MAV SIMULATION IN SCILAB FOR HARDWARE-IN-LOOP TESTING

AE 497 B.Tech. Project Stage II

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. After evaluating the feasibility of EtherCAT, RTAI-Lab tool chain and a Comedi Toolbox called Scicos-Comedi based solutions, it was decided to go ahead with Scicos-Comedi. This project deals with the design of an HIL system using Scilab/Scicos as the primary software for the graphical design of the control structure. The Comedi toolkit, a software tool which provides Scicos blocks for real time DAQ applications is used as an add-on library for Scicos. A linux based system patched with RTAI kernel and comedi drivers for real time interfacing is used.

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
CACSDS	Computer Aided Control System Design Software
RTAI	Real Time Application Interface
A/D	Analog to Digital
D/A	Digital to Analog
DAQ	Data Acquisition

Chapter 1

Introduction

Desktop PC based simulation, which includes designing a mathematical model for a physical system and executing it on a digital computer, has been the primary method for analyzing and studying the behavior of dynamic systems. Software tools such as computer aided control system design software (CACSDS) and programming tools are essential in the field of control systems design, modeling, and simulation. Simulating events in a time frame in which they would naturally occur is known as real-time simulation. The importance of timing accuracy of these simulations strongly depends on the application but may be crucial in ensuring intended system performance. The main objective of this project is to design an easy to implement HIL test bed using GUI based tools.

Software tools supporting real-time environments are required to achieve the above mentioned objective and to analyze its real world performance. Scilab/Scicos is one such tool that provides mathematical programming and graphical modeling platform for designing and simulating systems. An advantage of using Scicos is that makes the task of designing the control flow very easy compared to writing a complicated C code for real time execution since such a code requires extensive use of external dependencies (i.e. Comedi and RTAI functions) making the code writing process tedious. A package called Scicos-Comedi adds a palette of Comedi I/O devices to Scicos which provides a custom library of Scicos blocks. This can be used for general Linux based real-time applications.

1.1 The methodologies evaluated

1.1.1 The RTAI-Lab tool chain [1]

The RTAI-Lab tool chain represents an alternative to the commercial software listed below:

- Scilab/Scicos ⟶ Mathworks' MatlabR /SimulinkR
- Comedi ⟶ drivers supplied by signal acquisition hardware vendors, but most vendors don't supply Linux drivers
- RTAI ⟶ LynxOSR from LynuxworksR , MontaVista LinuxR , QNXR, VxWorksR , etc.

- RTAI-Lib $\longrightarrow$ Mathworks' Real-Time WorkshopR
- xrtailab $\longrightarrow$ LabView

The RTAI-Lab tool-chain is based on:

- Scilab/Scicos. Scilab is an open source CACSD software for numerical computation. Scilab includes Scicos, a block diagram editor that can be used to create simulations and automatically generate and compile code.
- Comedi drivers Comedi provides the drivers, library functions, and an API to interact with signal acquisition hardware. Hundreds of devices are supported.
- RTAI The Real-Time Application Interface (RTAI) is distributed as a package with a patch to apply to the Linux kernel. RTAI inserts a sub-kernel where prioritized, hard real-time tasks can run. FIFOs and shared memory can be used to transfer data between real-time and user space processes.
- RTAI-Lib. RTAI-Lib is a palette of Scicos blocks that let you design block diagrams with sensors and actuators. It provides an interface to RTAI and signal acquisition hardware. Block diagrams that use RTAI-Lib can be compiled into RTAI execuatble software. It is included in the RTAI package.
- xrtailab. xrtailab is an oscilloscope-like software that can connect to your real-time executables. It lets you visually monitor signals and real-time events using gauge, scope, and LED mock-ups. Xrtailab also lets you adjust parameters of the real-time executable while it runs. It is also part of RTAI. This is not used since the GPS position as well as IMU measurements are sent to a GUI which gives the real time position of the a/c on a map as well as its orientation with respect to an artificial horizon.

1.1.2 EtherLab [2]

The EtherLAB Scicos Toolbox is a HIL and Realtime Toolbox for data acquisition, data logging und system controlling. It's functionality is related to the RTAILAB- and HART-Toolbox. Many thanks for the inspiration. The signal IO (analog, digital, PT, Counter, SSI, CAN, Profibus) is based on the EtherCAT Fieldbus (a decentral high perfomance bus based on Ethernet frames). The toolbox uses a GPL and linux based fieldbus master driver. Other IO-Subsystems are possible. Signal IO can be used during simulation like HIL-Comedi blocks. The toolbox includes a Code generator for the Linux Userspace target. A preempt rt kernel allows realtime cycle frequencies up to or greater 20kHz (depends on the hardware). Datavisualisation and -logging is available after codegeneration.

1.1.3 Scicos-Comedi [3]

Scicos-Comedi provides a set of Scicos blocks which can be used for communicating with DAQ devices using Comedi driver libraries. It is a form of ScicosHIL wherein it is possible to use the GUI of Scicos to control the plant. It allows the implementation of scilab code which is not available in the other two.

1.2 Why the Scicos-Comedi Toolbox

In the first stage of this project two solutions were proposed for setting up the simulator.

1. RTAI-Lab as an add-on package for Scilab/Scicos
2. EtherCAT interface using Etherlab as an add-on package for Scilab/Scicos

First, it was decided to try RTAI-Lab over the EtherCAT based solution primarily because of zero cost of implementation of the required software, since all software used for RTAI-Lab are free and Open Source and requires no extra hardware need. On the other hand, EtherCAT requires proprietary I/0 interfacing boards to communicate with microcontrollers through the Ethernet port of a PC. These boards are expensive and not freely available in India. RTAI-Lab however presents the problem that it doesn’t allow the use scilab language for the computational function

1.3 Simulation Methodology

The HIL simulator design takes advantage of the simple and easy to use Graphical programmable block available in Scicos. Coupled with Scicos-Comedi, it provides powerful customisable blocks which can be used to interface with external hardware using Comedi drivers. A brief flow of the methodology:

1. Program the computational function in scilab which calculates the instantaneous state of the aircraft based on previous state and control inputs.
2. Design the Scicos block Diagram with the required input and output blocks available in Comedi palette.

3. Test the open loop realtime to match the output on CRO and output on screen.
4. Run HIL.

Aim of Report

The aim of this report is to describe the Scicos-Comedi tool chain GUI based Hardware in Loop simulation technique implemented for MAV testing.

Report Layout

Chapter 2 deals in-depth with the software. Chapter 3 explains the Scicos structure and methodology that has been implemented. Chapter 4 discusses the corrections made in the code to correct the errors in stage 1. Chapter 5 deals with the testing in Realtime. Chapter 6 presents the conclusions.

Chapter 2

The Softwares

2.1 Scicos [3]

Scicos is a graphical dynamical system modeller and simulator developed in the Metalau project at INRIA, Paris-Rocquencourt center. With Scicos, user can create block diagrams to model and simulate the dynamics of hybrid dynamical systems and compile models into executable code. Scicos is used for signal processing, systems control, queuing systems, and to study physical and biological systems.

2.1.1 Scicos Features

1. GUI and graphics

- GUI to model dynamical systems as block diagrams
- Palettes of standard blocks One can:
    - Modify existing open source blocks
    - Program new blocks in C, Fortran (dynamic link), or Scilab
    - Extend current palettes
- Animation block
- Interfaces to Tcl/Tk widgets to generate custom GUIs

2. Hierarchical structures

- Model components can be aggregated into Super Blocks to create a hierarchical block diagram
- Converts independent Super Blocks into C code

3. Simulations

- Batch mode simulations from Scilab

- Powerful formalism to model hybrid systems, i.e., possibility to combine continuous and discrete-time behaviors in the same model
- Choice of solvers:
    - Ordinary Differential Equations (ODE) with SUNDIALS Solver CVODE.
    - Differential Algebraic Equations (DAE) with SUNDIALS Solver IDA.
- "Implicit" blocks based on differential algebraic equations can be created and processed by the DASKR solver

4. Compilation, debugging

- Compiles and runs simulations on block diagrams
- Performs partial recompilations to save time
- Interfaces with external programs and operating systems
- Adjustable debugging levels
- Breakpoint placement for debugging

5. Code generation

- Generates C code
- Generates real-time code for RTAI (using RTAI-Lab)

6.Interfaces

- Can be piloted from Scilab programs
- Data and state variables can easily be exchanged with external programs
- Interfaces to digital acquisition cards with Scicos-HIL (Linux and Windows) and Scicos-RTAI (Linux RTAI)
- Can integrate Modelica objects, for example to model electrical and hydraulic circuits

2.2 Comedi [4]

Comedi is a *free software* project that develops drivers, tools, and libraries for various forms of *data acquisition*: reading and writing of analog signals; reading and writing of digital inputs/outputs; pulse and frequency counting; pulse generation; reading encoders; etc. The project's source code is distributed in two packages, comedi and comedilib, and provides several Linux *kernel modules* and a *user space* library:

- **Comedi** is a collection of drivers for a variety of common data acquisition plug-in boards (which are called "devices" in Comedi terminology). The drivers are implemented as the combination of (i) one single core Linux kernel module (called "comedi") providing common functionality, and (ii) individual low-level driver modules for each device.
- **Comedilib** is a separately distributed package containing a user-space library that provides a developer-friendly interface to the Comedi devices. Included in the *Comedilib* package are documentation, configuration and calibration utilities, and demonstration programs.
- **Kcomedilib** is a Linux kernel module (distributed with the comedi package) that provides the same interface as *comedilib* in kernel space, and suitable for *real-time* tasks. It is effectively a "kernel library" for using Comedi from real-time tasks.

Comedi works with standard Linux kernels, but also with its real-time extensions RTAI and RTLinux/GPL.

2.3 Real Time Application Interface [RTAI] [5]

The RTAI project began at the “Dipartimento di Ingegneria Aerospaziale del Politecnico di Milano” (DIAPM) in 1996/97. It stemmed from the need of making available a tool to support a varied set of internal research activities related to advanced active controls for generic aeroservoelastic systems, including large space structures, acoustics and flexible manipulators. Its aim was to make it possible their development, implementation and testing on standard 32 bits personal computers (PC) and data acquisition cards, by using high level language programming tools, so that anybody, including graduating students, could proceed to their implementation with a relative ease in building it all.
Budget constraints and the satisfaction of some DIAPM researchers in using Linux as a general purpose operating system brought the idea of adding hard real time capabilities to it. RTAI is integrated into Linux through a text file containing a set of changes to its kernel source code, known as a patch, and a series of add on programs expanding Linux to hard real time. As such it is bound to be a GPLed licensed [3] FOSS, as Linux is. So RTAI has been

freely available on the net from its very beginning. In 1999, after the appearance of the 2.4.xx release of Linux, RTAI began being relatively widely known and after some time it became an FOSS development effort with a team of developers worldwide.

2.3.1 How RTAI works

RTAI expands Linux to hard real time, to that end RTAI patches the Linux kernel by installing a generic Real Time Hardware Abstraction Layer (RTHAL). RTHAL performs three primary functions:

a) Gathers all the pointers to the time critical kernel internal data and functions into a single structure, to allow the easy trapping of all the kernel functionalities that are important for real time applications, so that they can be dynamically substituted by RTAI when hard real time is needed. Generally speaking such kernel functionalities include all functions and data structures required to manipulate: the hard interrupt flag, external interrupts and internal traps/faults, the system call, programmable interrupt controllers and hard timers. The related objects are substituted by pointers that can be changed dynamically.

b) Reworks the related Linux functions, data structures and macros to make it possible to use them to initialize RTHAL pointers for normal Linux operations.

c) Changes Linux to use what pointed in RTHAL for its operation.

2.4 Scicos-Comedi Palette

This palette is the set of Comedi I/0 blocks available. These blocks can directly be copied from the palette and used in the Scicos diagram.

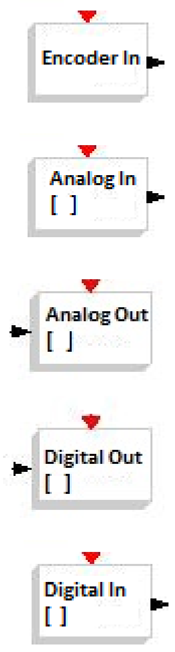

Figure 2.1: The Comedi palette

1. Encoder In: Read the data from encoder
2. Analog Input: Reads data from specified channel in ADC device
3. Analog Out: Writes data to specified channel in DAC device
4. Digital Input: Reads data from specified channel in serial port
3. Digital Out: Writes data to specified channel in serial port

Chapter 3

Scicos Block Diagram and Setup

3.1 Control Loop Structure in Scicos

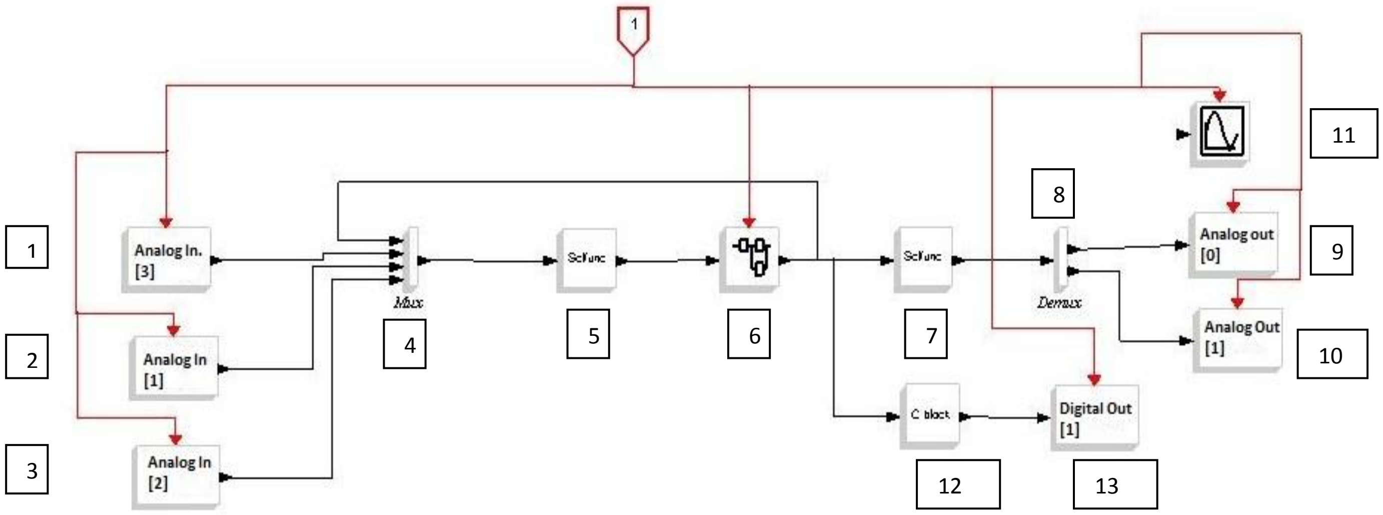

Figure 3.1: Scicos control loop structure

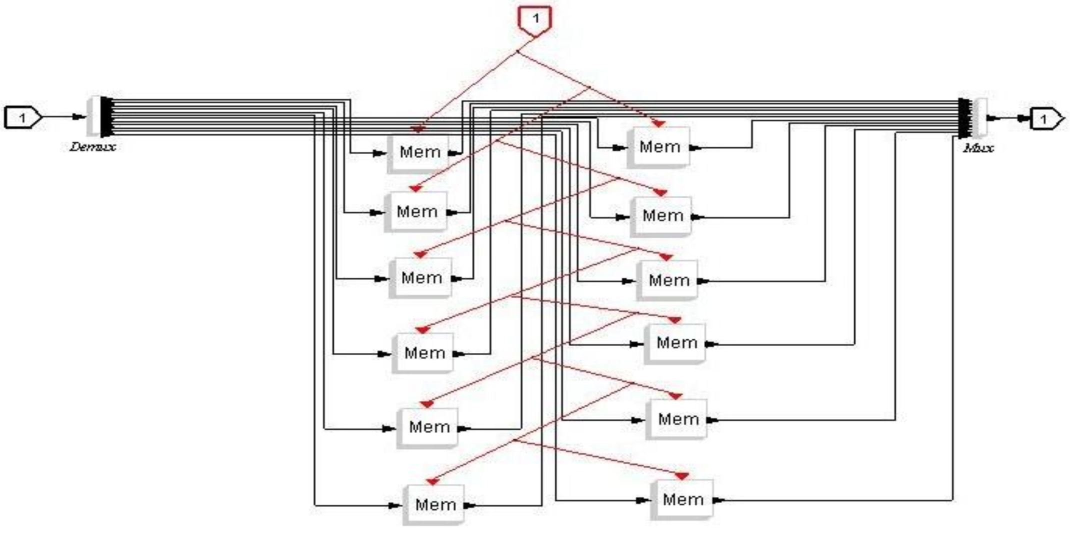

Figure 3.2: Memory hold superblock

3.1.1 Diagram Components

The diagram comprises of various blocks which are explained as follows:

1. Analog In [1] : This block reads voltage coming from aileron servo pot
2. Analog In [2] : This block reads voltage coming from throttle servo pot
3. Analog In [3] : This block reads voltage coming from elevator servo pot
4. Mux: This branching block takes in multiple inputs and combines them into 1 signal as in a Multiplexer
5. Scifunc computational block
6. : This a computational block with the function coded in scilab language. It takes in the previous state and control inputs and calculates the new state
7. Memory superblock: This block holds the value of the 12 parameters i.e $\mathbf{x} = [u\ v\ w\ p\ q\ r\ \phi\ \theta\ \psi\ x_e\ y_e\ H\ ]^T$ and feeds them to loop when the clock triggers. It breaks the algebraic loop which occurs, since without it the output of block 5 would be its input. The memory superblock basically stores the starting value of the state and and all subsequent values for that loop call.
8. Scifunc computational block: This block reads the state vector and maps the height and total speed to static and pressure sensor voltages respectively using the formulae:
   a. Static pressure voltage = -0.000327817 H + 4.13166784
   b. Dynamic pressure voltage = $-0.00076\ (V - 3\ )^2 + 0.0002V + 2.4$
9. Analog Out: Writes the static pressure voltage to the DAC channel [0]
10. Analog Out: Writes the dynamic pressure voltage to the DAC channel [1]
11. Scope: This scope reads and displays the voltage being written to either of the analog output channels
12. C block: This block converts the inertial measurements into hex code which is put in an array
13. Digital Out: Writes the IMU data coming in hex form from the c block to the serial port

3.2 Hardware Setup

The setup that was used for the HILS is as follows:

1. A linux based pc running Fedora Core 5 with kernel patched with RTAI
2. National Instruments DAQ cards:
    a. Analog to Digital Input: National Instruments DAS1002
    b. Digital to Analog Input: National Instruments DAC 08/16
3. Interfacing board: Indigenously designed board to connect output from PC to autopilot board

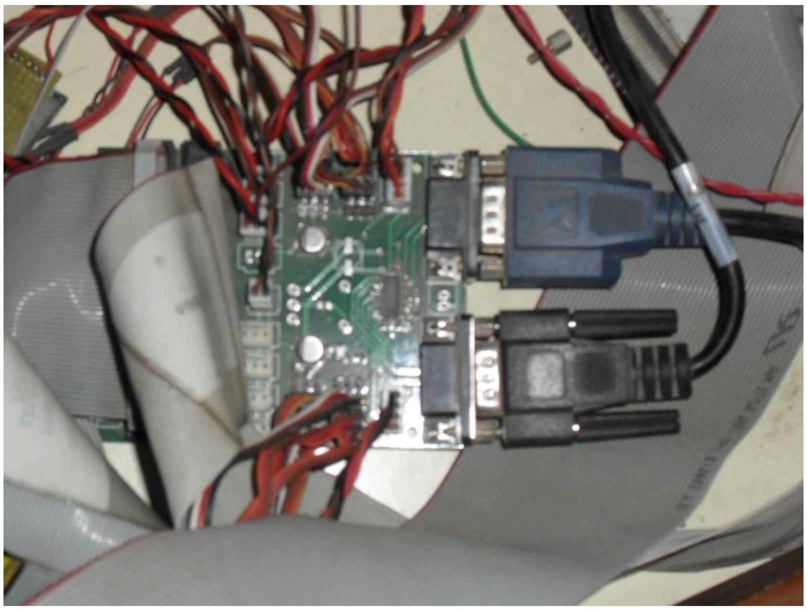

Figure 3.3: Interfacing board

4. Autopilot Board: Takes the pressure sensor and IMU inputs from the pc and generates the control command which is sent to the servos. The board contains a microcontroller (MSC 7121) programmed with a particular mission such as waypoint navigation, attitude holds etc.

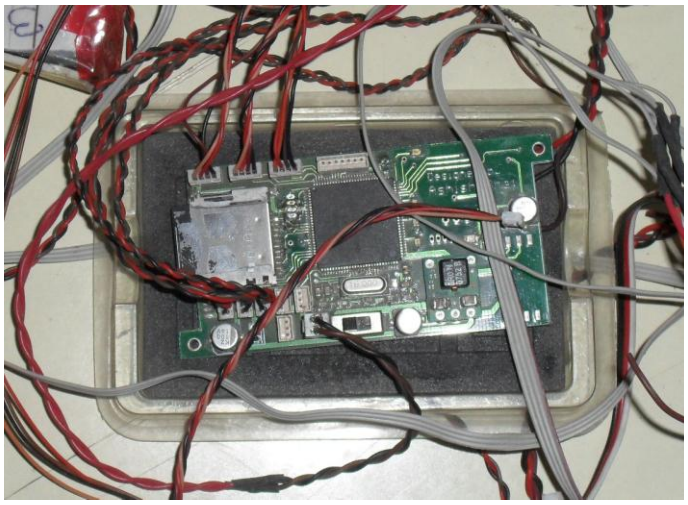

Figure 3.4: Autopilot board

5. Servos: There are 3 servo outputs from the autopilot i.e. Throttle, Elevator, Aileron. These servos are tapped using a potentiometer to read the servo deflection as voltage output and fed to the simulation through ADC input.

6. CRO: For testing the open loop HIL. It allows us to compare the output on the CRO with that on the pc to see whether the simulation is giving output in realtime to external devices.

3.3 Control Flow

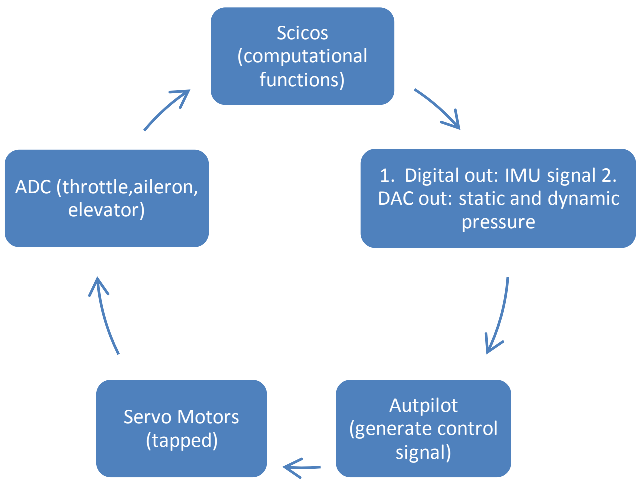
Scicos (computational functions)
1. Digital out: IMU signal 2. DAC out: static and dynamic pressure
Autpilot (generate control signal)
Servo Motors (tapped)
ADC (throttle,aileron, elevator)

Figure: Flow diagram of HIL

Chapter 4

Errors in Previous Simulation Results

4.1 Aileron pulse Input

In the first stage the simulation results did not predict roll stability about the x axis. This error was later attributed to an error in the code which has now been amended. 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.

Trim Conditions: dth=189, de = 1.2, alt = 992 m, theta = 0.097 rad

Shown here are the simulation results:

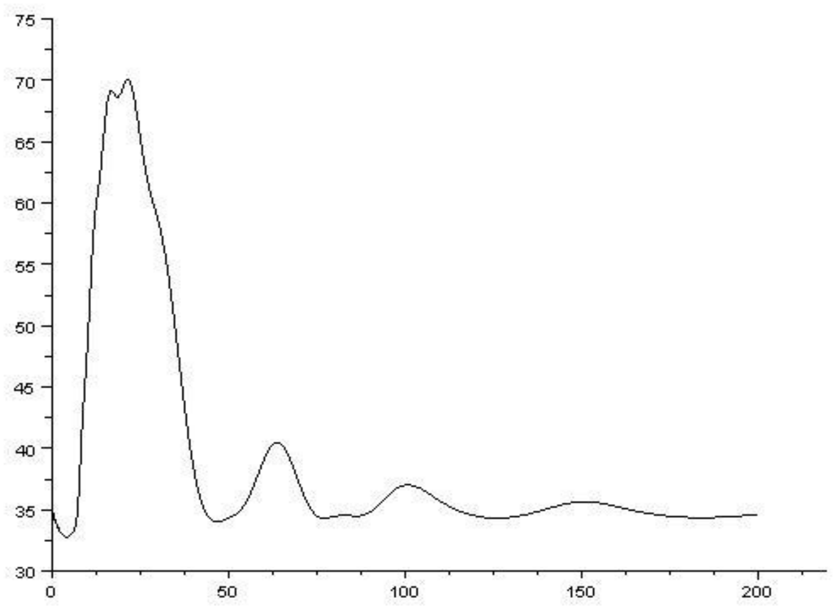

Figure 4.1: Total Velocity vs. Time

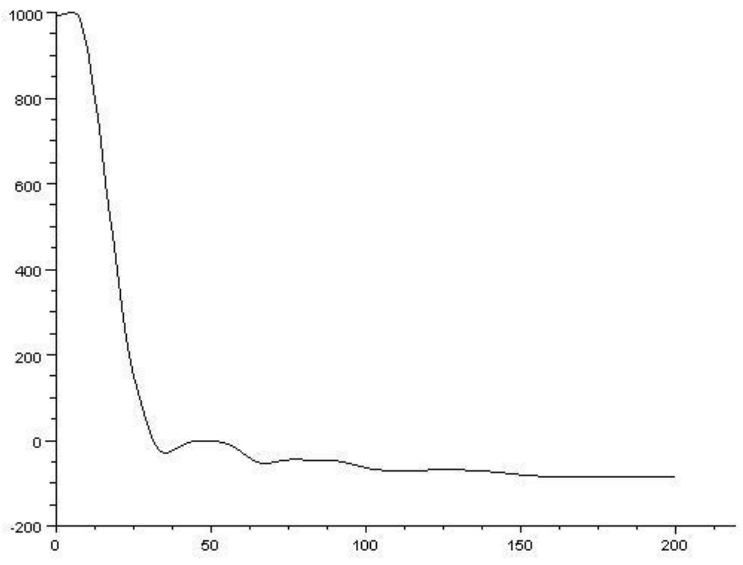

Figure 4.2: Altitude vs. Time

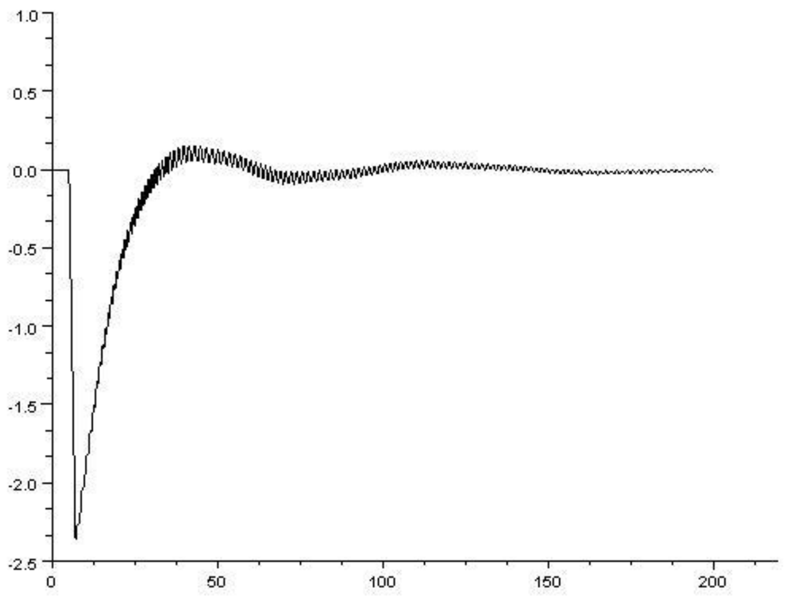

Figure 4.3: p vs. Time

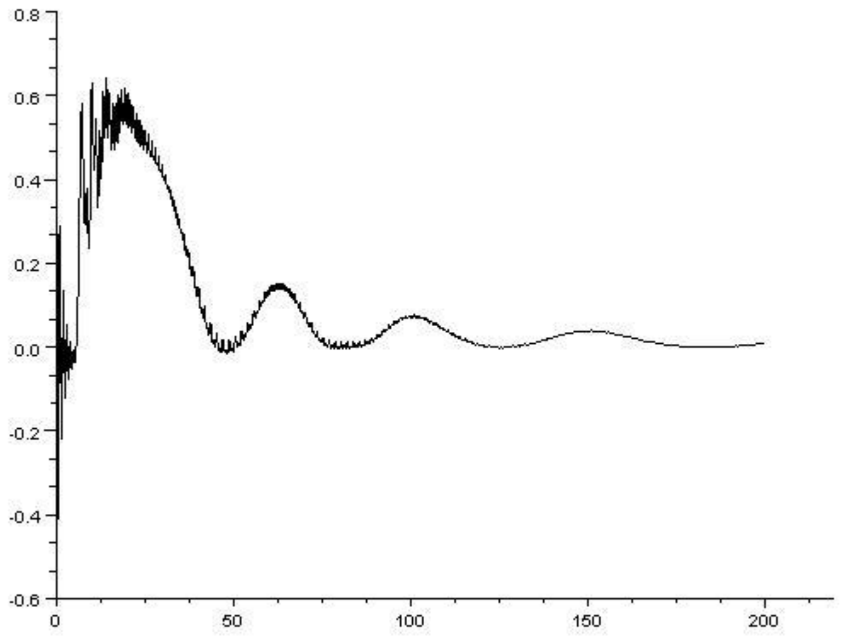

Figure 4.4: q vs. Time

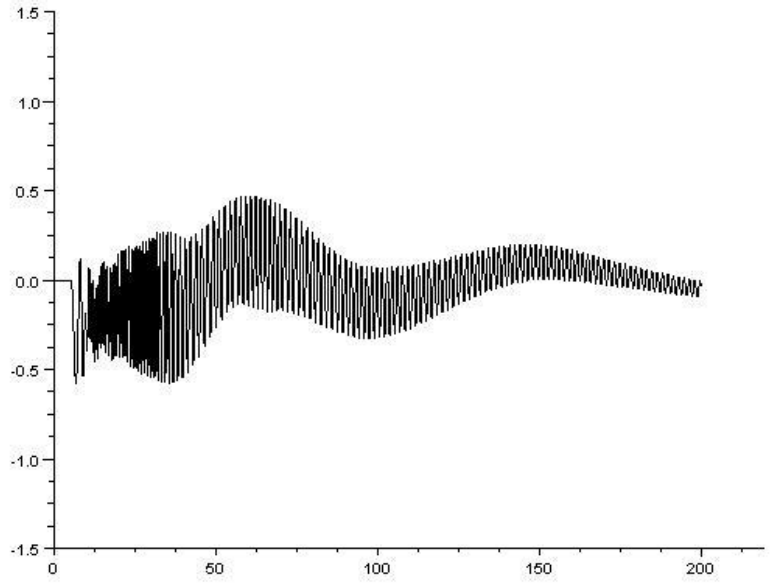

Figure 4.5: r vs. Time

Chapter 5

Realtime Testing

The simulator needed to be tested before being used for MAV simulations to verify its working. This was done by connecting the output of the DAC channel i.e. pressure sensor output to an oscilloscope. The simulation was then run and the output of the scope was recorded on a graph. The same simulation was then run in scilab without I/O. The results were then compared.

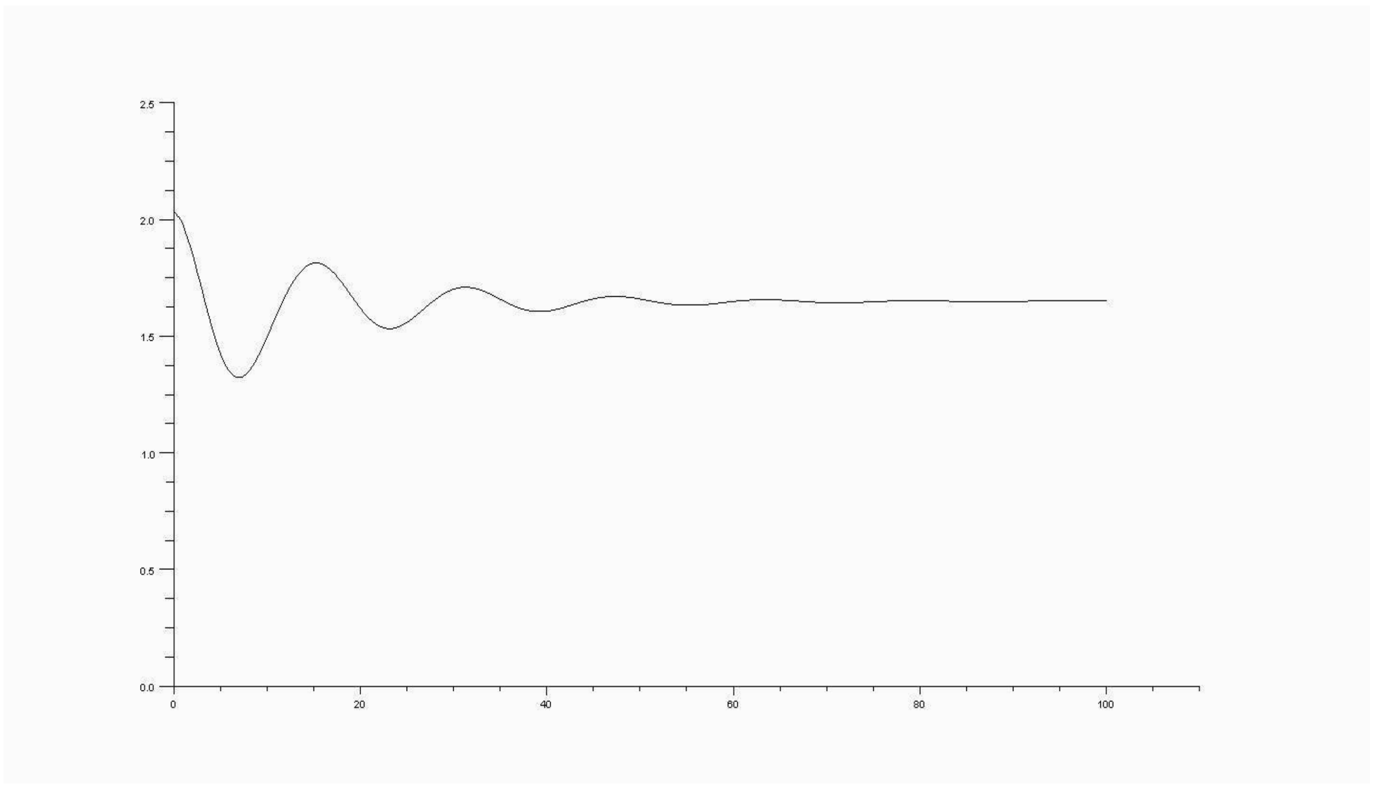

Figure 5.1: Dynamic Pressure (volts) vs. Time (s) (from scilab)

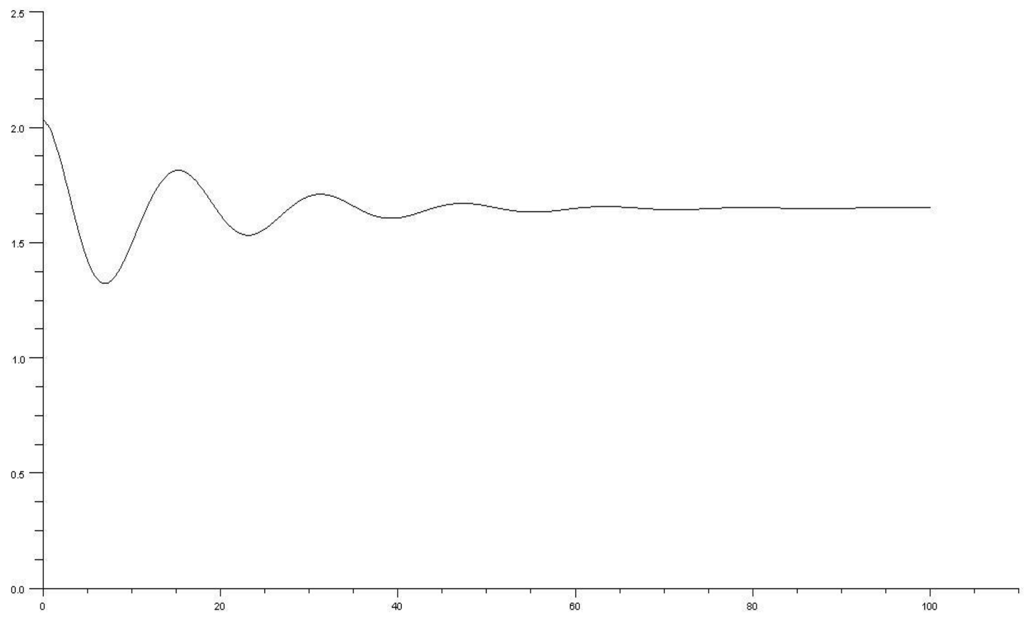

Figure 5.2: Dynamic Pressure (volts) vs. Time (s) (from scope)

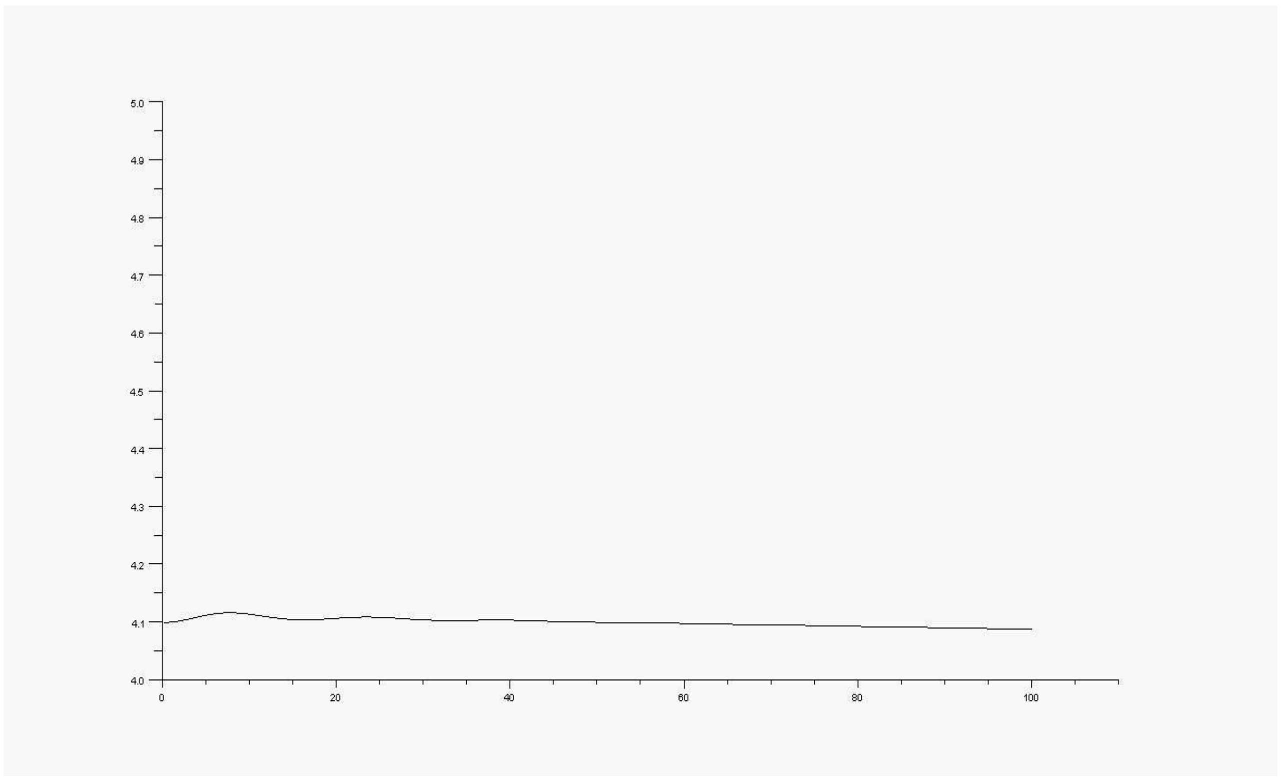

Figure 5.3: Static Pressure (volts) vs. Time (s) (from scilab)

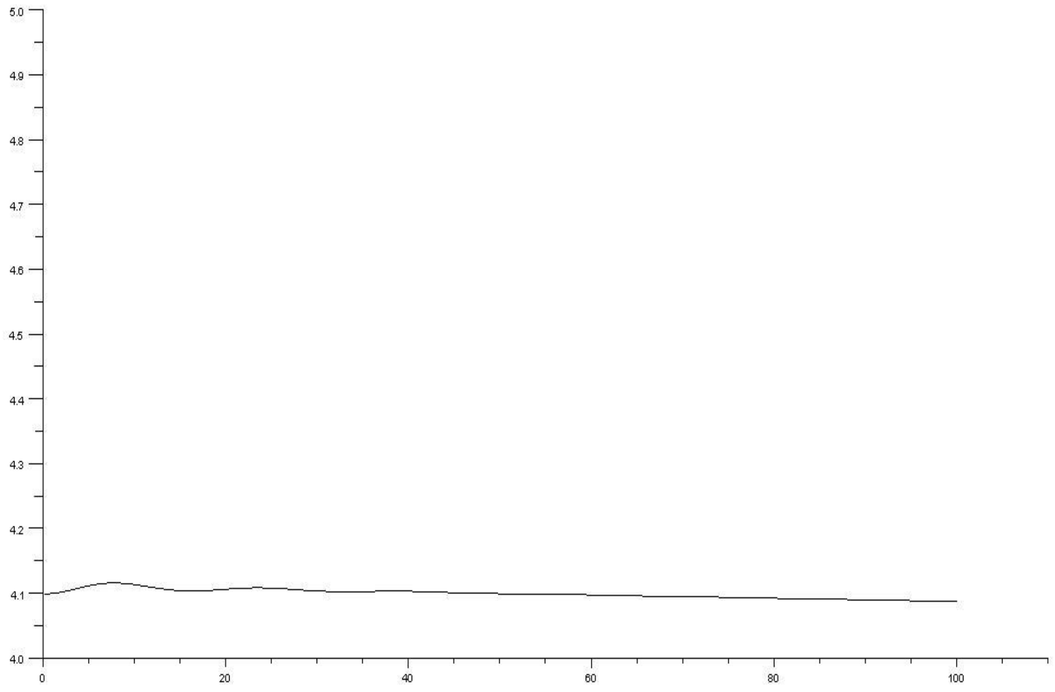

Figure 5.4: Static Pressure (volts) vs. Time (s) (from scope)

5.1 Observations

It is seen that the realtime open loops simulations and scilab simulation correspond. The output on the scope in the pc goes hand in hand with the ouput on the CRO. This demonstrates the ability of Scicos to conduct hardware in loop simulations in realtime. The voltage ouput for the sensors corresponds to the value of the voltage generated in scilab (using same mapping formulae). This verifies that the HIL works in accordance with our requirements.

Chapter 6

Conclusions

During the course of this project a lot of issues were faced concerning compatibility of software packages. Since the project depended on open-source development it was quite often that packages would come without the requisite documentation or support. RTAI-Lab is one such package. It offers a major handicap that its Codegen doesn’t work on scilab code. This takes away the advantage of using scilab language since it offers a wide variety of inbuilt computational functions such as interpolation which allows us to use non-linear aerodynamic data. C language does not have such easily available 2d and 3d interpolation functions in any standard library. However RTAI offers hard realtime capabilities which allows us to carry out simulations with a time step of 1 microsecond. This is the major advantage of using RTAI-Lab. Other methods such as Scicos-Comedi offer 99% realtime at best.

6.1 Advantages of Scicos-Comedi

The Scicos-Comedi toolbox is an easy to implement palette of comedi I/0 blocks. The biggest advantage it offers is that it can be implemented using the Scicos GUI and is compatible with scilab computational functions. There is no need for a codegen and the simulation can directly be run in the Scicos interface by setting realtime scaling to 1. This ensures that the simulation speed in software is equal to that in realtime.

6.2 Disadvantages of Scicos-Comedi

One major disadvantage is that it is not possible to increase the resolution of the simulation beyond 0.08 s. This doesn’t give accurate results as to achieve a high accuracy the simulation is desired to be carried out with a timestep of 0.001 s. This is primarily attributed to the heavy nature of the scilab computational function. As compared to C language, scilab code execution requires more processing time which adds to the load imposed by the GUI hence if the time step is lowered the output in realtime is not in sync with the output of the simulation. This creates an error in the data output which is undesirable. Also, this toolbox does not provide FIFO capability which does not allow us to send position data to the autopilot. Thus the HILS is limited to IMU and pressure measurements.

6.3 Future Applications

This methodology though not as accurate as C code executed in hard realtime, offers the ability to write simple codes in scilab and design an HIL in little or no time using scicos. It is felt that it can be used effectively as a teaching tool and as a preliminary setup for simplistic HIL testing. It is not recommended for designs which require a high accuracy and which involve complex numerical computations. Since this system promises only 99% real time it is not

References

[1] “RTAI-Lab”

http://www.rtai.org

[2] “EtherLab”

http://www.etherlab.org

[3] “Scicos”

http://www.scicos.org

[4] “Comedi”

http://www.comedi.org

[5] D. Lorenzo and M. Paolo, *“Linux Real Time Application Interface (RTAI) in low cost high performance motion control”*

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 this project and making it an invaluable learning experience. I would like to thank Kaustubh and Prasanna from the controls lab for their assistance and helping me in understanding the workings of all the hardware.

Date: April 13, 2009 Saurav Agarwal