Master's Thesis

Isaac Z. Witherspone 2014

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Faculty of Technology
Department of Mechanical Engineering
Laboratory of Intelligent Machines

Isaac ZIKI Witherspone

Comparison between dSPACE and NI systems based on real-time intelligent control of a teleoperated hydraulic servo system

Examiner: Professor Heikki Handroos, Supervisor: MSc Hamid Roozbahani
Supervisor: Professor Heikki Handroos

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology
Department of Mechanical Engineering
Laboratory of Intelligent Machines

Isaac ZIKI Witherspone

Comparison between dSPACE and NI systems based on real-time intelligent control of a teleoperated hydraulic servo system

Master Thesis
2014
76 pages, 41 figures and 3 tables
Examiner: Professor Heikki Handroos

Keywords: LabVIEW, DAQ, VeriStand, PID tuning, Hydraulic servo valve, dSPACE, ControlDesk, Force Sensor

The Laboratory of Intelligent Machine researches and develops energy-efficient power transmissions and automation for mobile construction machines and industrial processes.

The laboratory's particular areas of expertise include mechatronic machine design using virtual technologies and simulators and demanding industrial robotics. The laboratory has collaborated extensively with industrial actors and it has participated in significant international research projects, particularly in the field of robotics.

For years, dSPACE tools were the lonely hardware which was used in the lab to develop different control algorithms in real-time. dSPACE's hardware systems are in widespread use in the automotive industry and are also employed in drives, aerospace, and industrial automation.

But new competitors are developing new sophisticated systems and their features convinced the laboratory to test new products. One of these competitors is National Instrument (NI).

In order to get to know the specifications and capabilities of NI tools, an agreement was made to test a NI evolutionary system. This system is used to control a 1-D hydraulic slider.

The objective of this research project is to develop a control scheme for the teleoperation of a hydraulically driven manipulator, and to implement a control algorithm between human and machine interaction, and machine and task environment interaction both on NI and dSPACE systems simultaneously and to compare the results.

ACKNOWLEDGMENT

I dedicate this thesis to my late grandmother Mrs. Lydia Turner Witherspoon and to my mentor Mr. Abu V. Doumbia whose quest for knowledge empowers me to complete this work.

I express my deepest gratitude to my supervisor Prof. Heikki Handroos for giving me this unique opportunity to work with him and for providing full access to the Laboratory of Intelligent Machines to do this project. This thesis wouldn't have completed if not for the excellent guidance from my assistant supervisor M.Sc Hamid Roozbahani. His knowledge was of great help during the project. I thank him for the countless time spent with me in the laboratory giving advises and suggestions. Thanks and appreciation also goes to Mr. Juha Koivisto the Lab technician for providing a good working atmosphere for me and for making sure that I get the necessary hardware and software. I would like to thank Mr. Vesa Kyllönen the district sales manager from National Instruments Finland for providing the laboratory with all necessary tools used for the project.

I wouldn't have completed this work if not for the support and encouragement from my mom Mrs. Lydia Kun Saastamoinen and her husband Mr. Kalle Oskari Saastamoinen. Not to be overlooked is the indirect support from my siblings Makeme and Oskari and my niece Barbara.

Last but not the least an immerse thanks and appreciation go to all my relatives and friends here and aboard for their moral support and for believing in me and whose voices silently cheer me up to the finish.

Table of Contents Page

List of Figures

List of Tables

Nomenclature

ADAMS	Automated Dynamic Analysis of Mechanical Systems
RTI	Real-Time Interface
FEM	Finite Element Method
VI	Virtual Instrument
PWM	Pulse Width Modulation
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
FPGA	Field-Programmable Gate Array
Dof	Degree of freedom
ITER	International Thermonuclear Experimental Reactor
NI	National Instrument
DAQ	Data Acquisition
HDL	Hardware Description Language
I/O	Input/Output
A/D	Analog to Digital
D/A	Digital to Analog
ECU	Electronic Control Unit
LIN	Local Interconnect Network
ATI	The name of the company that manufacture the sensor
PCI	Peripheral Connect Interface
PCMCIA	Personal Computer memory Card Interface Adaptor
OMEGA 160	The manufacturer name of the sensor used
PID	Proportional Integral Derivative
K_p	Proportional gain
K_i	Integral gain
K_d	Derivative gain
GUI	Graphical User Interface
HIL	Hard-in-Loop
MIL	Model-in-loop
SIL	Software in the loop

1. Introduction

1.1 Basic Information

Project Leader: Professor Heikki Handroos, Lappeenranta University of Technology

This project was carried out in the Laboratory of Intelligent Machines at Lappeenranta University of Technology in Finland. The design software are LabVIEW, VeriStand and the devices used for acquiring and analyzing the data with the above listed software were provided by National Instrument. Matlab Simulink was also used as design software and ControlDesk was used as the real-time software. The dSPACE board DS1005 was used for data acquisition and the comparison was made with the NI systems.

1.2 Background

The laboratory of intelligent machines at LUT has done extensive researches with hydraulic manipulators. The laboratory has been able to implement multidisciplinary research combining mechanics, structural analysis, serial and parallel robots, hydraulic and servo control. These areas have become the core competence of the lab researches. Some of its significant undertaken projects include a log crane manufactured by John Deere forestry [Figure 1: d]. The laboratory been able to develop parallel and parallel hybrid robots for machining and welding in the ITER fusion reactor [Figure 1: a].

PENTA-WH is another 5-DOF inter sector weld/cut robot that is to be developed for the ITER fusion reactor in the international ITER program. The prototype of the robot is shown in [Figure 1: c]. The kinematic and dynamic behavior of those manipulators has been carefully analyzed by means of modeling, simulation and experiments. The kinematics and dynamics directly in work space have been studied. The hybrid position/force control of PENTA-WH has been carried out in impacting application.

Some tasks are controlled by a human operator onsite by control levers or joysticks. These systems form a closed-loop teleoperator, which is in terms of energy a two-port system where the master and the slave robots interact mechanically with the human operators and the slave environments respectively. In 1998 the idea of applying parallel structures in a working machine boom was created. The prototype was designed firstly by applying virtual prototyping. The first physical prototype named MULTIPOD [Figure 1: b]

was built in early 2000. The manipulator MULTIPOD is a six degree-of-freedom parallel manipulator equipped with six hydraulic servo-axes. It is based on two parallel 3-DOF mechanisms with a serial connection. (Roozbahani, 2011, p. 22)

The parallel manipulator MULTIPOD is specially designed for carrying out drilling tasks in mines.

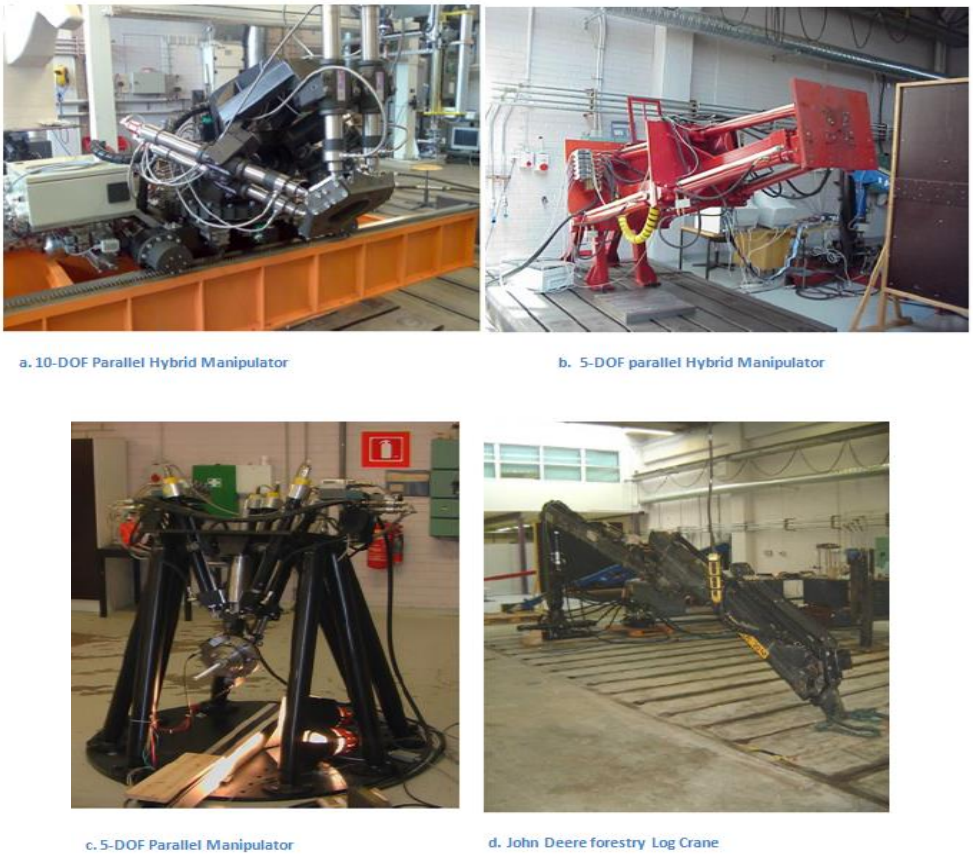

Figure 1: Hydraulic Machines

1.3 Project Introduction

The laboratory of intelligent machine does collaborate with companies and other universities for the implementation of research project. Because of its openness to new approaches and problem solving methods, new software for control and design are constantly brought it for utilization. In view of this, National Instrument provided the Laboratory with its software LabVIEW and VeriStand and data acquisition devices such as the NI USB-6259 and PXI-8186 to familiarize students with its state-of-the-art engineering tools.

In view of these, the laboratory decided to test these new software and devices to develop a control scheme for the teleoperated manipulator and to implement an ideal transparent mapping between human and machine interaction and machine and the task environment. [Figure 2]

Prior to this thesis, the laboratory of intelligent machines has never used NI system in any of its major projects. The laboratory is more familiar with Matlab & Simulink, dSPACE, ADAMS, FEMAP and SolidWorks. The unfamiliarity of NI system made the project difficult right from the start. Considerable amount of time was spent during the learning phase of the NI systems.

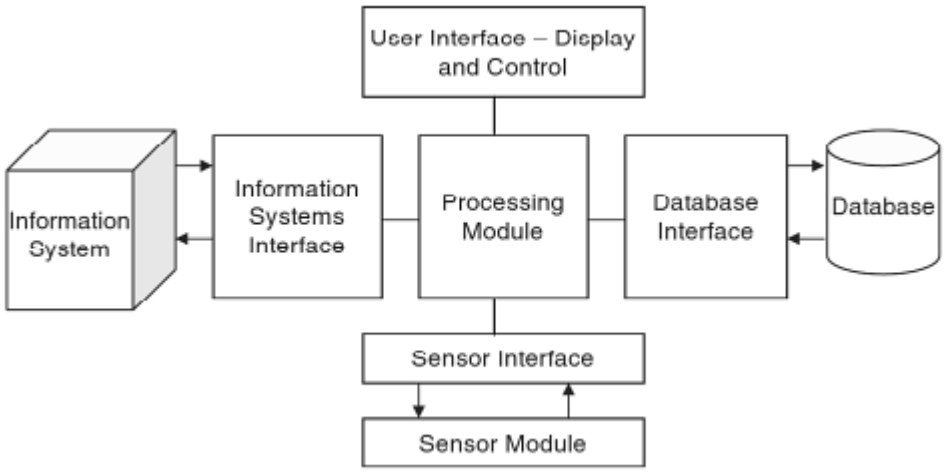

Figure 2: Schematic view of operator & machines

1.3.1 The research contribution and limitations

The project covers a period of roughly nine months. With the first three months spent on familiarization of NI-software and devices. During this period a course was attended and a mini project was implemented for more familiarity. At this time extensive literatures survey were carried out for the preparation of the experimental part and teleoperation experiment with joy-stick.

The research is limited to a single degree-of-freedom hydraulic slider. The contributing factor to the research is the experimental rig on which the slider sits, the hydraulic servo valve and a 6-DOF force sensor. In addition to these, the mathematical modelling of the

hydraulic servo valve was reviewed but was not use on the NI system but rather for Simulink and dSPACE.

1.4 Introduction to the Software and Hardware

Basically the software and hardware packages can be divided into two subgroups. We have on one hand NI system which comprises of LabVIEW, VeriStand as the software while NI USB-6259 and PXI-8186 are the DAQ devices. The other subgroup consists of ControlDesk and ds1005 data acquisition board both produced by dSPACE. Matlab Simulink is not produced by dSPACE but as many other software, it is fully compatible with dSPACE. Programs made in Simulink can be link with dSPACE, for this reason the both will be treated as one to avoid confusion.

1.4.1 NI LabVIEW and VeriStand Software and Hardware

LabVIEW is a programming environment in which programs are created using graphical notation connecting functional nodes via wires through which data flows. In this regard, it differs from traditional programming languages like C, C++ or Java in which text is use for programming. (Travis Jeffrey, 2009, p. 3) LabVIEW can create programs that run on a variety of embedded platforms, including FPGAs, Digital Signal Processors (DSPs), and microprocessor. LabVIEW uses a graphical programming language call “G” for graphical. It is specially designed to take measurements, analyze data and present the result.

What makes LabVIEW different from standard C or Java development systems is that while other programming systems use text-based languages to create lines of code, LabVIEW uses a graphical programming language to create pictorial code called block diagram. The graphical programming eliminates a lot of the syntactical details associated with text-based language, such as where to place semicolons and curly braces. Execution in LabVIEW is based on the principle of dataflow in which functions execute only after receiving the necessary data.

1.4.1.1 NI LabVIEW VIs

A LabVIEW program consists of one or more virtual instrument (VIs). VIs are called such because their appearance and operation actually imitate actual physical instrument. However behind the scenes they are analogous to main programs, functions, and subroutines from popular programming languages like C or Basic. A LabVIEW program is called VI (pronounced “vee eye”,). A VI has three main parts: a front panel, a block diagram and icons. (Travis Jeffrey, 2009, p. 7)

- The front panel is the interactive user interface of a VI. It can contain knobs, push buttons, graphs and many other controls which are user inputs and indicators which are program inputs. (See figure 3)

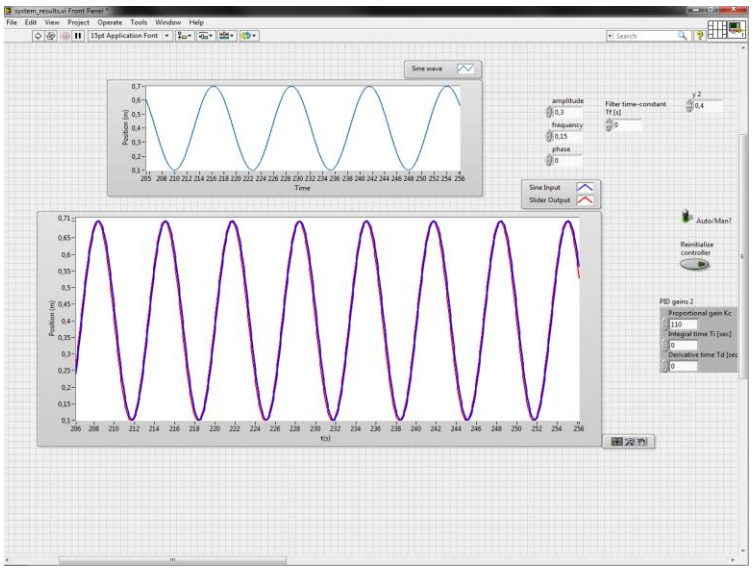

Figure 3: LabVIEW front panel displaying sinus wave and inputs parameters

- The block diagram is the VI’s source code constructed in LabVIEW’s graphical programming language, G (see Figure 4). The block diagram is the actual executable program. The components of a block diagram are lower-level VIs, built-in functions, constants, and program execution control structures. Wires are drawn to connect the appropriate objects together to define the flow of data within the VI. Front panel object have the corresponding terminal on the block diagram which enable data to pass from the user to the program and back to the user.

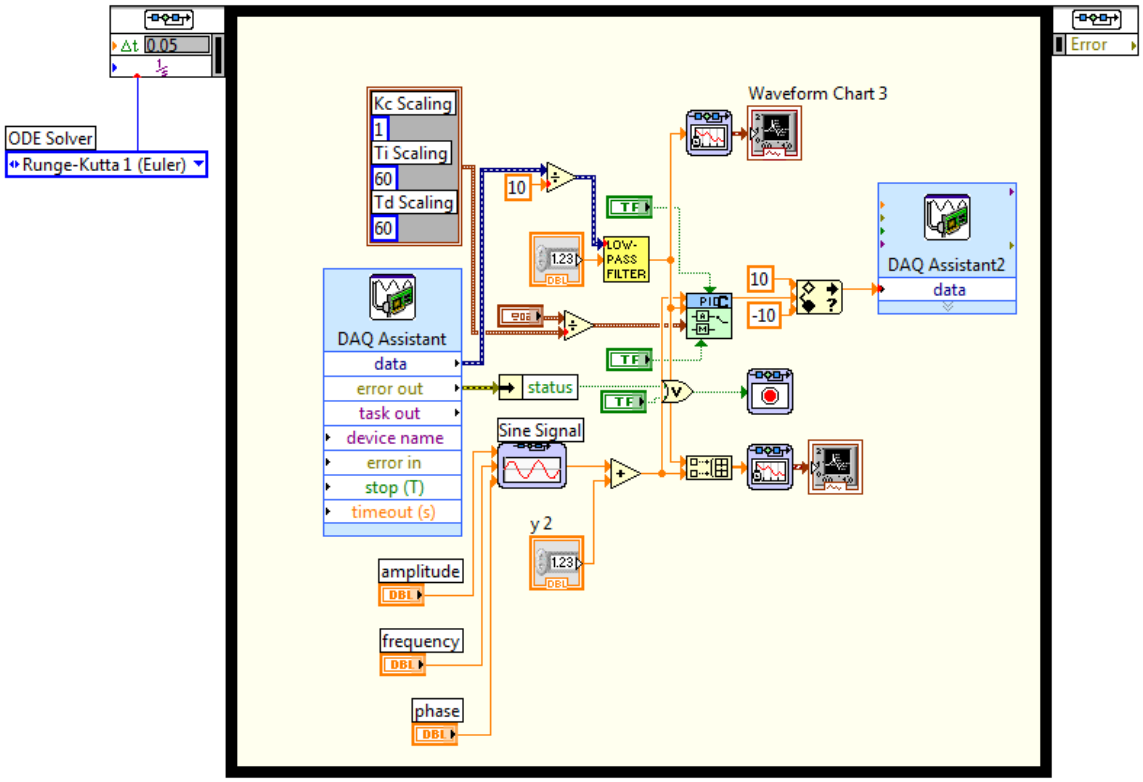

Figure 4: LabVIEW block diagram with G codes

- In order to use a VI as a subroutine in the block diagram of another VI, it must have an icon with a connector. A VI that is used within another VI is called a subVI and is analogous to a subroutine. An icon is the VI's pictorial representation and is used as an object in the block diagram of another VI.

The below table illustrates how LabVIEW differ from conventional programming language. It presents a list a few common LabVIEW terms with their conventional programming equivalents.

Table 1: LabVIEW terms and their conventional equivalents

LabVIEW	Conventional Language
VI	program
function	function or method
subVI	subroutine, subprogram, object
front panel	user interface
block diagram	program code
G	C, C++, Java, Pascal, BASIC, etc

1.4.1.1.1 Advantages and Disadvantages of NI LabVIEW

Just as there are pros and cons in every other conventional software, LabVIEW also has its advantages and disadvantages. One thing to be considered is that this list is based on the personal experience of the user and is not to be taken on the general level. What is consider a disadvantage for one, may be just what the other needs. With this in mind the advantages includes:

- Graphical interface is flexible and easy to use without any programming skills
- Provides a universal platform for numerous application in diverse fields
- Easy to reuse embedded code. It provides easy integration with embedded C and HDL code
- LabVIEW is excellent for data acquisition because LabVIEW can command DAQ devices to read analog input signal, generate analog output signals, read and write digital signals, and manipulate the on-board counters for frequency measurement, pulse generation, quadrature encoder measurements and so on, to interface with the transducers

The disadvantages can be listed as follows:

- Inability to write descriptive comments
- Nonlinear graphical programming interface

- Impossible to debug and impossible to insert new commend into the established program without ruining the organization structure
- Difficult to make wire connection in complex program
- LabVIEW has a serious version compatibility problem. The run engines of older versions are not compatible with the newer one.
- Toolkits for third party software are very expensive
- LabVIEW is expensive compare to other programming languages

1.4.1.2 NI VeriStand

An NI VeriStand real-time test application typically consists of one or more real-time execution targets that communicate to a host system via Ethernet. Each real-time execution target is running the NI VeriStand Engine, which is configured from the Windows-based host system and deployed over Ethernet. Once your NI VeriStand Engine configuration is deployed, you use the NI VeriStand Workspace window and the tools it provides, such as the Stimulus Profile Editor, to interact with your test system at run time. When developing and running NI VeriStand applications, you use three primary windows: the System Explorer, the Workspace, and the Stimulus Profile Editor.

1.4.1.2.1 NI VeriStand System Explorer

The System Explorer window is where the system is defined. It contains setting for the hardware I/O as well as functionality from other programming or modeling environments. The configuration of a system definition is done by adding item to the system definition tree and setting items configuration options. After this process, the system definition is deployed to the execution target to be use in the NI VeriStand Workspace.

1.4.1.2.2 NI VeriStand Workspace

The Workspace is the user interface of a deployed system definition. Here, interface objects are placed and mapped to channels in the system real-time application. NI VeriStand allows the usage of multiple Workspaces for a single real-time application to organize the system controls and indicators into logical groups. There also have a user access management feature which allows the designer to control the access privileges of user based on their login account.

When working with the run-time editable Workspace, a variety of controls and indicators can be added to the Workspace and connected to NI VeriStand channels. While NI VeriStand includes a variety of these objects, it is possible to use NI LabVIEW to create custom controls and indicators that can be used with the NI VeriStand Workspace. Some examples include adding user interface objects that more closely mimic the systems interfaces or creating user interface objects with custom functionality such as inline processing or alarming. Though NI VeriStand provides many tools that can use to monitor and interact with the NI VeriStand Engine, in LabVIEW run-time tools can be created and added to the NI VeriStand Workspace

1.4.1.2.3 NI VeriStand Stimulus Profile Editor

The Stimulus Profile Editor is a tool in the NI VeriStand Workspace for creating stimulus generation and logging tasks that are deployed to the NI VeriStand Engine for deterministic execution of test profiles.

Stimulus profiles are created by specifying a list of stimulus generation steps that the NI VeriStand real-time engine will perform. There are steps for generating waveforms, playing back data files, setting channel values, as well as a conditional step for implementing branching and looping structures in the stimulus generators. Multiple logging tasks can be added with independent logging rates and trigger conditions to the stimulus profile. For example, one log file can capture data at a reduced rate for slow changing channels and another file can be set to acquire at a higher rate if a trigger condition occurs during the test.

Stimulus profiles execute in the NI VeriStand real-time engine, however additional test automation capabilities from the host interface using the NI VeriStand Workspace macro recorder or by using other tools such as NI TestStand or Iron Python.

In addition to the Stimulus Profile Editor, the NI VeriStand Workspace includes many other tools that are useful when working with real-time testing applications. There are tools for monitoring alarms, calibrating hardware I/O, and channel value forcing. There is also a real-time console viewer, which monitors the operation status of the real-time execution target.

1.4.1.2.4 NI VeriStand Engine Architecture

The NI VeriStand Engine is the non-visible execution mechanism that is responsible for executing hardware I/O, models, procedures, alarms, and other test system tasks that are specified in the system definition file. The engine controls the timing of the entire system as well as the communication between the NI VeriStand Engine and the Workspace.

The NI VeriStand Engine consists of multiple Timed Loops whose execution timing is controlled by hardware events with microsecond resolution. Deterministic memory buffers provide communication between tasks in different loops without inducing jitter into engine execution. With this multi loop architecture, the NI VeriStand Engine naturally takes advantage of the parallel processing power of multi-core processors, increasing the system performance.

A variety of engine execution settings can be configured when creating the system definition including the ability to choose between parallel and low-latency and sequential architectures. Additionally, the NI VeriStand engine publishes a variety of system health parameters that can be access at run time, or can be used in the NI real-time Execution Trace Toolkit for greater visibility into the application's execution.

The engine's real-time I/O tasks use a hardware-timed, single-point I/O structure that is ideal for simulation, control, and point-by-point analysis tasks. However, support for higher-speed, buffered signal generation and acquisition can also be added using an NI VeriStand custom device. The NI VeriStand Engine can run on PCI- and PXI-based real-time systems from National Instruments as well as NI CompactRIO and NI Single-Board

RIO interfaces that have 128 MB of DRAM or greater. A real-time system gives the ability to execute tests deterministically with synchronized I/O, a critical capability for applications that are implementing closed-loop control or system simulations that interact with real-world components. However, for systems with lower performance needs or for implementing model-in-the-loop (MIL) or software-in-the-loop (SIL) tests, the NI VeriStand engine can be run on the same computer as the user interface.

1.4.1.2.5 NI VeriStand Models

NI VeriStand can import compiled code you create in LabVIEW, The MathWorks, Inc. Simulink software, SimulationX from ITI, MapleSim from Maplesoft, GT-POWER from Gamma Technologies Inc, and many other modeling and programming environments. With this capability, real-time closed-loop control, system simulation, signal processing, and signal generation are added to NI VeriStand applications. While many environments are already supported, support for other environments capable of producing C code using the NI VeriStand Model Framework that is provided with the product can be included.

1.4.1.2.6 NI VeriStand FPGA Personalities

When adding real-time I/O hardware interfaces to NI VeriStand, a variety of standard analog, digital, and communication bus interfaces can be configured; however, NI VeriStand also provides the ability to create user-defined I/O hardware using LabVIEW FPGA-based reconfigurable I/O (RIO) devices. With this capability user-defined I/O hardware interfaces that implement custom signal processing, simulation, triggering, and/or control tasks that execute at rates as fast as 25 nanoseconds and that do not consume any of the processing bandwidth of the real-time application can be created. Additionally, because the I/O interface is FPGA-based, the reconfiguration personality or behavior of the device to adapt to new requirements or to create test systems capable of being used for multiple applications without changing I/O interface hardware.

1.4.1.3 NI data acquisition

Data acquisition is the process of measuring an electrical or physical phenomenon such as voltage, current, temperature, pressure or sound. The goal of data acquisition is to capture data from one or more instruments so that it can be analyzed and stored. The term data acquisition refers to the process of automatically importing data from one or more sensors or transducer directly into a computer system.

In this context, a sensor refer to a device that responds to a physical change and output an electrical signal, while a transducer is a device that converts energy from one form to another. For example, a thermocouple is a sensor that generates an electromotive force (emf) because of two dissimilar metals joined at the thermocouple junction. The emf generated is a low-level voltage measure in mV, which when sent down a wire, becomes a voltage signal. A transducer can be used to convert the low-voltage to a higher voltage.

A simple system requires a transducer for signal output, data acquisition hardware (DAQ) and a computer. (See Figure 5) Most modern measurement devices routinely bundle sensors and transducers together to generate and transform the output signal to a useful form. (Larsen, 2011, p. 143)

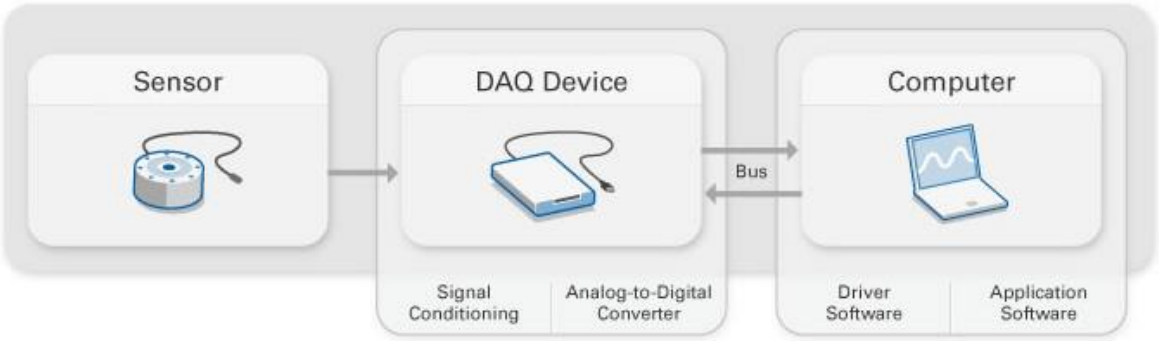

Figure 5: Data acquisition system

1.4.1.4 NI DAQ devices

During the course of this project, two NI-DAQ devices were used; NI USB-6259 and NI PXI-1031 with embedded controller NI PXI-8186. The embedded controller has three chassis boards, NI-6289, NI-6602 and NI-6733. The two latter chassis boards were not

used. For the NI USB-6259, an USB cable was used to communicate with the device and for embedded controller NI PXI-8186, and local area network was set up and an Ethernet cable was used. The PXI-1031 is more suitable for industrial applications and it provides a lot more options to communicate with it. The USB-6259 is easy to set-up and is more suitable for laboratory experiment. More information about the devices will be given in (chapter 5).

1.4.2 dSPACE software

dSPACE is a single-Board Hardware for building a complete real-time control system with just one controller board. dSPACE provides the working environment for developing, programming and testing control systems. With dSPACE, ideas are put into practice faster and errors are eliminated sooner. Figure 6 illustrates a dSPACE board hardware.

1.4.2.1 dSPACE ControlDesk Software

ControlDesk Software is universal experiment software for electronic control unit development. It has the capability of integrated electronic control unit calibration measurement and diagnostic access. It gives synchronized data capture across electronic control units, RCP and HIL platforms and bus system. This software has a powerful layout, measurement and post-processing. ControlDesk standard can be operated in two models: The developer mode gives the designer the full functionality, and the operator mode protects the experiments against unauthorized changes. (Embedded Success dSPACE, 2010, s. 186)

ControlDesk provides a platform to experiment seamless ECU development. All necessary tasks are performed in a single working environment from start to finish. ControlDesk can be used for experimentation that involves rapid control prototyping full-pass or bypass, Hardware-in-the-loop simulation, ECU measurement, calibration and diagnostics. It provides access to bus systems such as CAN, LIN and FlexRay and virtual validation with dSPACE VEOS. ControlDesk can access virtual electronic control units

generated with SystemDesk and Simulink plant models that are simulated offline on the PC. The figure below shows an example layout written on ControlDesk.

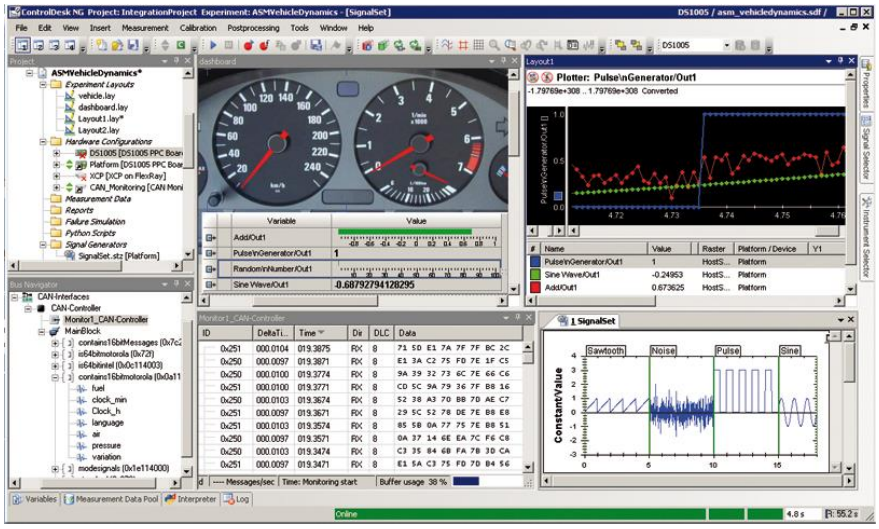

Figure 6: ControlDesk's graphical interface for dSPACE Simulator

The next figure shows a designed layout in ControlDesk. In this layout, the measured forces and torques from the force sensor is figured numerically and graphically. The transferred position data from the joystick and the real-time position of the hydraulic slider is shown in the same graphical box. The layout is the central hub between parts of the system, the hardware and the written codes. ControlDesk offers a variety of virtual instruments for building and configuring virtual instrument panels according to the needs. Any set of instruments in a virtual instrument panel that is specific to an application can be added by drag and drop.

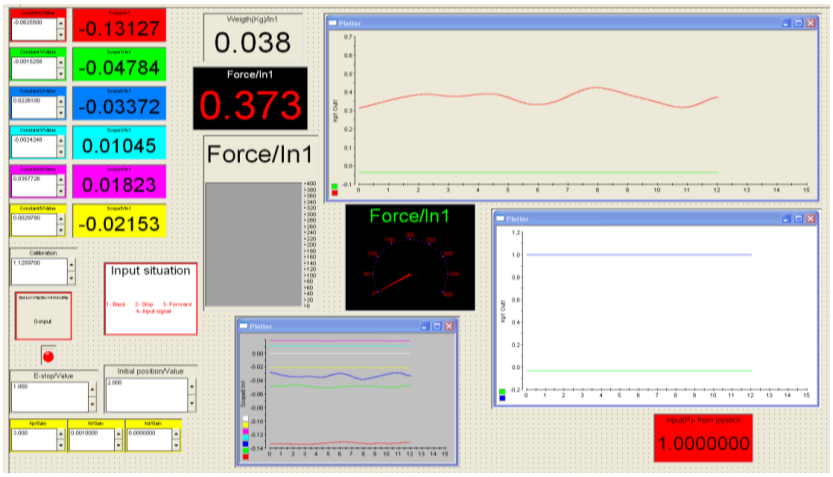

Figure 7: ControlDesk designed layout of force sensor signals

2. Description of servo hydraulic system

2.1 Introduction to a servo system

What is a control system? A system can be thought of as a collection of interacting components, although sometimes interest might lie just in one single component. These components will often be discrete physical elements of hardware, but can equally well be functional parts of such physical components. The system of interest might be a power station, a steam turbine in the power station, or a control valve of the turbine; it might be an airplane, its air conditioning, an engine or part of an engine; a process plant for the production of a chemical, or a large or small part of the plant; a human being, or some part of the body such as the muscle control mechanism for a limb; or it might be the economic system of a country, or any other from a wide range of fields. (Schwarzenbach J., 1992, p. 1)

A control system is an interconnection of components forming a system configuration that will provide a desired system response. The basic for analysis of a system is the foundation provided by linear system, which assumes a cause effect relationship for the components of a system. The component or process to be controlled can be represented as an open loop control without feedback or as a close loop control with feedback.

An open-loop control system utilizes a controller or control actuator to obtain the desired response. The open-loop control system utilizes an actuating device to control the process directly without using device and a feedback measurement or sensor. Figure 8 shows the block diagram of an open-loop system.

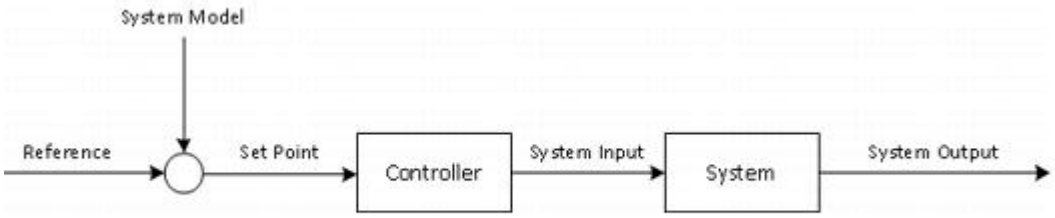

Figure 8: Open-loop block diagram

A close-loop control system utilizes additional measure of the actual output to compare the actual output with the desired output response. The measurement of the output is termed as the feedback signal. A feedback control system tends to maintain a relationship of one system variable to another by comparing functions of these variables and using the difference as a means of control. As the system becomes more complex, the interrelationship of many control controlled variable may be considered in a control scheme. (Dorf C. Richard, 2001, p. 25)

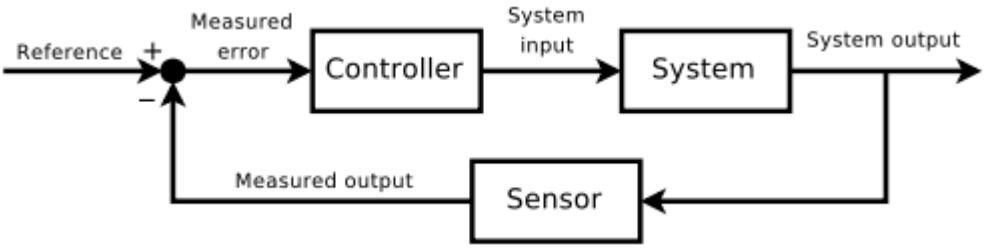

Figure 9: Close-loop block diagram

2.2 Design of a control System and the system modeling

The studied system comprises a directly operated proportional servo solenoid valve with position control, cylinder, power unit, four pressure sensors, and a single displacement sensor (Figure: 10). Since the type of control is of influence on the outcome of the output, e.g., the position of the mass, the system is in closed loop positional control, i.e., the position of the mass as feedback was summed with the signal from the pulse generator to form the input signal to the system. The mathematical model of the system involves a large number of parameters, which may be completely unknown or only known within certain ranges. (Roozbahani, 2011, p. 32). The mathematical modeling of the hydraulic valve was not needed to implement the experiment base on the NI-systems but it was use for the experimentation base on Simulink and dSPACE.

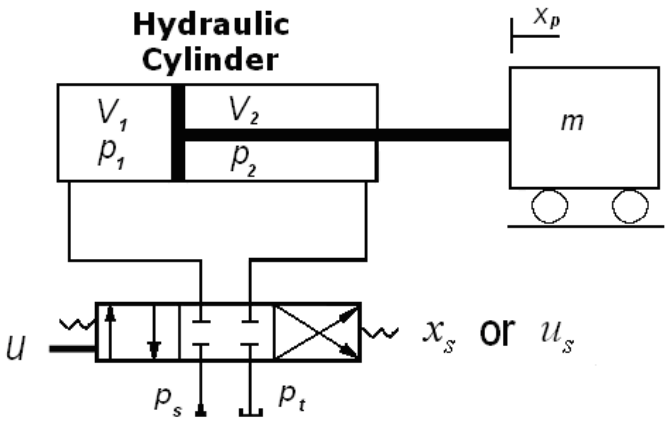

Figure 10: Schematic diagram of the servo hydraulic system

3. Hydraulic Slider

A Hydraulic cylinder (also called a linear hydraulic motor) is a mechanical actuator that is used to give a unidirectional force through a unidirectional stroke. It has many applications, notably in engineering vehicles and aero-planes. A hydraulic cylinder consists of the following major parts: Cylinder barrel, Cylinder Bottom or Cap, Cylinder Head, Piston, Piston Rod, Rod gland.

Hydraulic cylinders get their power from pressurized hydraulic fluid, which is typically oil. The hydraulic cylinder consists of a cylinder barrel, in which a piston connected to a piston rod moves back and forth. The barrel is closed on each end by the cylinder bottom (also called the cap end) and by the cylinder head where the piston rod comes out of the cylinder. The piston has sliding rings and seals. The piston divides the inside of the cylinder in two chambers, the bottom chamber (cap end) and the piston rod side chamber (rod end). The hydraulic pressure acts on the piston to do linear work and motion. A hydraulic cylinder is the actuator or "motor" side of this system. The "generator" side of the hydraulic system is the hydraulic pump which brings in a fixed or regulated flow of oil to the bottom side of the hydraulic cylinder, to move the piston rod upwards.

The piston pushes the oil in the other chamber back to the reservoir. If we assume that the oil pressure in the piston rod chamber is approximately zero, the force F on the piston rod equals the pressure P in the cylinder times the piston area A:

$$F = P \times A \tag{1}$$

The piston moves instead downwards if oil is pumped into the piston rod side chamber and the oil from the piston area flows back to the reservoir without pressure.

3.1 Modeling of the hydraulic valve

Voltage u (V) is the valve input. When the input is applied to the valve, spool is shifted and openings are produced. The shift of the spool, namely position displacement x_s (mm), is in both directions.

This displacement is small (of the scale 10^{-1} mm) and not measureable; the full displacement is also not available. But actuator and the spool are connected to the linear variable differential transducer (LVDT). The range of the LVDT signals u_s (V) is ± 10 V for an input of ± 10 V and u_s is testable. In this study, voltage u_s is measured and directly used for providing information of the spool displacement. Using the normalized spool displacement, that is., $u_s/10$ would be another option.

The main spool of the valve is a mass held in position by a spring system. The main spool is the key component of the flow divider. The relation between the simulated valve spool position x_s (m) and the input voltage u can be of first order as:

$$G_v(s) = x_s(s)/u(s) = \tau_1/(s+\tau_2), \text{ or } \dot{x}_s = t_1 \cdot u - t_2 \cdot x_s, \quad (2)$$

where term τ_1 has no unit and term τ_2 has unit (1/sec or s^{-1}). But term τ_1 should have unit as ($ms^{-1}V^{-1}$) since τ_1 multiply by u is measured in (ms^{-1}). Similarly we can also represent the transfer function of the valve dynamics, between u_s and u, using the first order system, as the following:

$$\dot{u}_s = t_1 \cdot u - t_2 \cdot u_s . \quad (3)$$

Here term t_1 is the gain (s^{-1}) and t_2 the time constant (s^{-1}).

The relationship between u_s and u can also be given as:

$$\dot{u}_s = (K \cdot u - u_s)/T, \quad (4)$$

where K is the gain (no physical unit) and T the time constant (s).

A first order model can only be applied in case of limited frequency range, well below the natural frequency of the valve; the second order model responds the servo valve dynamics through a wider frequency range. A linearized model for an electro hydraulic servo system with a two-stage flow control servo valve and a double ended actuator has

revealed that the higher order model fits closer to the experimental data because of the reduced un-modeled dynamics.

When a second order transfer function is used to represent the valve model, the valve's dynamics could be as the following:

$$\ddot{u}_s = k \cdot \omega_n^2 \cdot u - 2 \cdot \zeta \cdot \omega_n \cdot \dot{u}_s - \omega_n^2 \cdot u_s, \tag{5}$$

where k is the gain (no physical unit), ζ the damping ratio (no physical unit), and ω_n the natural angular frequency (radian/s).

The valve flow gain depends upon the rated flow and input current. The rate of change of input signal is also limited, in such control boards in order to provide a well behaving response of the valve. In addition, the servo solenoid valve under study has an on-board electronics (OBE), providing position feedback of the spool of the valve. Disturbances as friction or flow forces on the spool are rejected.

Using the Newton's second law, the equation of motion for the servo hydraulic system becomes:

$$m \cdot \ddot{x}_p = p_1 \cdot A_1 - p_2 \cdot A_2 - F_f. \tag{6}$$

Here, m denotes the mass weight (kg), x_p the displacement of piston (m), A_1 and A_2 the piston areas (m^2), p_1 and p_2 the pressures (Pa), and F_f the friction force (N).

The pressures at valve ports were described as:

$$\begin{aligned} \frac{dp_1}{dt} &= \frac{\beta_{e1}}{V_1}(Q_1 - A_1 \cdot \dot{x}_p + Q_{Li} - Q_{L1}), \\ \frac{dp_2}{dt} &= \frac{\beta_{e2}}{V_2}(-Q_2 + A_2 \cdot \dot{x}_p - Q_{Li} - Q_{L2}). \end{aligned} \tag{7}$$

Where, p_1 and p_2 are pressures at valve ports, Q_1 and Q_2 the valve flows, Q_{Li} the internal leakage flow, Q_{L1} and Q_{L2} the leakage flows (m^3/s), V_1 and V_2 the chamber volumes (m^3), β_{e1} and β_{e2} the effective bulk modules (Pa) of the cylinder characterized by:

$$\beta_{ei} = a_1 \cdot E_{\max} \cdot \log(a_2 \cdot \frac{p_i}{p_{\max}} + a_3), \tag{8}$$

where, $E_{\max} = 1.8 \times 10^9$ Pa, $p_{\max} = 2.8 \times 10^7$ Pa, and a_i (no unit) constant.

The volumes in Eq. (7) are calculated as:

$$\begin{aligned} V_1 &= A_1 \cdot x_p + v_{01}, \\ V_2 &= A_2 \cdot (L - x_p) + v_{02}, \end{aligned} \tag{9}$$

being v_{01} and v_{02} the pipeline volumes (m^3) at the two ports respectively, and L =1 m the maximum stroke of the piston.

The following equations describe the valve flows in Eq. (7):

$$\begin{aligned} Q_1 &= \begin{cases} c_s \cdot u_s \cdot sign(p_s - p_1) \cdot \sqrt{|p_s - p_1|}, & u_s \geq 0, \\ c_s \cdot u_s \cdot sign(p_1 - p_t) \cdot \sqrt{|p_1 - p_t|}, & u_s < 0, \end{cases} \\ Q_2 &= \begin{cases} c_s \cdot u_s \cdot sign(p_2 - p_t) \cdot \sqrt{|p_2 - p_t|}, & u_s \geq 0, \\ c_s \cdot u_s \cdot sign(p_s - p_2) \cdot \sqrt{|p_s - p_2|}, & u_s < 0, \end{cases} \end{aligned} \tag{10}$$

being c_s the flow constant ($m^3 s^{-1} v^{-1} Pa^{-1/2}$), p_s the supply pressure, and p_t the tank pressure. [8]

The internal leakage flow in Eq. (8) is calculated by:

$$Q_{Li} = L_i \cdot (p_2 - p_1), \tag{11}$$

being L_i the laminar leakage flow coefficient ($m^3 s^{-1} Pa^{-1}$).

When designing an optimal controller based on estimated state parameters, the consideration of the internal leakage flow between chambers of cylinder is enough in the system model. For a more accurate model, the external leakage model is considered.

The model of the external leakage flows in Eq. (8) was built as follows:

$$\begin{aligned} Q_{L1} &= l_1 \cdot (p_1 - p_t), \\ Q_{L2} &= l_2 \cdot (p_2 - p_t), \end{aligned} \tag{12}$$

being l_1 and l_2 the laminar leakage flow coefficients ($m^3 s^{-1} Pa^{-1}$).

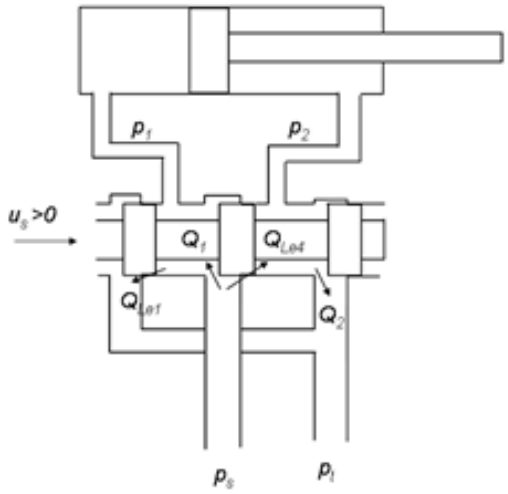

Figure 11: Pressure flow in the valve

3.2 Position-sensing hydraulic cylinder

The position-sensing feature in the position-sensing cylinder provides instantaneous analog or digital electronic position feedback information from the cylinder that indicates the amount of rod extension throughout the range of stroke. The rod have a maximum stroke length of 1 meter.

3.3 Magnetostrictive transducer based on Widermann effect

The position of the slider in this project is sensed based on Magnetostrictive transducer based on Widermann effect. Magnetostrictive materials convert magnetic energy to mechanical energy and vice versa. As a Magnetostrictive material is magnetized, it strains; that is it exhibits a change in length per unit length. Conversely, if an external force produces a strain in a Magnetostrictive material, the material's magnetic state will change.

This bi-directional coupling between the magnetic and mechanical states of a Magnetostrictive material provides a transduction capability that is used for both actuation and sensing devices. Magnetostriction is an inherent material property that will not degrade with time. An Magnetostrictive effect used in devices is the Widermann effect, a twisting which results from a helical magnetic field, often generated by passing a current through the Magnetostrictive sample.

The existence of Widermann effect leads to two modes of operation for Magnetostrictive transducers: (1) transferring magnetic energy to mechanical energy and (2) transferring mechanical energy to magnetic energy. The first mode is used in design of actuators for generating motion and/or force, and in design of sensors for detecting magnetic field states. The second mode is used in design of sensors for detecting motion and/or force in design of passive damping devices, which dissipate mechanical energy as magnetically and/or electrically induced thermal losses, and in design of devices for inducing change in a material's magnetic state.

The hydraulic slider use in this project is a one cylinder. The mass of the cylinder is mounted to a rail. The force sensor is mounted to the mass by a 90 degree rectangular metal plate. (See Figure 12)

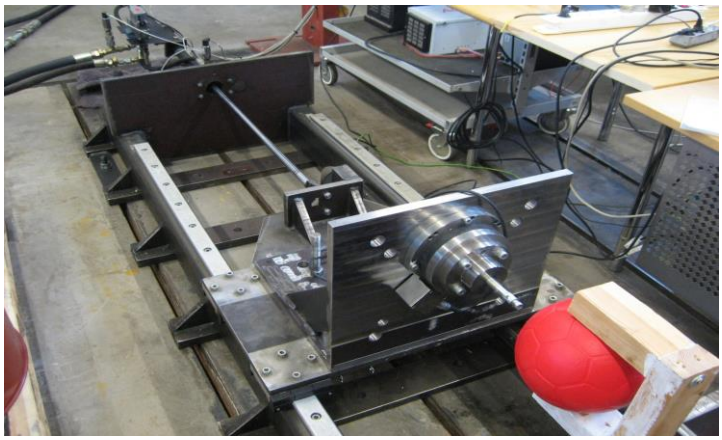

Figure 12 The hydraulic slider used in this project

4. Joystick

The joystick used in the project for position control is FORCE 3D PRO (See figure 13) produce by Logitech. It is UBS connectable. LabVIEW has a build-in package for joysticks. That package was modified to meet the control need of the project. Of all the configurations that the joystick has, only the y-axis was used. The slider lies in a horizontal position mounted to a rail with only 1-DOF, it does not twist nor rotate. The movement is restricted in the plus-minus y-axis. Matlab Simulink understands the codes that control the joystick. In fact there are several ways to use the joystick in Matlab

Figure 13: Logitech FORCE 3D PRO joystick

4.1 Joystick position against slider position

The reason for using the joystick is to be able to control the speed of the slider and to have the effect of the force feedback as the ball come in contact with the sensor. The logic behind this idea is that the feedback that is generated can limit accidents and help the operator to be more award and cautious. It tells that something is going wrong and is the operator still in control.

5. Force Sensor

A load cell is a transducer that is used to convert a force into electrical signal. This conversion is indirect and happens in two stages. Through a mechanical arrangement, the force being sensed deforms a strain gauge. The strain gauge measures the deformation (strain) as an electrical signal, because the strain changes the effective electrical resistance of the wire.

A load cell usually consists of four strain gauges in a Wheatstone bridge configuration. Load cells of one strain gauge (Quarter Bridge) or two strain gauges (half bridge) are also available. The electrical signal output is typically in the order of a few millivolts and requires amplification by an instrumentation amplifier before it can be used.

The output of the transducer is plugged into an algorithm to calculate the force applied to the transducer.

5.1 Selecting a Force/Torque Transducer

Moment capacity is usually the determining factor in choosing the best transducer model for our application. The end-effector attached to the transducer as well as the tasks being performed will generate forces on the transducer, which will result in a moment. The moment is the applied force (dynamic and static together) multiplied by the distance from the transducer origin to the point at which the force is applied. It is important to also consider overload conditions beyond the normal operating forces and moments the transducer will experience.

5.2 Transducer strength and resolutions

The first step to choose a proper transducer is to identify the transducer strength with attention to the application necessities. Next, the required resolution should be considered. A fine resolution requirement can conflict with a transducer chosen based on moment capacity. Transducers with larger ranges have coarser resolutions.

After studying the application needs in this project and the available force sensor catalogues, OMEGA 160 from famous sensor producer, ATI Industrial Automation Co, was chose. Table 2 shows the main specifications of OMEGA 160 IP60.

Table 2: OMEGA 160 IP60

Model	Max Fx, Fy*	Max Tx, Ty*	Weight**	Diameter*	Height**
Omega 160 IP60***	±2500 N	±400 N-m	7,67kg	190mm	58mm

**Maximum sensing range along the axis*
***Specifications include standard interface plates*
****Ingress Protection (IP) Ratings:*

- IP60 - Ingress Protection Rating "60" designates protection against dust

After choosing the proper transducer the detailed specifications of the chosen transducer should be compared to those of our application requirements to be certain the chosen transducer is appropriate for the application.

5.3 ATI Multi-Axis Force/Torque Sensor system

The ATI Multi-Axis Force/Torque Sensor system measures all six components of force and torque. It consists of a transducer, shielded high-flex cable, and intelligent data acquisition system, Ethernet/DeviceNet interface or F/T controller. Force/Torque sensors are used throughout industry for product testing, robotic assembly, grinding and polishing. In research area ATI sensors are used in robotic surgery, haptics, rehabilitation, neurology and many others applications.

Transducer measuring and outputting forces and torques from all three Cartesian coordinates (x, y and z). A six-axis force/torque transducer is also known as a multi-axis force/torque transducer, multi-axis load cell, F/T sensor, or six-axis load cell. Figure 14 illustrates the 3D schematic and interface view of a typical transducer.

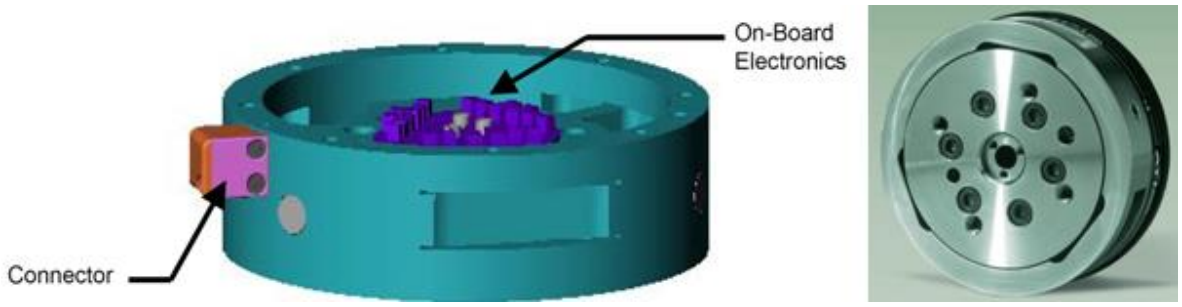

Figure 14: Schematic view of the transducer

5.4 Description of the force Sensor

The Force/Torque (F/T) sensor system measures the full six components of force and torque (F_x, F_y, F_z, T_x, T_y, T_z) using a monolithic instrumented transducer. The F/T transducer uses silicon strain gauges for excellent noise immunity. The use of silicon gauges allows the F/T transducer to have high stiffness and increased overload protection. The transducers are equipped with either DAQ F/T or Controller F/T interfaces.

The DAQ F/T allows the transducer to connect to an analog Data Acquisition (DAQ) card (PCI, USB, PCMCIA, etc.) making it easy to read sensor data with the PC or robot controller. The F/T strain gauge signals are conditioned and transmitted to the DAQ card. Next, the ATI DAQ software works with a computer to convert strain gauge data into force/torque data. The DAQ F/T consists of a transducer, an interface board, a power supply board, a DAQ card, software and long-life flexible cables designed to shield against outside electrical noise. The Controller F/T processes the F/T strain gauge information and outputs serial and analog force/torque data. Controller functions provide tool transformations, peak capture, biasing and discrete I/O. (Roozbahani, 2011, p. 97)

The chosen transducer also has other beneficial specifications which had positive effects such as: Overload protection, High signal-to-noise ratio, High-speed output, Software Tool Transformations, Versatile Outputs and Temperature Compensation.

5.5 Multi-Axis Force/Torque Sensor system components

Multi-Axis Force/Torque Sensor system is consisted of:

Transducer: The transducer senses applied loading with six degrees of freedom (F_x, F_y, F_z, T_x, T_y, and T_z). OMEGA 160 transducer models have the interface board inside the transducer. Output is un-calibrated. ATI software must be used to produce calibrated output.

Transducer Cable: For other transducers the transducer cable is attached with a connector. The transducer cable is a long-life flexible cable specially designed for noise immunity. This cable protects the transducer signals from electrical fields and mechanical stress.

Interface Board: The interface board electronics receive transducer gauge signals and convert them to readable. DAQ card signals using noise immunity technology. Each interface board is calibrated to mate to a specific transducer. The interface board is mounted within the OMEGA 160 transducer and is located in the interface power supply box (IFPS). Since transducer output is un-calibrated, ATI software must be used to produce calibrated output.

Power Supply: The power supply converts readily available 5 volt (275mA) power from the PC through the DAQ card connection to regulated power used by the transducer. The power supply is mounted in a small box that connects to the transducer cable on one end and to the data acquisition card on the other. When not mounted on the transducer, the interface board is mated directly to the power supply.

Power Supply Cable: The power supply cable conducts 5 volt power to the power supply box or interface power supply box and transmits the transducer signals to the data acquisition card. The cable is a flexible long-life design with special noise immunity features.

Data Acquisition (DAQ) Card: The data acquisition card plugs into the PC, receives the analog transducer signals via the power supply cable and (with ATI software on the computer) converts them into data to be used by computer programs. The DAQ F/T outputs amplified, conditioned strain gauge signals to a data acquisition card—not the resolved force and torque data. ATI software (included) running on the host computer performs

computations to convert the strain gauge voltage data into force/torque data. All six strain gauge channels must be acquired in order to calculate any of the forces and torques. (Roozbahani, 2011, p. 98)

5.6 Interface Plates

While standard systems provide all the necessary components for measuring force and torque, also there are options available which may aid in interfacing the F/T sensor system with special applications. All F/T transducers come with standard interface plates. Some models have threaded holes patterns machined into both sides that are used for attaching to the other equipments. Others have a threaded holes pattern on the tool side and a blank plate on the mounting side. The blank plate is machined by the customer to accommodate specific mounting requirements.

5.7 OMEGA160

ATI force/torque sensors use simple ActiveX controls that make it compatible with Open Robot Control Architecture. Ease of integration, rugged design and excellent performance.

OMEGA 160 has mounting plate bored for a 40mm through-hole in our cases which fitted with dust protector. In this transducer, EDM wire cut from high yield-strength stainless steel which gives maximum allowable overload values are 4.2 to 14.4 times rated capacities. Silicon strain gauges provide a signal 75 times stronger than conventional foil gauges. This signal is amplified, resulting in near-zero noise distortion. An IP60 version is for use in dusty environments. Figure 15 shows the Omega160 F/T transducer which is made of hardened stainless steel, and the tool and mounting adapters are made of high strength aircraft aluminum.

Figure 15: Omega 160 F/T Transducer

Some of the applications of this sensor are: rehabilitation research, product testing, orthopedic research, robotic assembly, tele-robotics, part placement and removal in precision fixtures.

Table 3: OMEGA 160 data sheet

Single-Axis Overload	Metric
F_x and F_y	±18000 N
F_z	±48000 N
T_x and T_y	±1700 Nm
T_z	±1900 Nm
Stiffness (Calculated)	**Metric**
X-axis & Y-axis force (F_x, F_y)	7.0×10^7 N/m
Z-axis force (F_z)	1.2×10^8 N/m
X-axis & Y-axis torque (T_x, T_y)	3.3×10^5 Nm/rad
Z-axis torque (T_z)	5.2×10^5 Nm/rad
Resonant Frequency (Non-IP rated, Measured)	
F_x, F_y, T_z	1300 Hz
F_z, T_x, T_y	1000 Hz
Physical Specifications	**Metric**
Weight*	2.7 kg
Diameter*	160 mm
Height*	55.9 mm

*Specifications include standard interface plates and are for non-IP rated models.
Diameter excludes any connector block.

Figure 16 illustrates the tool side view reference origin of the tool with the sensing reference frame origin.

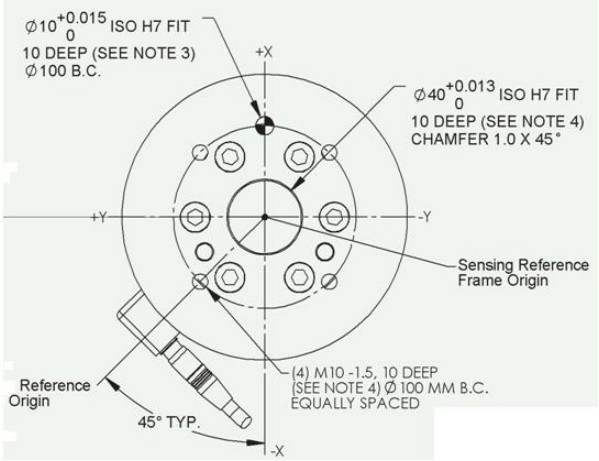

Figure 16: Side view of the transducer tool

5.8 Transducer

The transducer is a compact, rugged monolithic structure that converts force and torque into analog strain gage signals. The transducer is commonly used as a wrist sensor mounted between a robot and a robot end-effector. Figure 17 shows the transducer with a standard tool adapter.

The transducer is designed to withstand extremely high overloading through its use of strong materials and quality silicon strain gages. OMEGA160 use a hardened stainless steel with twice the strength of titanium for overload protection while other transducers use mechanical overload pins to prevent damage.

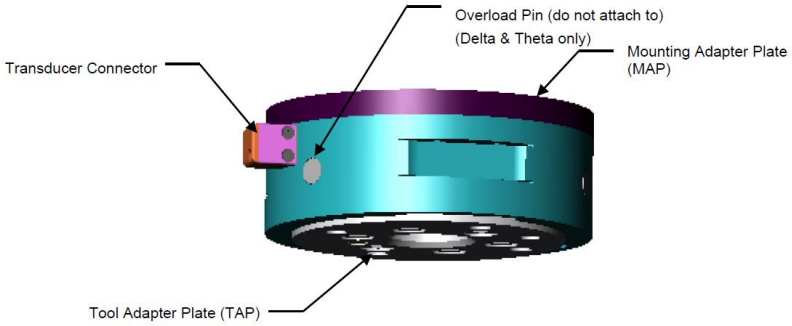

Figure 17: Standard tool adapter of transducer

5.9 Mechanical functionality

The property of forces was first stated by Newton in his third law of motion: *For every action there is always an opposed or equal reaction; or, the mutual action of two bodies upon each other are always equal, and directed to contrary parts.* The transducer reacts to applied forces and torques using Newton's third law. The force applied to the transducer reflexes three symmetrically placed beams using Hooke's law:

$$\sigma = E \,.\, \varepsilon \tag{13}$$

σ = Stress applied to the beam (σ is proportional to force)

E = Elasticity modulus of the beam

ε = Strain applied to the beam

The transducer is a monolithic structure. The beams are machined from a solid piece of metal. This decreases hysteresis and increases the strength and repeatability of the structure.

The resistance of the strain gage changes as a function of the applied strain as follows:

$$\Delta R = S_a \cdot R_o \cdot \varepsilon \tag{14}$$

ΔR = Change in resistance of strain gage

S_a = Gage factor of strain gage

R_o = Resistance of strain gage unstrained

ε = Strain applied to strain gage

Figure 18 shows the applied force and torques vectors on transducer which the forces, stresses and strains are calculated based on.

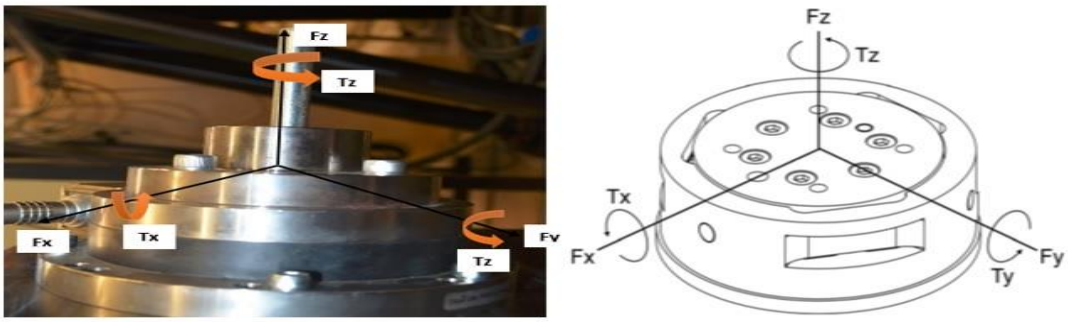

Figure 18: Transducer with torques and forces vectors

5.10 Developing the force matrix for LabVIEW

The force sensor uses a six-by-six matrix to communicate with the NI DAQ devices. The matrix is put into LabVIEW to generate the Fx, Fy and Fz forces and Tx, Ty and Tz torques. LabVIEWas a formula node which understands math script file for implementation of mathematical notation. The formula node allows direct entry of algebraic formulas into the block diagram. This feature is useful for solving long formula.

Initially it was important to know the corresponding signal for each force and torque vectors. To achieve this, the 6 by 6 matrix was divided into 6 parts of 1 by 6 matrix. Figure 19 shows the LabVIEW G code for the force matrix. The code produces the signals in figure 20.

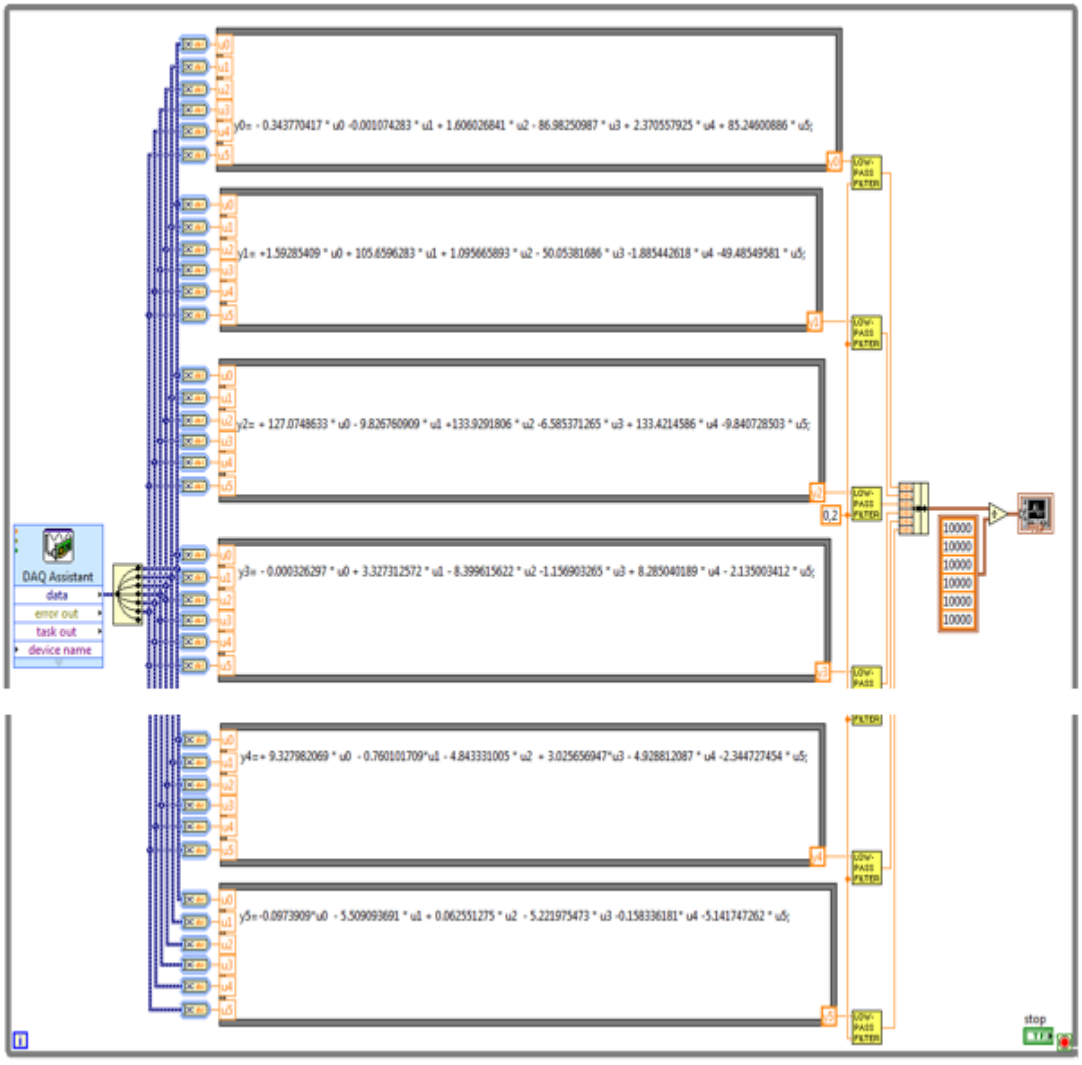

Figure 19: Force and torque vectors matrix in LabVIEW G code

The LabVIEW model for the force sensor gives the output signal for the 6 output signals Tx, Ty,Tz, Fx, Fy and Fz.

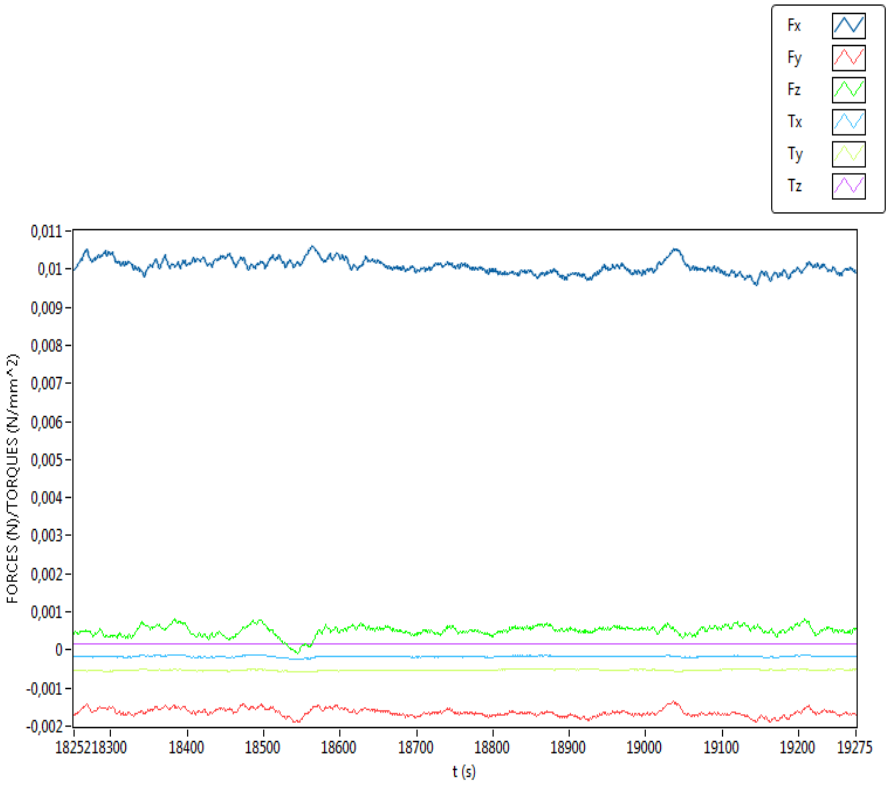

Figure 20: Unfiltered forces and torques signals from the force sensor

After it was determined which signals belongs to which vector, the other signals were eliminated because only the force of the Fz-axis is needed in this project. It is the force that makes contact with the ball. (See Figure 21 and compare Figure 18)

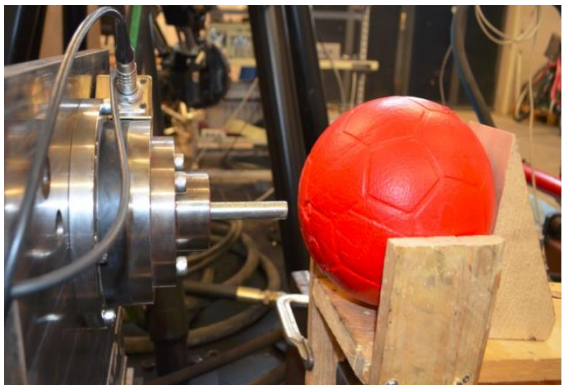

Figure 21: Force sensor mounted into the cylinder mass

It was observed that the output signals were very noisy. The source of the noise varies from the sensor resolution to the cabling connection or connection of other advices in the circuit to the force sensor itself. The filter can be eliminated by filtering the signal. A low pass filter is used to eliminate the noise.

LabVIEW has many build in filters that can be used. Though these filters did eliminate the noisy, they caused so much delay to the system. This is not good for real-time performance because real-time mean absolute reliability. So using the formula node, a lower pass filter was built that reduces the delay and give a reasonable time response.

The output after applying the lower pass filter is shown in Figure 22. The selection of the proper filter is very important. It should have good noise compensation and very little response delay.

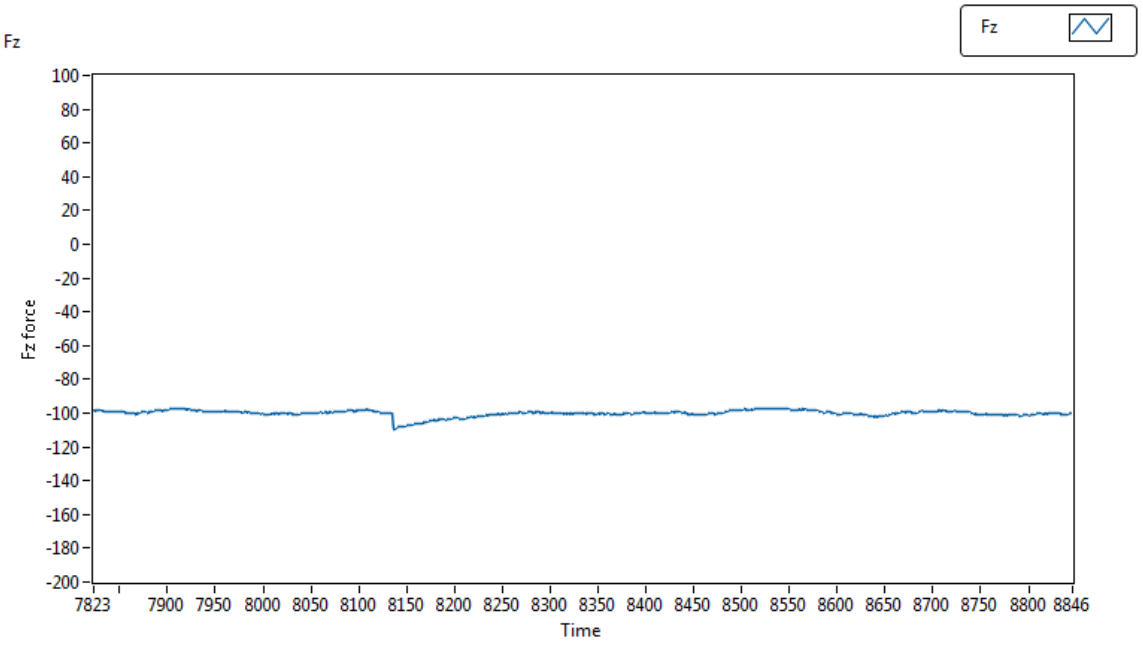

Figure 22: Fz signal before applying low pass filter

6. NI and dSPACE hardware

6.1 NI PXI-1031

PXI is a rugged PC-based platform for measurement and automation systems. PXI combines PCI electrical-bus features with the modular, Eurocard packaging of CompactPCI and then adds specialized synchronization buses and key software features. PXI is both a high-performance and low-cost deployment platform for applications such as manufacturing test, military and aerospace, machine monitoring, automotive, and industrial test. It was first developed in 1997 and launched in 1998. PXI is an open industry standard governed by the PXI Systems Alliance (PXISA), (NI, 2014)

The NI PXI-1031 used for the study, has three chasises boards and multiple I/O connections including USB, LAN and Ethernet. It runs on a driver that has to be installed separately. And it is very important to know that the version of the driver most be compactable with that of the NI software. (See Figure 23: PXI-1031).

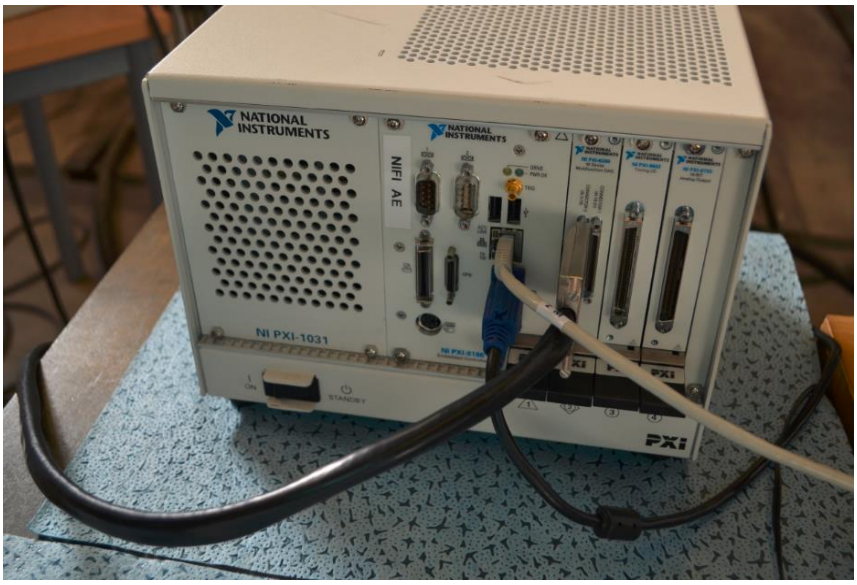

Figure 23: NI-PXI 1031 with embedded NI-PXI 8186

6.2 NI USB-6259

The NI USB-6259 used has 16 differential BNC analog inputs of 16-bits, 4 BNC analog outputs, 48 digital I/O and two 32-bit counters. (See Figure 24: NI USB-6259) It only source of communication is via a USB cable which makes it easy to use. The device is compatible with LabVIEW, ANSI C/C++, and Visual Basic. It is easy to connect and the driver is easy to install and up-date.

For this project, all I/O were connected by BNC cable. Six inputs were used for the force sensor and one input one output for the hydraulic slider.

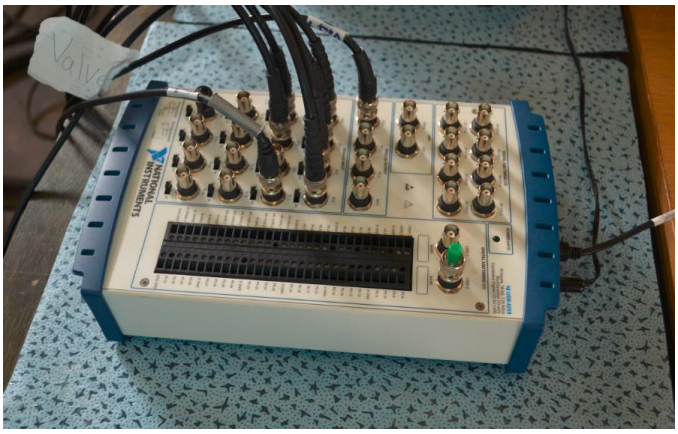

Figure 24: NI-USB 6259

6.3 dSPACE DS1005

dSPACE is a single-Board Hardware for building a complete real-time control system with just one controller board. Figure 25 shows a dSPACE board hardware. The DS1005 is compatible with all of its earlier versions. Programs or models compiled for the older versions run on the DS1005 without modifications. Multi-processor systems can mix old and new DS1005 boards without impairing performance. The DS1005 PPC board can easily be program from Simulink via real-time interface. A set of Simulink blocks gives access to the entire range of dSPACE I/O boards. Dialog for altering simulation parameters such as solver options and simulation time, and for generating C code, are directly accessible from the Simulink environment. Multiprocessor system can be set up via RTI-MP blocks.

Figure 25: dSPACE DS1005 single control board

Controller development and rapid control prototyping are ideal application areas for the DS1005 Controller Board Controller Board. The DS1005 (ISA bus) can be installed in an AutoBox or a dSPACE Expansion Box.

The key benefits of using dSPACE board in this project is that, DS1005 combine compactness with high performance and a high number of features. Used with RTI, the controller boards are fully programmable from the Simulink block diagram environment. The controller board contains a real-time processor for fast calculation of the model, and a variety of I/O interfaces that allow to carry out the control developments.

In this project a DS1005 controller board was used. This board is a versatile controller board with an unparalleled number of I/O modules, including CAN, and ample calculation power for control prototyping. The unparalleled number of I/O interfaces makes the DS1005 a versatile controller board for numerous applications. The control of electrical drives requires accurate recording and output of I/O values. It is possible to synchronize the A/D channels and D/A channels, and the position of the incremental encoder interface, with an internal PWM signal or an external trigger signal. Also, the serial interface (UART) is driven by a phase-locked loop to achieve absolutely accurate band rate selection.

7. Experiments

All the experiments carried out during the course of the project were done at LUT laboratory of intelligent machines. Before the experiments can be started, all factors that affect the performance of the system must be thoroughly checked and running properly.

7.1 Hydraulic power supply or Pump

The hydraulic pump supplied pressure range is 0-100 bar and the power supply range is also 0-100 volts. Base on empirical knowledge, the pressure and power needed must be set at 40 bar and 40 volts respectively. A motor drives the hydraulic pump which draws hydraulic oil from the reservoir and pressurizes it to the set pressure. The pressurized fluid is supply the valve through a set of flexible hoses.

7.2 Power supply to the Magnetostrictive displacement Sensor and Force Sensor

The magnetostrictive sensor which is used to measure linear position is supply a 24volt. It basically senses the slider position to determine the distance between the permanent magnet and the sensor head. The main components of the magnetostrictive sensor are:

- Position magnet
- Strain pulse detection system
- Waveguide and
- Damping module

The waveguide which is the basic part of a Magnetostrictive sensor is made of ferromagnetic materials such as iron, nickel, cobalt and their alloy. When a position has to be determined the sensor electronics sends a current pulse through the waveguide and its starts the timer. A magnetic field is created around the waveguide. When the magnetic fluid of the position magnet interacts with the magnetic field around the guide a strain pulse is generated which travels at the speed of sound on both sides. On one side this strain pulse is detected by the strain pulse detection system and then processed by the electronic and

converted into electrical pulse. The position is determined based on the time the strain pulse takes to reach the strain pulse detection system. Figure 26 shows the Magnetostrictive sensor used in this project mounted to the side of the rail on which the mass of the hydraulic cylinder rest.

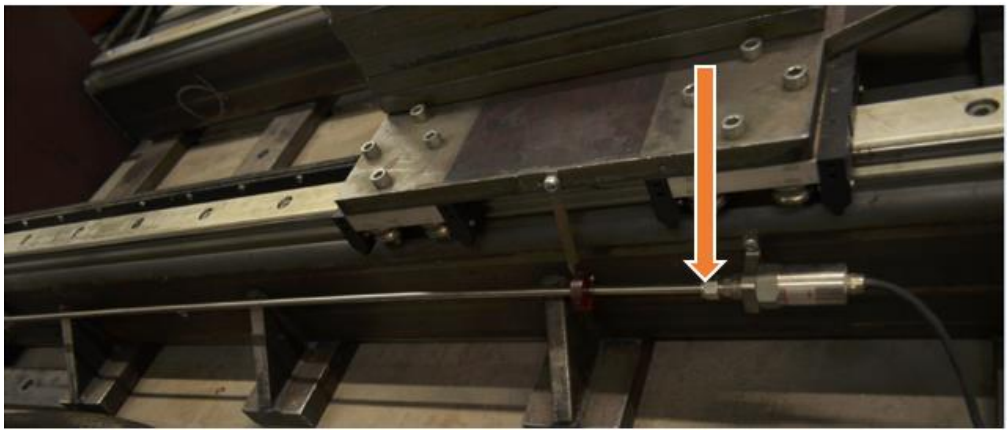

Figure 26: Magnetostrictive displacement transducer Model: Gystc-03

7.3 PID tuning

PID controllers are probably the most commonly used controller structures in industry. They do however present some challenges to control the system in the aspect of tuning for the gains required for stability and good transient performance. There are several rules used in PID tuning. One example is that proposed by Ziegler and Nichols in the 1940's.

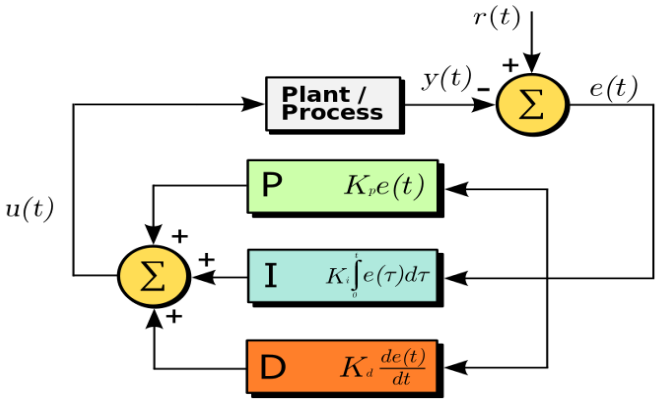

Figure 27: PID control in feedback loop

7.3.1 Ziegler-Nichols method

As a result of empirical tests on a wide variety of process plant, Ziegler and Nichols propose a simple rule of thumb procedure for estimating the values of the controller setting K_p, K_i and K_d for existing operating plant in order to achieve an optimum transient response. There are two methods, one based on the step response of the open loop system and the other based on information obtained at the stability limit of the process under proportional control. (Schwarzenbach J., 1992, p. 218)

In the first method with the loop open, the plant is subjected to a step change of manipulated variable and the resulting output response curve is characterized by two measured parameters N and L (See Figure 28). N is the maximum slope of the curve for a change M of manipulated variable and L is the time at which the line of maximum slope intersects the time axis. The recommendations which Ziegler and Nichols put forward for the controller setting are:

$$\left.\begin{array}{rl} Kc = \frac{M}{NL}, & for\ P\ control \\ Kc = 0.9\frac{M}{NL}, Ti = 3.3L & for\ P + I\ control \\ Kc = 1.2\frac{M}{NL}, Ti = 2L, Td = 0.5L & for\ P + I + D\ control \end{array}\right\} \quad (15)$$

Where Kc, Ti and Td are the parameter values of controller gain, integral action time and derivative action time respectively, as they appear in the control law

$$Gc(s) = kc\left(1 + \frac{1}{T_i s} + T_d s\right) \quad (16)$$

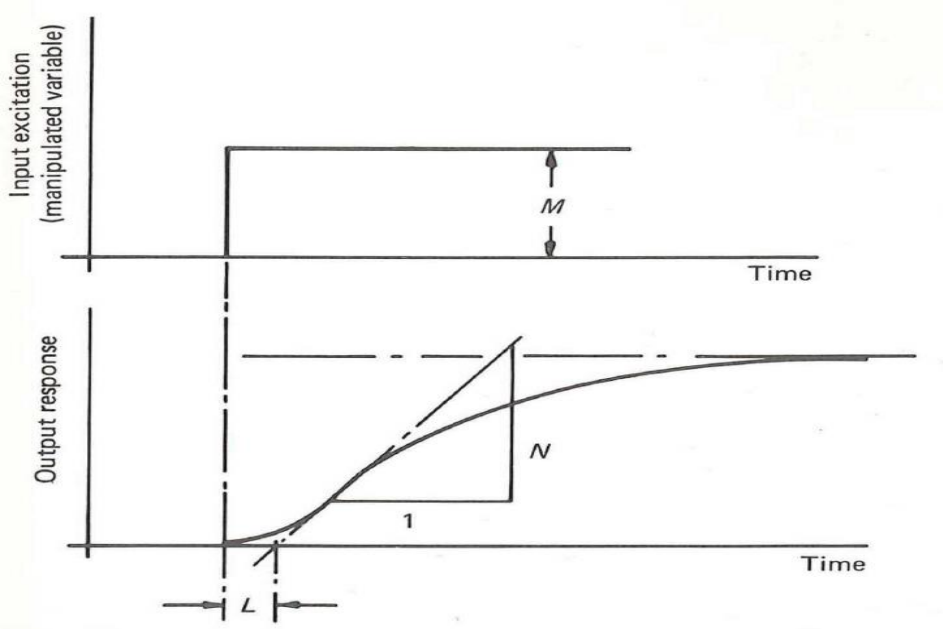

Figure 28: Ziegler-Nichols first method

The procedure of the second method is to determine experimentally the limiting condition of stability of the closed loop system under proportional control only, and to use the resulting information to calculate the controller settings. If the limiting value of the gain for stability is k_{crit} and the time period of oscillation is $P_{crit,}$ then the Ziegler-Nichols recommended controller setting are:

$$\left.\begin{array}{rl} Kc = 0.5k_{crit}, & for\ P\ control \\ Kc = 045\ k_{crit}, Ti = 0.83P_{crit} & for\ P + I\ control \\ Kc = 0.6k_{crit}, Ti = 0.5P_{crit}, Td = 0.125P_{crit} & for\ P + I + D\ control \end{array}\right\} \quad (17)$$

7.4 PID tuning trial and error method

PID tuning is the process of finding the proper values of $K_{p,}$ K_i and K_d gains. The aim of the controller tuning is to obtain both of the following for the control system if possible:

- Fast response, and
- Good stability

Unfortunately, for practical system these two wishes cannot be achieved simultaneously. In other words:

- The faster the response, the worst the stability and
- The better the stability, the slower the response

So for the slider control system, some compromises had to be made that will still give the system a real-time response. Real-time does not mean very fast but rather absolute liability. The compromises were to have:

- Acceptable stability, and
- Medium fastness of response

The only way the goal could be achieve was by trial and error; that is, we run the system and set all gains to zero. Little by little the gains are change until the desire state is met. One may ask how we classify acceptable stability more specifically. There is exact definition but a rule of thumb is when the setpoint assume a positive step change. Acceptable stability is when the undershoot that follows the first overshoot of the response is small or barely observable.

8. Results

8.1 System Outputs from LabVIEW DAQ device and dSPACE

For analysis and design certain basic reference inputs function are chosen, mainly to bring the analysis into regions of reasonably simple mathematics, but often they may also represent the most typical or the most severe form of disturbance to which the system is subjected.

Traditional experimental procedures involve subjecting the system to step, ramp, pulse or sinusoidal inputs variations and then carrying out relatively simple analysis of the output response curves. The advantages of these test inputs is not all are applicable in every working condition, where a step function is undesirable or is not physical possible a ramp function, or step change in velocity can be used. (Schwarzenbach J., 1992, p. 107) A parabolic input function or step change in acceleration could be used in situations where even the ramp input is too severe.

8.1.1 Sinusoidal Input

This result was generated with the NI system using a sine wave as reference input. With the right PID parameters, the system was able to perfectly follow the input signal. It is not always easy to achieve such result with the PID tuning. The tuning can be tedious since there is no guarantee that certain parameters will produce the desire response.

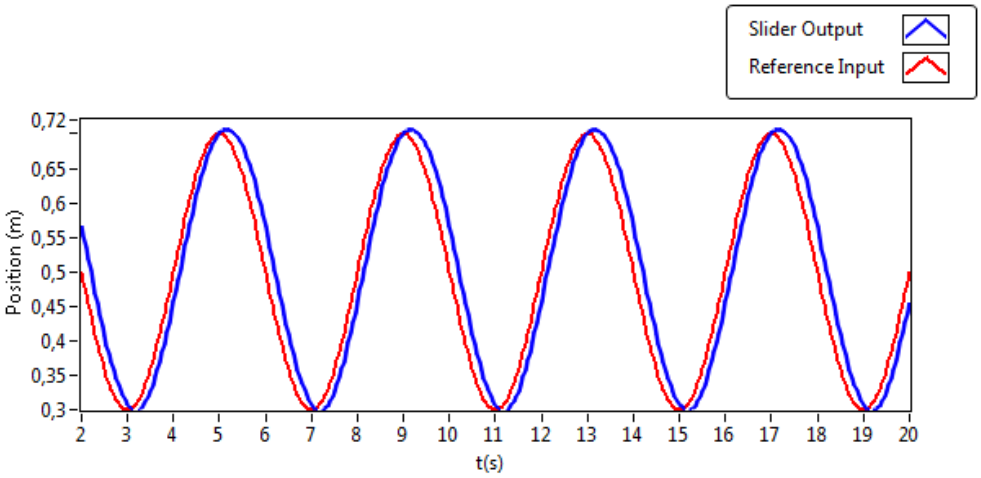

Figure 29 System response to Sinus input in NI-system

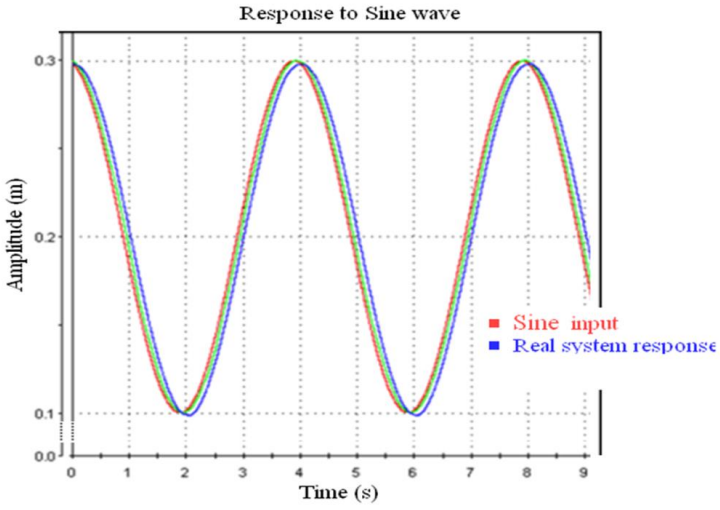

Figure 30: System response to sinus input in dSPACE

After designing the model it's compiled as a real-time model to dSPACE environment. In dSPACE environment the read position from joystick device is applied to this model and

then the real position of the slider is applied to the controller optimization to prepare the most optimized PID controller. Figure 30 shows the slider position has it follows the input signal in sine wave. In this figure the red curve is the position from joystick and the blue curve is the real position of the slider in same time. So the figure is a proof for successful modeling and communication between joystick, servo valve, slider and all written codes and algorithms.

8.1.2 Pulse Input

The pulse signal is affected by other parameter besides the PID. The duty cycle, period, amplitude and frequency and noise all affect the output of the signal.

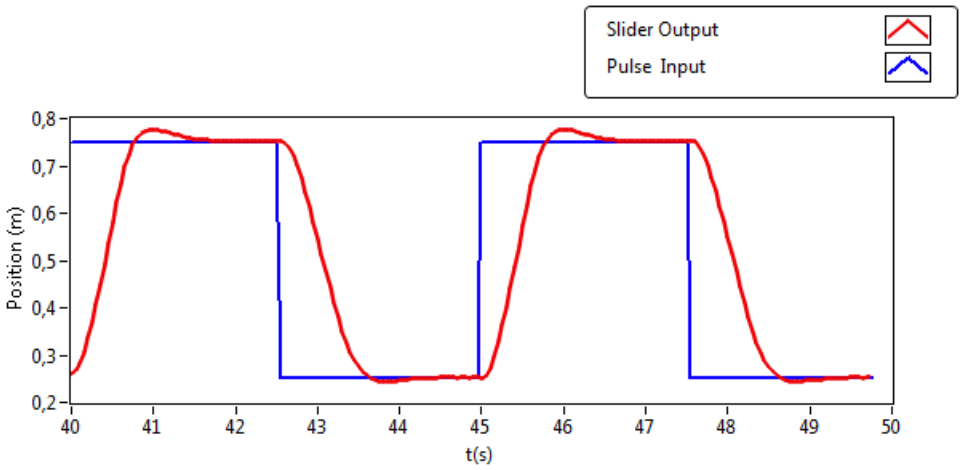

Figure 31: System response to pulse input in NI system

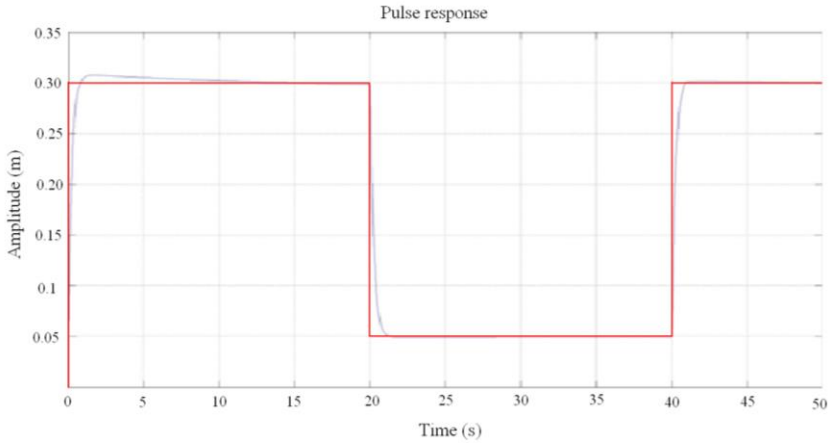

Figure 32: System response to pulse input in dSPACE

8.1.3 Ramp Input

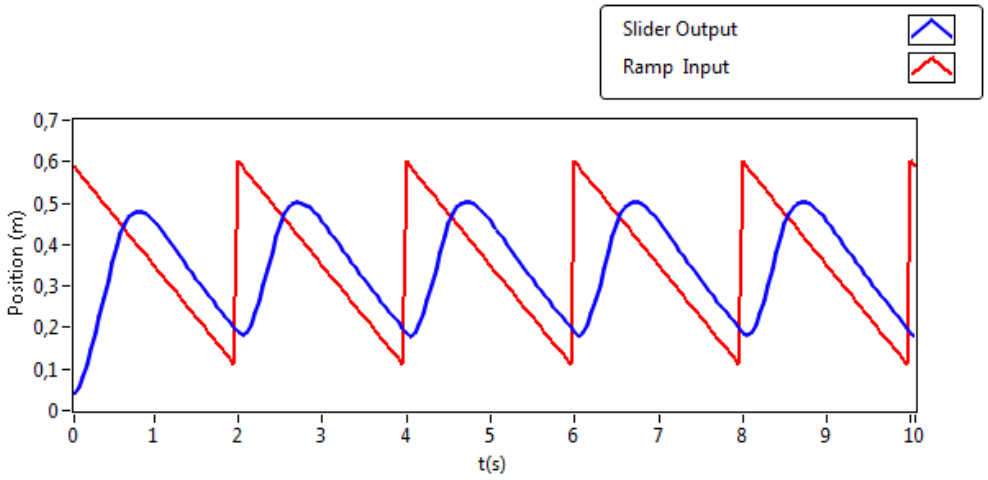

Figure 33: System response to ramp input in NI system

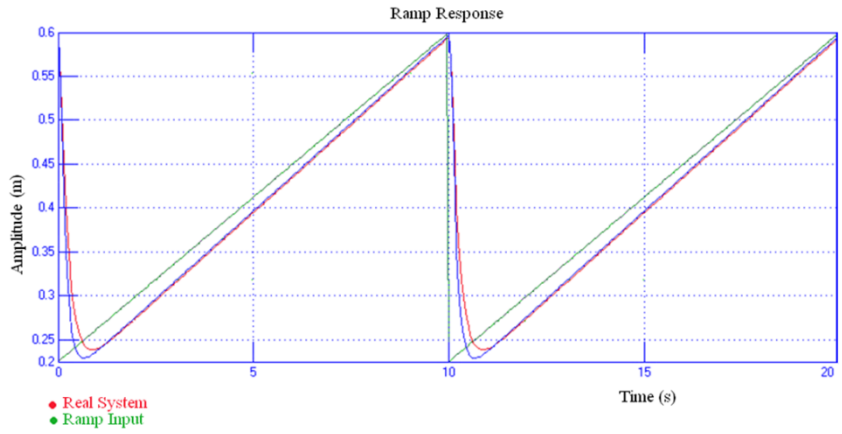

Figure 34: System response to ramp input in dSPACE

8.1.4 Step Input

When concerned with dynamic systems it is of interest to know how the output of the system will change as a result of specific types of input change. On the basic of some appropriate criterion, an assessment can be made of whether or not the system behavior is satisfactory and if not what can be done to improve the response. (Schwarzenbach J., 1992, p. 31)

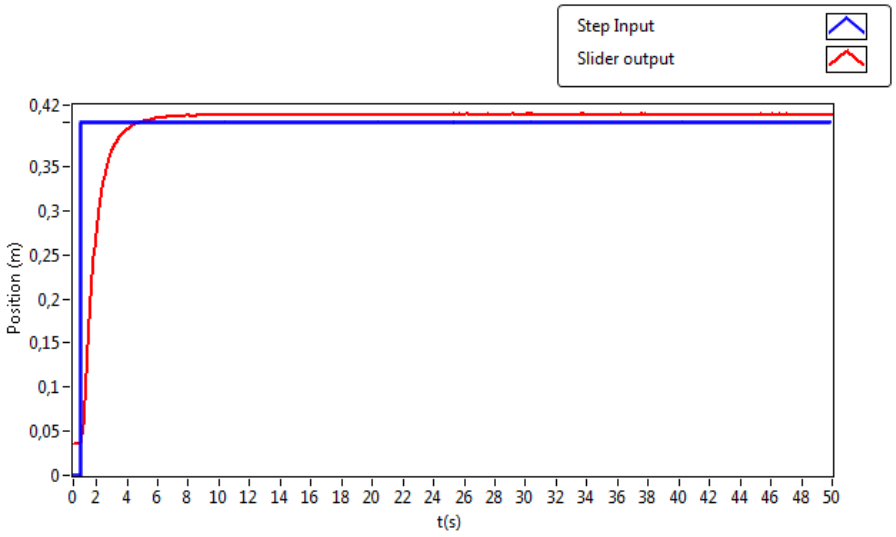

Figure 35: System response to step input in NI system

The step response in dSPACE was achieved by using a linear PID controller. The response of the real system is compared with the model. In spite of some small differences between modelled and real system because of effect of external disturbances such as friction forces on real system, the model system followed the real system accurately. This output is achieved by applying a simple linear PID controller to both systems. In order to design the nonlinear intelligent PID tuned controller the combination of PSO and bacterial foraging algorithms has been used. (Roozbahani, 2011, p. 60)

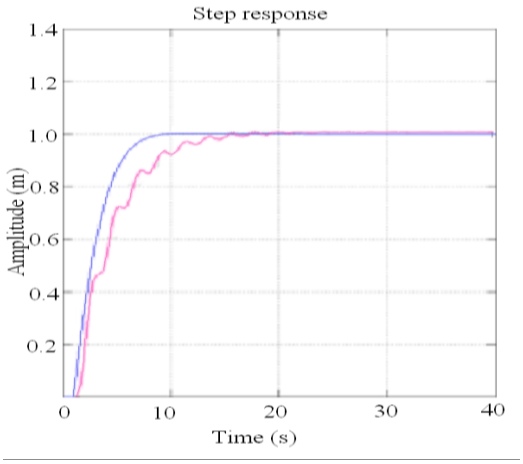

Figure 36: System response to step input in dSPACE

8.1.5 Joystick and Inputs from NI system and dSPACE

During the test it was noticed that the FORCE 3D PRO joystick locks a lot of stability to perform the task property. It was impossible to achieve a smooth translation with the advice. Nonetheless the slider was still able to follow as close as possible with good PID parameters. The result could have been better with an advice with higher precision and more stability.

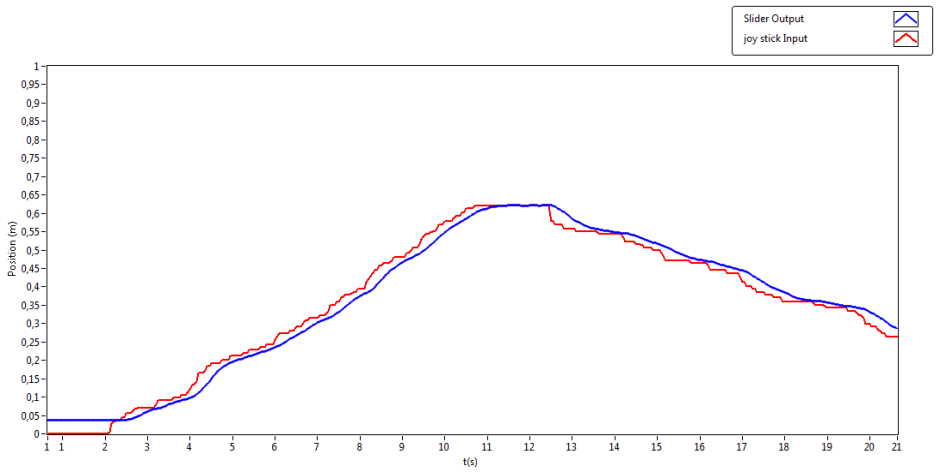

Figure 37: System response to Joystick input in NI system

The model from Simulink is compiled as a real-time model to dSPACE environment. In dSPACE environment the read position from joystick device is applied to this model and then the real position of the slider is applied to the controller optimization to prepare the most optimized PID controller with the aim of the slider position trying to follow the joystick.

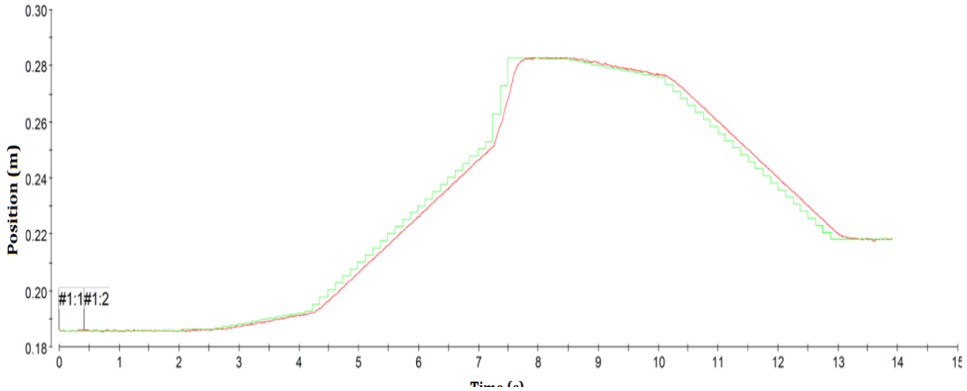

Figure 38: System response to joystick input in dSPACE

The red curve represents the position from joystick and the green curve is the real position of the slider at the same time. The result proofs a successful modeling and communication between joystick, servo valve, slider and all written codes and algorithms.

8.2 Systems response from Force Sensor

8.2.1 NI system Contact with ball

Prior to any contact with the ball the force signal is at zero. As the sensor comes in contact, there is a rise in peak indicating the amount of compressive force exalted on the ball. According to the manufacturer catalogue, the OMEGA160 is built to generate up to ±2500N around the Fz-axis. Due to the nature of the surface of contact, a maximum force of 140N could be achieved during the test. The compressive force was multiplied by -1 that is why it shows positive in figure 38. As the sensor, detaches from the ball (See Figure 39), the force sensor signal respond by gradually descending to the initial position of zero. In this graph, the change in position is against time.

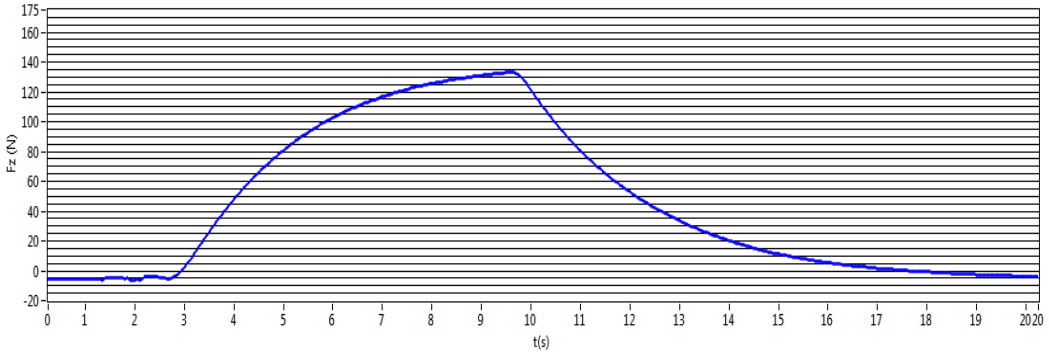

Figure 39: Force sensor response to external force with ball

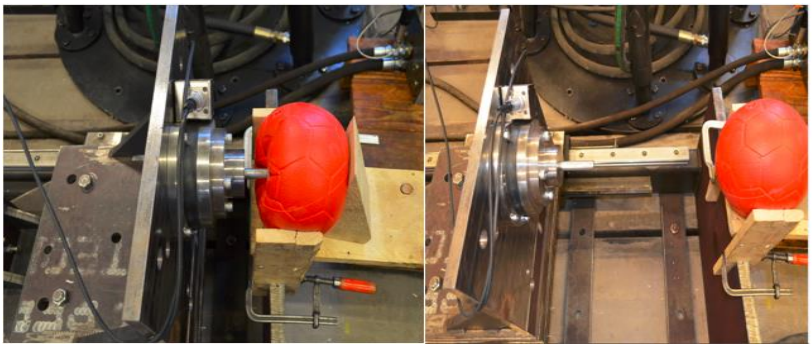

Figure 40: Force sensor at contact and after contact with the ball

8.2.2 Joystick contact with ball in dSPACE system

In here the change is with respect to position not time. What this means in reality is that since the ball is located at the end of the slider, as the sensor makes contact with the ball, the force starts to increase and the distance also increases but not much. It increases only by the length of the tip of the sensor which is about 75mm that protrudes beyond the 1 meter length of the slider.

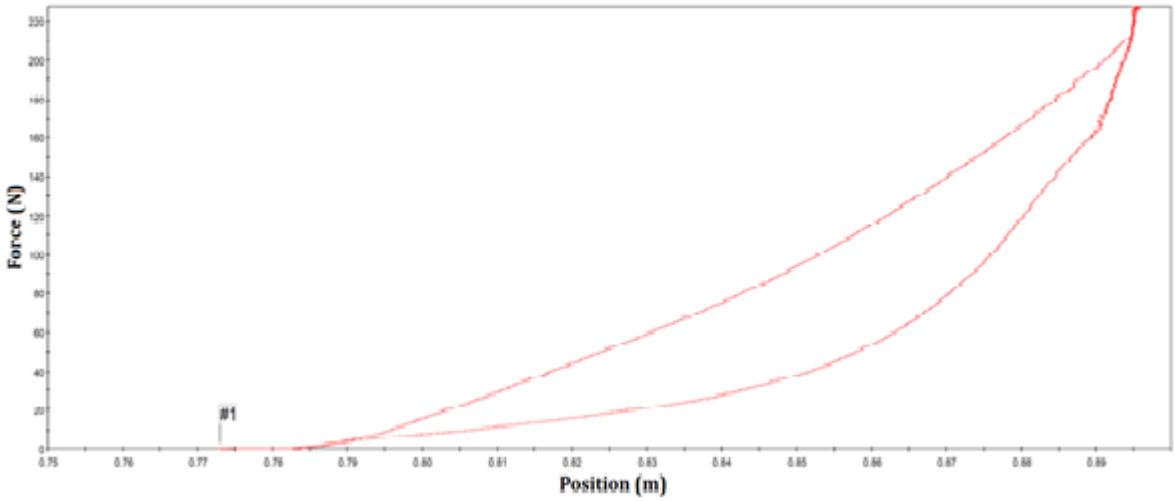

Figure 41: Force sensor response to external force in dSPACE

8.3 Drawback of the result

The primary reason for using the joystick in the project was to get the haptic feedback as the ball comes in contact with the ball. This could be in the form of force, vibration or motion to the operator. In teleoperators and simulators when contact force is produce to the operator it is called haptic teleoperation. (Haptic Technology, 2014). This keeps the operator alert and focus reducing accidents and injuries.

During the course of the project, considerable amount of afford was made both on the NI and dSPACE systems to achieve this effect but to no avail. One of the possible reason of the failure could be the rumble in the joystick is not that sophisticate for this application. Another suggestion was to open the joystick and directly connect the potentiometer and connect the correct pin to the BNC cable. This approach was found to be too tedious and was not attempted.

9. Discussion and Conclusion

One of the major attributes of Labview is that it is a visual programming language. It uses the graphical language code called G code, which is a dataflow programming language.

Data can be inputted or extracted from the running virtual instrument, and since the front panel can be used as a programmatic interface. The VI can be run as a program or define the inputs and outputs of a node. In other words, a software engineer has the option to either program certain parts by themselves or use the already pre-programmed components that LabVIEW offers. For example LabVIEW offers pre-programmed filters, PIDs, joystick controller and many more. Also the graphical option makes it possible for non-programmers to build programs by simply connecting and moving the virtual representation of the tools that they are familiar with. The LabVIEW development environment allows the easy creation of small programs with the help of the examples and documentation provided. It has a good library that helps new user get jump-started with ready to use programs.

One of the reasons why LabVIEW became very successful over the years is the benefits that it provides to all its users regardless of skill level. A wide variety of drivers and specification for many different buses and instruments are already pre-programmed as graphical nodes. The LabVIEW library also includes an extensive number of functions for mathematics, statistics, analysis, data acquisition, etc. With the help of the graphical language where a person can better understand what is going on, a good amount of nonprogrammers are able to use the LabVIEW platform to create automated systems within a short period of time and with very little frustration, something no other programming language can brag about.

Despite many of the benefits that are achievable by using NI systems, there are some weak points which should be considered. One area of concern is about the activation process. A newly purchased NI product has to be activated either by internet or phone which makes the user highly dependent on the vendor, which might result in possible privacy and security problems.

Another problem is that LabVIEW G code is not really a heavy programming language and it is more an application-specific software environment for measurement and automation. Another flaw is the challenging process of setting up NI products. Though easy to use, the setup of LabVIEW requires that certain parts to be installed separately. For example, the installation of the runtime engine and library may only be setup by the administrator. This can cause problems if the user does not have permission to install any additional files on the host network. Then not to be over look is the problem of version compatibility among NI devices and software. For example programs made on a lower version of LabVIEW cannot be deployed to the newer version of VeriStand and the same goes for the drivers.

LabVIEW is definitely the first choice for instrument control, and interface design and for designing the system block diagram. But it is not really designed for serious algorithm development.

One great thing about LabVIEW is that it is supported by a single company. But it also adds some limitations to it. With C, C++, etc. researcher has so many 3rd party sources but with LabVIEW the operator is just connected to NI which creates dependency.

With LabVIEW, the paradigm of using wires instead of variables makes some sort of sense, except that for anything reasonably complex, user spends more time trying to arrange wires than does actually coding. Sometimes, finding how data flows by tracing wires can be tedious especially since user can't click and highlight a wire to see where it goes – clicking only highlights the current line segment of the wire. And since most wires aren't completely straight, you have to click through each line segment to trace a wire to the end. In normal code, it doesn't matter how far away variables are. In LabVIEW by breaking VIs down into smaller chunks, many of editor performance issues are eradicated. About the many wires in LabVIEW, it should be mention that subVI clusters and classes help a lot to limit the wirings. But the over use of SubVI leads to a bad program which can be compensated by using local and global variables.

In normal text-based code, if coder needs to add another condition or another calculation somewhere it's possible to add a line of code. In LabVIEW, if designer needs to add another calculation, he/she have to start hunting around for space to add it just as in Simulink. If coder doesn't have space near the calculation, then he/she can add it

somewhere else but then suddenly wires going halfway across the screen and back. In text based programming, when it is necessary to take something out of a statement, that's easy, just cut and paste. But in LabVIEW by cutting, all wires disappear. In C it is possible to add codes anywhere while in LabVIEW being diagram based can't write everywhere. But in C, it requires a lot of memory management which is not required in LabVIEW. Also all loops are auto indexed in LabVIEW, if data is coming from somewhere, no need to find out beforehand how much is coming. In the other words, LabVIEW benefits from data flow properties. The advantage of dataflow is that, the programmer does not have to do explicit memory allocation declarations and he/she still gets efficient use of memory.

Compiling in real-time is an important issue with programming. With LabVIEW it can take 1-5 minute for long operations. Maybe with C++ it can take even longer. However, C++ doesn't make compiles blocking, so it is possible to modify code or document code while it compiles but not with LabVIEW. Once the code is compiled to be deploy in VeriStand, any needed changes must be made from the original code and redeployed for update otherwise the program will continue to use the previously deployed codes.

Another issue is matrix in LabVIEW; creating them, replacing elements, accessing elements, any sort of mathematical expression takes time. Copy & Paste also is an important issue which is weak with LabVIEW. For example having N x M constant matrix and want to import or export data. Unfortunately, copy and paste only works with single cells. So with non-single cells copying and pasting N*M individual numbers is the only way. Exporting the matrix by copying the whole thing to other software also would not fix the problem. By copy- pasting the matrix to Excel an image of the matrix would be available.

It is admirable that LabVIEW does have awesome hardware integration and definitely eases deployment. For LabVIEW beginners, it takes time of trying to get LabVIEW to do things that they could have accomplished in much less time using a more traditional language. But lack of in-depth knowledge of the LabVIEW editor is definitely is the major problem.

With LabVIEW very less run time errors and lesser program crash happens. Crashes from problems like array indexing etc are unheard of in LabVIEW in comparison to C. Also availability of whole range of built in functions in Labview saves lot of development

time. Furthermore, it's easy to create GUI's in LabVIEW in comparison to C in which lots of time will be used in creating menu's and checkboxes where for a Non Computer Engineering background above advantages are of great use.

Labview has this behavior that When user gets deeper in LabVIEW programming and in all its shortcuts then he understands and get used to it. But unfortunately things are a bit more complicated on the LabVIEW real-time target. The initial workflow of LabVIEW is very good for non-programmers working with instrument communication. However, as user skills advance, the efficiency curve for LabVIEW begins to level off much more significantly than if working inside a text editor language. LabVIEW has very poor support for complicated problems, and as mentioned earlier, the only support comes from NI who most of the time provides very shadow explanation for complicated issues.

In comparison between LabVIEW to excel, LabVIEW graphs are way more flexible and accessible; especially in case of a fixed data file format and batch processing.

Labview is a largely functional language. It is quite possible to have functional text-based languages, but explicitly representing the data dependencies as wires does reveal an important aspect of parallelism. LabVIEW's functional nature is certainly coming in handy in the world of multi-core processing. Functional code is also safer than code relying in implicit side-effects and sequencing but it requires relying on the compiler to avoid unnecessary copies, which can be a problem when the compiler lets you down. Functionality is mostly good, and is synergistic with the rather graphical data-flow representation.

LabVIEW is better than C++ because it is vectorized, rather like Matlab. It is possible to get some of those effects in C++ by overloading, but it's already there in LabVIEW, and user doesn't have to worry about the buffer management and in-place optimizations. In Matlab user can write very concise and cryptic array expressions and translating Matlab code into Labview is a nontrivial operation.

Overall it seems that robotics is an application that does fall into LabVIEW's spot. The biggest advantage of LabVIEW for robotics is that designer can in many cases directly visualize relevant time-variation, and where the change is faster, he/she can still visualize in real-time using graphs that update every second or so. When the system is running in LabVIEW, programmer can mentally correlate in real time the displayed state of the

software with what user can see with eyes and hear with ears. The ease of making animated GUIs is crucial here. The functional aspect of LabVIEW and the intertwined nature of data flow with GUI are definite strengths of LabVIEW.

LabVIEW does very well in producer-consumer architecture for simple data acquisition and control, usually in a bench environment. It's well established that a separation-of-concerns approach to software architecture yields huge benefits in terms of code re-usability. That is to say, the advantages are greater than code re-use derived from appropriate style guidelines or simple modularization. A true separation-of-concerns approach allows for modularity on a larger scale, permitting sweeping changes to user-interface as well as changing out hardware with ZERO modifications required in other sub-systems.

10. References

[1] Burrows, C., & Edge, K. (2001). *Power Transmission and Motion Control.* London: Professional Engineering Publishing Limited.

[2] Dorf C. Richard, B. H. (2001). *Modern Control Systems.* Chicago: Prentice Hall.

[3] Dr. Johnson, D., & Professor Plummer, A. (2010). *Fluid Power and Motion Control.* UK: Hadleys Ltd.

[4] Embedded Success dSPACE. (2010). Paderborn: dSPACE.

[5] Gene, F. F., David, P. J., & Abbas, E.-N. (2002). *feedback control of Dynamic systems fourth edition.* New Jersey: Prentice Hall.

[6] *Haptic Technology*. (2014, April 13). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Haptic_technology

[7] Larsen, R. W. (2011). *LabVIEW for Engineers.* New Jersey: Prentice Hall.

[8] NI. (2014, March 15). *National Instruments*. Retrieved from National Instrument: http://www.ni.com/veristand/whatis/

[9] Roozbahani, H. (2011, March 17). *Novel force control methods in a teleoperation system of a hydraulic slider.* Retrieved from Lappeenranta University of Technology: http://www.doria.fi/bitstream/handle/10024/72534/nbnfi-fe201111075801.pdf?sequence=3

[10] Schwarzenbach J., G. K. (1992). *System Modelling And Control.* London: Hodder and Stoughton Limited.

[11] Travis Jeffrey, K. J. (2009). *LabVIEW For Everyone.* Boston: Pearson.

11. Appendix:

Force Sensor matrix in C code

vtoft.c

```
#define S_FUNCTION_NAME vtoft
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include <stdlib.h>
#include <stdio.h>
#include <_syslist.h>

static void mdlInitializeSizes(SimStruct *S)
{

   ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      /* Return if number of expected! = number of actual parameters */
      return;
   }

   ssSetNumContStates(S, 0);
   ssSetNumDiscStates(S, 0);

   if (!ssSetNumInputPorts(S, 1)) return;
   ssSetInputPortWidth(S, 0, 6);
   ssSetInputPortRequiredContiguous(S, 0, true); /*direct input signal access*/
   /* */
   ssSetInputPortDirectFeedThrough(S, 0, 1);

   if (!ssSetNumOutputPorts(S, 1)) return;
   ssSetOutputPortWidth(S, 0, 6);

   ssSetNumSampleTimes(S, 1);
   ssSetNumRWork(S, 0);
   ssSetNumIWork(S, 0);
   ssSetNumPWork(S, 0);
   ssSetNumModes(S, 0);
   ssSetNumNonsampledZCs(S, 0);
```

```
   ssSetOptions(S, 0);
}


/* Function: mdlInitializeSampleTimes =========================================*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);

}

#define MDL_INITIALIZE_CONDITIONS   /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
  static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif /* MDL_INITIALIZE_CONDITIONS */


#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  static void mdlStart(SimStruct *S)
  {
  }
#endif /*  MDL_START */

/* Function: mdlOutputs ======================================================= */

 /*********************************  declear*****************************/


/*****************************************************************************/
static void mdlOutputs(SimStruct *S, int_T tid)
{

   const real_T *u = (const real_T*) ssGetInputPortSignal(S,0); // start reading?
   real_T       *y = (double*) ssGetOutputPortSignal(S,0);  // out put data
```

```
y[0]=-0.343770417*u[0]      -0.001074283*u[1] +1.606026841*u[2]-86.98250987*u[3] +2.370557925*u[4]
          +85.24600886*u[5];
y[1]=1.59285409*u[0]        +105.6596283*u[1]+1.095665893*u[2]-50.05381686*u[3] -1.885442618*u[4] -
49.48549581*u[5];
y[2]=127.0748633*u[0]       -9.826760909*u[1] +133.9291806*u[2]-6.585371265*u[3] +133.4214586*u[4]-
9.840728503*u[5];
y[3]=-0.000326297*u[0]      +3.327312572*u[1]-8.399615622*u[2] -1.156903265*u[3] +8.285040189*u[4]-
2.135003412*u[5];
y[4]=9.327982069*u[0]       -0.760101709*u[1] -4.843331005*u[2] +3.025656947*u[3]-4.928812087*u[4] -
2.344727454*u[5];
y[5]=-0.0973909*u[0]   -5.509093691*u[1]   +0.062551275*u[2]      -5.221975473*u[3] -0.158336181*u[4] -
5.141747262*u[5];




} //end of main program


/*********************************************************************/
#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */


static void mdlTerminate(SimStruct *S)
{
}


/*=============================================================*
```

```
 * See sfuntmpl_doc.c for the optional S-function methods *
 *=====================================================*/

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef  MATLAB_MEX_FILE    /*
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

.