results:

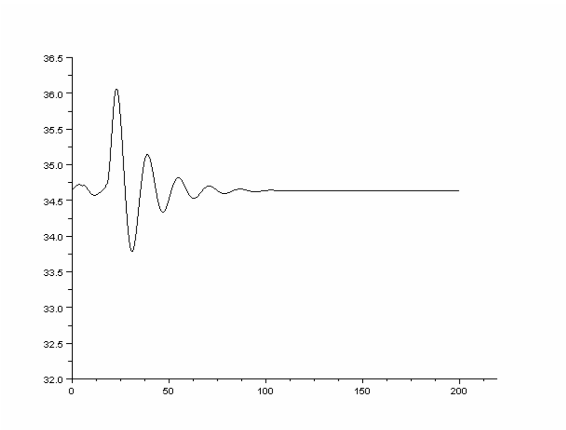


Figure 4.16: Total Velocity vs. Time

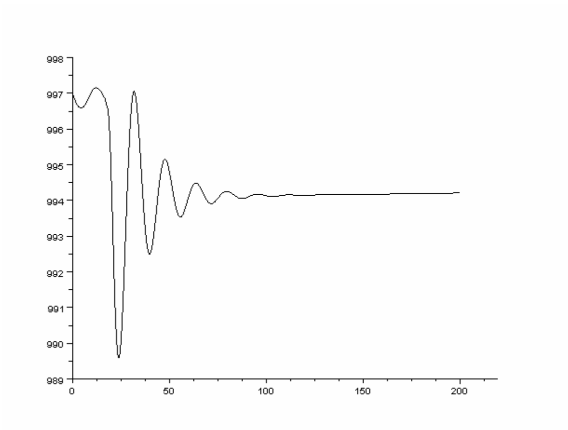


Figure 4.17: Altitude vs. Time

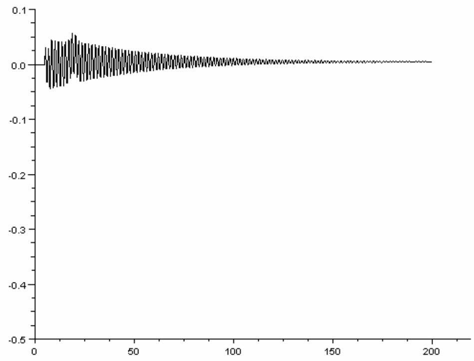


Figure 4.18: p vs. Time

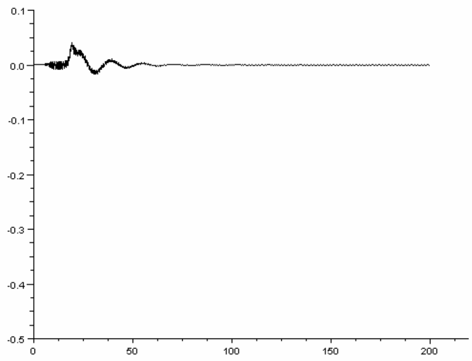


Figure 4.19: q vs. Time

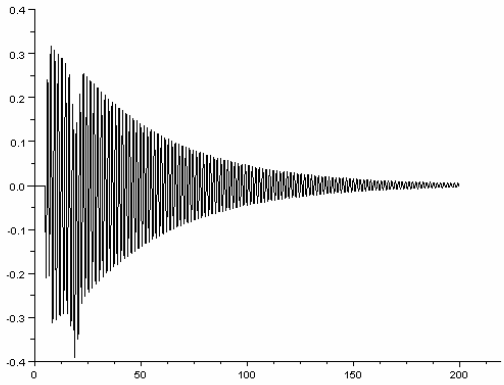
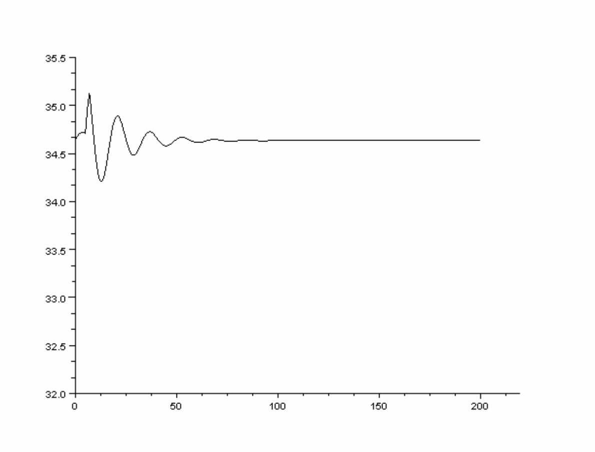


Figure 4.20: r vs. Time

4.5 Throttle Impulse Input

Starting from trim conditions, the throttle was increased to 200 for 2 seconds at t = 5s and then brought back to 0 at t = 7 s. We see that aircraft is tends to climb, but settles down to constant state after the disturbance is removed.

Shown here are the simulation results:



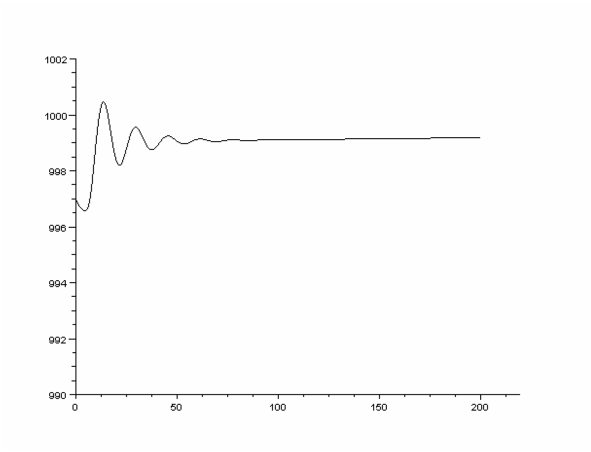
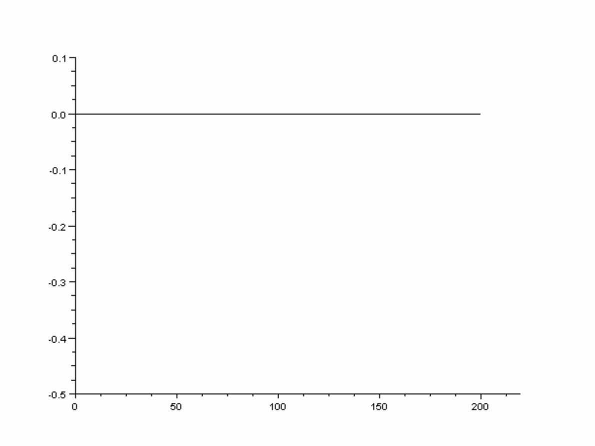


Figure 4.21: Total Velocity vs. Time

Figure 4.22: Altitude vs. Time



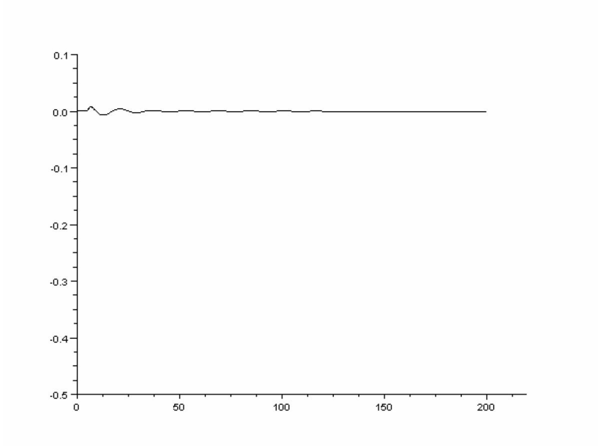


Figure 4.23: p vs. Time

Figure 4.24: q vs. Time

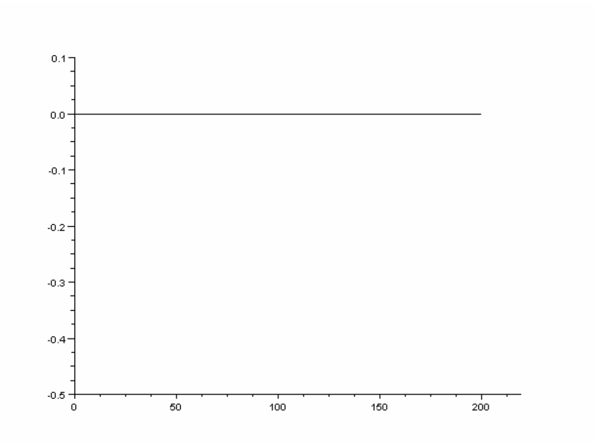


Figure 4.25: r vs. Time

Chapter 5

Future Work

At the first stage of this project, the 6 DOF aircraft model as described by [5]Stevens and Lewis, has been implemented and the results have been compared with simulations carried out in existing software built and verified in the department. All results are satisfactory except the aileron disturbance simulation which is yet to be verified. The future course of this project will be to analyse freely available tools which allow for integration of Scilab codes with hardware for real time data acquisition. Presently, National Instruments offer LabVIEW as a package for real time data handling and HILS testing. The interface between Scilab and LabVIEW is provided through a Scilab script node that you can include text-based Scilab programming with (Scilab macro language) in Virtual Instruments you create with LabVIEW. However, since LabView is a proprietary tool it is quite costly to procure. On the other hand, EtherLab is an open source tool combining hardware and software for test and automation purposes. EtherLab is not a product but a technique built from reliable and well known components and also new technical approaches. It works as a Real Time kernel module attached to the open source operating system Linux communicating with peripherals devices by a special Ethernet technology, known as EtherCAT.

The next step would be to choose the optimum tool for implementation of this integration and then develop a fully functional test bed.

References

[1] "RQ-11 Raven", Wikipedia Webpage
http://en.wikipedia.org/wiki/RQ-11_Raven

[2] "AeroSim Blockset", Unmanned Dynamics Products
http://www.u-dynamics.com/aerosim/

[3] "Flight Gear Flight Simulator"
http://www.flightgear.org/

[4] "Scilab"
http://www.scilab.org

[5] Stevens B.L. and Lewis F.L., *Aircraft Control and Simulation*, John Wiley & Sons, Inc., 1992.

[6] "Aircraft Dynamics Models for Use with FlightGear", *UIUC Applied Aerodynamics Group*
http://www.ae.illinois.edu/m-selig/apasim/Aircraft-uiuc.html

Acknowledgments

It is with a great sense of gratitude that I acknowledge the support and guidance given by Prof. Hemendra Arya during the course of the first stage of this project and making it an invaluable learning experience.

Date: April 13, 2009 Saurav Agarwal