İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

SURFACE TRACKING WITH CAD DATA AND FORCE CONTROL USING AN INDUSTRIAL ROBOTIC ARM

M.Sc. Thesis by
Okan TÜRKMEN

Department : Mechanical Engineering

Programme : System Dynamics and Control

JANUARY 2011

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

SURFACE TRACKING WITH CAD DATA AND FORCE CONTROL USING AN INDUSTRIAL ROBOTIC ARM

M.Sc. Thesis by
Okan TÜRKMEN
(503071621)

Date of submission : 20 December 2010
Date of defence examination: 28 January 2011

Supervisor (Chairman) : Assis. Prof. Dr. İlker Murat KOÇ (ITU)
Members of the Examining Committee : Prof. Dr. Ata MUĞAN (ITU)
Assoc. Prof. Dr. Salman KURTULAN (ITU)

JANUARY 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL BİR ROBOT KOLU KULLANILARAK CAD DATA TAKİBİ VE KUVVET KONTROLÜ

YÜKSEK LİSANS TEZİ
Okan TÜRKMEN
(503071621)

Tezin Enstitüye Verildiği Tarih : 20 Aralık 2010
Tezin Savunulduğu Tarih : 28 Ocak 2011

Tez Danışmanı : Yrd. Doç. Dr. İlker Murat KOÇ (ITU)
Diğer Jüri Üyeleri : Prof. Dr. Ata MUĞAN (ITU)
Doç. Dr. Salman KURTULAN (ITU)

OCAK 2011

FOREWORD

I would like to express my deep appreciation and thanks for my family, my advisor İlker Murat Koç, and my teachers Ata Muğan, Salman Kurtulan and Zeki Bayraktaroğlu. This work is supported by ITU Mechatronics Education and Research Center. Thanks to all my friends, especially Emre Akça, Ayşegül Güvenç, Çağrı Dikilitaş, Hasan Heceoğlu, and Ziya Ercan, who have helped throughout the study.

December 2010

Okan TÜRKMEN

Control Engineer

TABLE OF CONTENTS

Page

FOREWORD ..... **v**
TABLE OF CONTENTS ..... **vii**
ABBREVIATIONS ..... **xi**
LIST OF TABLES ..... **xiii**
LIST OF FIGURES ..... **xv**
SUMMARY ..... **xix**
ÖZET ..... **xxi**
1. INTRODUCTION **1**
1.1 Purpose of the Thesis 1
1.2 Robotics 2
1.2.1 Robotics background 3
2. ROBOTIC BASICS **5**
2.1 Definitions 5
2.2 Application Areas of Robotics 6
2.3 Common Robot Designs 6
2.3.1 Cartesian 6
2.3.2 Cylindrical 6
2.3.3 Spherical 7
2.3.4 Articulated 7
2.3.5 SCARA (Selective Compliance Articulated Robot Arm) 7
2.4 Technical Robotics Terms 7
2.5 Robot Sensors 7
2.6 Robotic Kinematics 8
2.6.1 Position and orientation representation 9
2.6.2 Direct kinematics 11
2.6.2.1 Denavit-Hartenberg convention 12
2.6.3 Inverse kinematics 15
2.6.4 Jacobian matrix 15
2.6.4.1 Jacobian matrix calculation 16
2.6.5 Statics 17
3. STAUBLI RX-160 INDUSTRIAL ROBOTIC ARM **19**
3.1 Technical Properties 19
3.2 Components 22
3.2.1 Controller 22
3.2.2 Manual control pendant (MCP) 23
3.2.3 ATI force-torque sensor 25
3.2.3.1 Force-Torque transducer 26
3.2.3.2 Force-Torque controller 28
3.3 RX-160 Manipulator Kinematics 28

3.3.1 Attaching frames to the joints 29
3.3.2 Calculating rotation and transformation matrices 31
3.3.3 Calculating jacobian matrix 32
3.3.4 Verifying jacobian matrix 33
3.4 Programming with VAL3 Language 36
3.4.1 VAL3 language elements 36
3.4.1.1 Applications 36
3.4.1.2 Programs 37
3.4.1.3 Data Types 37
3.4.1.4 Libraries 38
3.4.1.5 Tasks 38
3.4.2 User interface 38
3.4.3 Tasks 40
3.4.3.1 Task sequencing 40
3.4.3.2 Synchronous tasks 42
3.4.4 Arm positions 43
3.4.4.1 Arm position types 43
3.4.4.2 Arm position instructions 43
3.4.4.3 Arm configurations 45
3.4.5 Motion control 47
3.4.5.1 Motion descriptor type 47
3.4.5.2 Motion instructions 48
3.4.6 Real-Time motion control 50
3.4.6.1 Compliant motions with force control 50
3.4.6.2 Alterable motions 52
4. MOTION AND FORCE CONTROL 55
4.1 Motion Control with CAD Data 55
4.1.1 Introduction 55
4.1.2 Creating CAD files 56
4.1.2.1 Creating 2D drawings 57
4.1.2.2 Creating 3D drawings 59
4.1.3 Importing CAD files 61
4.1.4 Extracting points from CAD file to transformation data 61
4.1.5 Converting transformations to points 64
4.1.6 Moving the arm through gathered points 65
4.2 Force Control 66
4.2.1 Introduction 66
4.2.2 Stiffness control 67
4.2.3 The hybrid (position/force) control 69
4.2.4 Impedance control 72
4.2.5 Force control algorithm and discretization 73
4.2.5.1 PID controller 74
4.2.5.2 Sliding mode controller 81
4.2.5.3 PID and sliding mode controller comparison 85
4.3 Following CAD Data with Force Control 90
5. APPLICATIONS AND EXPERIMENTAL RESULTS 93
5.1 Force Control Applications and Experimental Results 93

5.1.1 Force data gathering library 93
5.1.2 Coupling effect measuring program and experimental results 95
5.1.3 Program for measuring friction forces on surfaces and experiments... 106
5.1.3.1 Friction measurements on textile fabrics 108
5.1.3.2 Friction measurements with teflon probe on elastomer surface... 111
5.1.3.3 Friction measurements with glass probe on elastomer surface.... 113
5.1.3.4 Theoretical friction calculation for comparison..... 115
5.1.4 Program for human robot collaboration..... 117
5.2 Program for 3D CAD Data Tracking and Experimental Results..... 118
5.3 Program for Tracking Surfaces Using CAD Data with Constant Forces 121
5.4 Program for Coordinate Measurement and Experimental Results..... 126
6. CONCLUSION AND RECOMMENDATIONS 133
REFERENCES..... 135
APPENDICES 137
CURRICULUM VITAE..... 139

ABBREVIATIONS

CAD : Computer Aided Design
DOF : Degrees of Freedom
DH : Denavit-Hartenberg
F/T : Force/Torque
FF : Friction Force
MCP : Manual Control Pendant
PF : Preload Force
SMC : Sliding Mode Controller

LIST OF TABLES

Page

Table 3.1 : Technical specifications of RX-160. 22
Table 3.2 : Technical specifications of F/T transducer. 28
Table 3.3 : DH parameters. 31
Table 3.4 : Error in percent for calculated Jacobian. 35
Table 3.5 : Shoulder configurations. 45
Table 3.6 : Elbow configurations. 46
Table 3.7 : Wrist configurations. 46
Table 3.8 : Motion descriptor fields. 48
Table 4.1 : DXF file section explanations. 57
Table 4.2 : DXF file point group codes. 62

LIST OF FIGURES

Page

Figure 2.1 : Frame representation with respect to reference frame. 9
Figure 2.2 : Manipulator with n+1 links. 11
Figure 2.3 : DH method for assigning frames. 13
Figure 3.1 : RX-160 elements. 19
Figure 3.2 : RX-160 dimensions. 20
Figure 3.3 : RX-160 work envelope. 21
Figure 3.4 : Controller of the industrial arm. 23
Figure 3.5 : Manual control pendant. 24
Figure 3.6 : F/T transducer on wrist of RX-160. 25
Figure 3.7 : F/T sensor system components. 26
Figure 3.8 : F/T transducer. 27
Figure 3.9 : Axes of F/T transducer. 27
Figure 3.10 : RX-160 frame assignment. 30
Figure 3.11 : User interface page dimensions. 39
Figure 3.12 : Task execution times. 41
Figure 3.13 : Shoulder configurations. 46
Figure 3.14 : Elbow configurations. 47
Figure 3.15 : Wrist configurations. 47
Figure 4.1 : 2D triangle drawing. 58
Figure 4.2 : 3D helix drawing. 59
Figure 4.3 : 3D helix drawing with points. 60
Figure 4.4 : Mass spring system. 67
Figure 4.5 : Block diagram of mass spring control system. 68
Figure 4.6 : Simplified manipulator with 3-DOF. 70
Figure 4.7 : Simplified block diagram of hybrid control. 70
Figure 4.8 : Block diagram of hybrid control. 71
Figure 4.9 : Block diagram of impedance control. 72
Figure 4.10 : Relationship between force and displacement. 74
Figure 4.11 : Block diagram of PID force control algorithm. 75
Figure 4.12 : Ziegler Nichols method implementation diagram 78
Figure 4.13 : Ziegler Nichols experiment 1 78
Figure 4.14 : Ziegler Nichols experiment 2 79
Figure 4.15 : Ziegler Nichols experiment 3 79
Figure 4.16 : PID response for Ziegler Nichols coefficients 80
Figure 4.17 : Response for fine tuned PID parameters. 81
Figure 4.18 : Block diagram of SMC force control algorithm. 81
Figure 4.19 : Tune results for $ksmc = 0{,}01$. 82
Figure 4.20 : Tune results for $ksmc = 0{,}02$. 83
Figure 4.21 : Tune results for $ksmc = 0{,}05$. 83

Figure 4.22 : Tune results for $ksmc = 0{,}1$. 84
Figure 4.23 : Tune results for $ksmc = 0{,}2$. 84
Figure 4.24 : Comparison results for 10N preload. 85
Figure 4.25 : Friction forces for 10N preload. 86
Figure 4.26 : Comparison results for 12N preload. 86
Figure 4.27 : Friction forces for 12N preload. 87
Figure 4.28 : Comparison results for 14N preload. 87
Figure 4.29 : Friction forces for 14N preload. 88
Figure 4.30 : Comparison results for 16N preload. 88
Figure 4.31 : Friction forces for 16N preload. 89
Figure 4.32 : Comparison results for 18N preload. 89
Figure 4.33 : Friction forces for 18N preload. 90
Figure 4.34 : Block diagram of force control and position tracking algorithm. 91
Figure 5.1 : PN graph of force data gathering library. 94
Figure 5.2 : Polyamide probe. 96
Figure 5.3 : Metal dome probe. 96
Figure 5.4 : Pointer probe. 97
Figure 5.5 : Flexible probe. 97
Figure 5.6 : Teflon probe 98
Figure 5.7 : PN graph of coupling effect measurement application. 99
Figure 5.8 : Coupling measurement for polyamide probe. 100
Figure 5.9 : Coupling measurement for metal dome probe. 101
Figure 5.10 : Coupling measurement for pointer probe. 102
Figure 5.11 : Coupling measurement for flexible probe. 103
Figure 5.12 : Coupling measurement for teflon probe. 104
Figure 5.13 : Force and position of end effector on Z axis for flexible probe. 105
Figure 5.14 : Change in force with respect to position for flexible probe. 105
Figure 5.15 : Change in force with respect to position for teflon probe. 106
Figure 5.16 : PN graph of friction measurement application. 107
Figure 5.17 : Preloads of friction measurements on satin fabric. 108
Figure 5.18 : Friction forces for satin fabric. 108
Figure 5.19 : Preloads of friction measurements on plain fabric. 109
Figure 5.20 : Friction forces for plain fabric. 109
Figure 5.21 : Preloads of friction measurements on twill fabric. 110
Figure 5.22 : Friction forces for twill fabric. 110
Figure 5.23 : Preloads for elastomer and teflon. 111
Figure 5.24 : Friction forces for elastomer and teflon. 112
Figure 5.25 : Friction coefficients with respect to preloads. 112
Figure 5.26 : Friction coefficients with respect to speed values. 113
Figure 5.27 : Preloads for elastomer and glass friction. 114
Figure 5.28 : Friction forces for elastomer and glass. 114
Figure 5.29 : Friction coefficient with respect to preload. 115
Figure 5.30 : Comparison of experimental and theoretical values 116
Figure 5.31 : PN Graph of human robot collaboration application 118
Figure 5.32 : Simulation of 3D helix path tracking 1. 119
Figure 5.33 : Simulation of 3D helix path tracking 2. 120
Figure 5.34 : 2D circle drawing. 121

Figure 5.35 : Simulation results for circle. 122
Figure 5.36 : Force measurements for circle..... 122
Figure 5.37 : 2D triangle drawing. 123
Figure 5.38 : Simulation results for triangle. 123
Figure 5.39 : Force measurements for triangle. 124
Figure 5.40 : 2D drawing of curl. 124
Figure 5.41 : Simulation results for curl. 125
Figure 5.42 : Force measurements for curl. 125
Figure 5.43 : Surface tracking pattern..... 126
Figure 5.44 : Circular shape. 127
Figure 5.45 : Scan results for circular shape. 127
Figure 5.46 : Triangular shape. 128
Figure 5.47 : Scan results for triangular shape..... 128
Figure 5.48 : Pentagonal shape. 129
Figure 5.49 : Scanning results for pentagonal shape. 129
Figure 5.50 : Square shape. 130
Figure 5.51 : Scanning results for square shape..... 131
Figure 5.52 : Scanning results of İTÜ. 132
Figure 5.53 : Scanning results of MEAM. 132

SURFACE TRACKING WITH CAD DATA AND FORCE CONTROL USING AN INDUSTRIAL ROBOTIC ARM

SUMMARY

The application areas for robotics are limitless. Today, robots are used for almost every work where humans require help. In all these tasks, interaction with environment is an important issue, as without this interaction application areas would be limited. Additionally, a flexible and easy user-programming interface is essential, as every user should be able to attain robot programming.

One of the main purposes of this study is to achieve force interaction of robot with its environment. The forces that arise on robotic arm's end-effector are measured using a force/torque sensor. In order to achieve measurements with good quality, several probes are investigated and suitable ones are chosen for desired applications. The force control algorithms that can be used are investigated and suitable control algorithms are chosen for applications. Stiffness contact model is used for modeling contact between end effector of robot and the environment. Force control is achieved using two different algorithms; PID and sliding mode controller. These two algorithms are compared with each other using experimental results.

Besides, conventional robot programming in industry, using the robot manual teach control pendant, is a laboring and time consuming task which also requires a programmer with recent experience. Another purpose of this study is to create an interface that even a programmer, who has no experience on robotics, can interact with robot very easily and fast while the program created handles the process on a perfect way. For this purpose, path following with reference to CAD data is investigated.

Additionally, tracking the path defined in CAD files and force control applications are combined together. After a satisfying background on both subjects is provided, force control and CAD tracking is achieved at the same time on different axes.

Lastly, the industrial robotic arm STAUBLI RX-160 that has six degrees of freedom is used for applications. The force measurements are obtained using the ATI Delta force/torque measurement system. The transducer of this system is mounted on wrist of the robotic arm. Several applications are experimented using this setup including human machine collaboration, friction measurements, and surface scanning which yield promising results.

ENDÜSTRİYEL BİR ROBOT KOLU KULLANILARAK CAD DATA TAKİBİ VE KUVVET KONTROLÜ

ÖZET

Günümüzde robotlar insanların ihtiyaç duyduğu sınırsız sayıda uygulamada kullanılmaktadır. Tüm bu uygulamalarda, robotların çevre ile etkileşimi uygulama alanlarının genişliğinin ve esnekliğinin sağlanması açısından önemli bir yer teşkil etmektedir. Bunun yanında, roborları her kullanıcının kolay bir şekilde programlaması amacı ile esnek ve kolay bir kullanıcı programlama arayüzü temel bir ihtiyaçtır.

Bu çalışmanın temel amaçlarından biri robotların çevreleri ile kuvvet etkileşimlerinin sağlanmasıdır. Bu hedefin gerçekleştirilmesi amacı ile çeşitli kuvvet kontrol algoritmaları incelenmiş, bu algoritmalardan gerçekleştirilecek uygulamalara uygun olanlar seçilmiştir. Robotun uç elemanında oluşan kuvvetler robota bağlı bir kuvvet tork sensörü yardımı ile ölçülmektedir. Bu ölçümlerin en iyi şekilde alınması amacı ile robot uç elemanına bağlanan çeşitli tipler üzerinde denemeler yapılmış, uygulamarda kullanılabilecek en uygun tipler belirlenmiştir. Uygulamarda robot uç elemanı ve çevre arasındaki etkileşimin modellenmesi amacı ile yay modeli kullanılmıştır. Kuvvet kontrolü, PID ve sliding mode kontrolörleri yardımı ile sağlanmıştır ve bu iki farklı algoritma uygulamar yardımı ile karşılaştırılmıştır.

Endüstride kullanılmakta olan robotlar genelde manüel kontrol pendantları ile çalışma noktaları tanımlanarak programlanmaktadır. Bu çok zaman tüketen, zahmetli bir iştir ve bu işi sadece robot hakkında yeterli bilgi sahibi kullanıcılar gerçekleştirebilir. Bu çalışmanın temel amaçlarından biri de robot hakkında temel bilgi sahibi bir kullanıcının bile rahat bir şekilde programlama yapabilmesini sağlayacak bir programlama arayüzü sağlanmasıdır. Bu ihtiyacın karşılanması amacı ile CAD verisi kullanılarak rota takibi konusundan yararlanılmıştır.

Bunlara ek olarak, CAD verisi kullanılarak rota takibi ve kuvvet kontrolü uygulamaları birleştirilmiş ve kuvvet kontrolü ile CAD takibinin farklı eksenlerde eş zamanlı uygulanabilmesi sağlanmıştır.

Son olarak, uygulamalarda STAUBLI RX-160 altı eksenli endüstriyel robot kolu kullanılmıştır. Kuvvet ölçümleri robot kolunun bileğine bağlanmış olan ATI kuvvet tork sensörü yardımı ile yapılmıştır. Bu düzenek kullanılarak insan robot işbirliği, sürtünme ölçümü ve yüzey taraması gibi uygulamalar gerçekleştirilmiştir.

1. INTRODUCTION

Today, robots are making a great effect on modern life. They are used in almost all application areas including transportation, industrial manufacturing, healthcare, space exploration and many more. From the beginning of time, it has been a dream of the humanity to create skilled and intelligent machines. And today, this dream started to become one of the most important realities in our world [1]. In the last 25 years, robotics has become a great centre of interest for scholars and therefore the literature became richer, in terms of textbooks, journals, and monographs. Because of the fact that robotics is an interdisciplinary subject having roots in different scientific areas including cybernetics, mechanics, controls, computers, bioengineering, and electronics, it can get inspired from a wide range of scientific developments [2]. And additionally, it can inspire a wide range of scientific areas.

1.1 Purpose of the Thesis

There are various application areas for robotics, from the easiest tasks in daily life, to the hardest ones in industrial manufacturing. In all these tasks, interaction with the environment is essential and has an important role. One of the main purposes of this study is to achieve force interaction of robot with its environment. This aim is being achieved with a force-torque sensor attached to the robot. On the first step, gathering measurements from the sensor will be achieved. Next, gathered data will be processed and interpreted. And lastly, based on the information gathered robot will be commanded.

Additionally, achieving an easy programming interface is a very important issue in robotics. As the application tasks get harder and more complicated, it also gets harder to model, program, and simulate programs for robots. Another purpose of this study is to inspect applicability and utility of CAD offline programming in robotics. For this reason, the general usage of CAD files will be studied and examples of CAD drawings will be inspected.

Besides, there are many processes which require precise and complex surface tracking in robotics, like deburring, polishing, and quality control. One of the fastest and precise ways to achieve these applications is using CAD data. Another aim of this study is to learn how to use CAD data for robotic programming and use this data for surface tracking. First, how to create CAD files will be inspected. Then, how to transfer these files to robotic controller will be inspected. Next, how to read created files will be inspected. And at last, robotic tracking problem with CAD data will be inspected.

With the use of our knowledge gathered from the studies mentioned above, case study examples like pushing with constant forces, friction measurement, human machine collaboration and CAD data tracking applications will be inspected.

1.2 Robotics

Robot is a term used for both virtual and mechanical artificial agents that are used in order to replace human beings in the execution of tasks [2, 3]. These tasks can be either physical activities or mental activities like decision-making. Generally, robots are electro-mechanical machines, which are directed by computer or electronic programs, being able to do desired tasks on their own [3].

Robotics is the engineering science, technology and design, briefly all the knowledge that is used in order to build robots. Therefore, it is an interdisciplinary subject including mechanic, electronic, control and computer disciplines [2, 4].

At present, there are many application areas for robots. Considering production purposes and application areas robots can be classified into two types, general-purpose autonomous robots and dedicated robots [3]. Humanoid robots, which mimic human beings and resemble them in appearance, are an example for general-purpose autonomous robots. On the other hand, industrial robots are an example for dedicated robots.

1.2.1 Robotics background

The word “robot” was firstly introduced by a Czech writer Karel Capek in his play named Rossum’s Universal Robots, published in 1920 [2]. It was inspired from the term “robota” which means worker in Slav languages [2]. In the 1940’s a well-known writer Isaac Asimov came up with the idea of robots having human appearance but no feelings and described robots as mechanical artifacts [2]. Getting inspirations from science fiction people started to pay more attention to the potential of robotics and developments gained speed those years, leading the production of first commercial robot in the year of 1956 and then the first industrial robot in 1961 [3]. Factory floor was like the playground of robotics’ childhood. Industrial robots were produced in order to achieve applications like automatic spraying, spot welding, grinding, material handling, and parts assembly. In the late 1970s, the world “mechatronics” were firstly used by Japanese, this word defined machinery, in which electronics was combined with mechanical systems [1]. During the 1980s, getting inspired from many study cases, from rovers for extreme environments, to service robots, a broad research domain occurred for robotics. Today this domain gets larger every moment in an inevitable way as researchers are ambitious to explore new horizons. Kinematics, dynamics, and control system theory were refined and applied to real complex robot mechanisms [1]. In order to achieve various tasks, robots were made environment conscious. With the use of systems like vision, tactile and force sensing, and voice recognition, robotics became a pioneer science branch for studying connection of sensing to actuation [1].

In robotics, force control is one of the most important issues. There are several researches on this subject from medical to industrial applications. Force control is used for surface cleaning tasks. Hybrid control of a 7-DOF redundant manipulator [5] and hybrid control of a mobile manipulator [6] can be given as examples on this subject. In these applications manipulators are equipped with cleaning tools on their end-effectors and the aim is to track the surface with desired forces in order to achieve cleaning task. Besides, force control is used for peg-in-hole applications. Impedance control of an industrial robot using active robotic auxiliary device [7] and fuzzy adaptation impedance control of a 6-DOF robot [8] can be given as examples on this subject. Human-Robot collaboration is one of the most used force control applications. Robust control of force-coupled human-robot interaction in assembly processes [9] and robot assisted microsurgical manipulation [10] are examples for human-robot collaboration tasks. There are several medical tasks which require force control. A cascade controlled orthopedic surgery robot [11] and hybrid impedance controlled catheter insertion [12] can be given as examples on this subject. Additionally, there are several force control tasks which involve polishing [13] and grinding [14] applications.

CAD based robot programming is another important issue in robotics. Both two dimensional and three dimensional designs are used in tasks depending on needs. Processing metal profiles using 2D CAD based programming [15], mold polishing using CAD based position force controller [16], and CAD based programming for robotic deburring cell [17] are decent examples for CAD based robot programming. In these applications the CAD data is used in order to achieve robot programming for desired manipulation tasks.

2. ROBOTIC BASICS

2.1 Definitions

In this section most common definitions in robotics will be emphasized [2]:

- A robot is an electromagnetic device with multiple degrees of freedom (dof) that is programmable to accomplish a variety of tasks.
- Robotics is the science of robots. Humans working in the robotics area are called roboticists.
- Degree of freedom is the number of independent motions a device can make. It is also called mobility.
- A manipulator is an electromechanical device capable of interacting with its environment.
- The term anthropomorphic is used to describe the objects designed or appearing like human beings.
- An end-effector is the tool, gripper, or other device mounted at the end of a manipulator, in order to accomplish useful tasks.
- Workspace is the volume in space which a robot's end effector can reach, both in position and orientation.
- Position is the translational location of an object.
- Orientation is the rotational location of an object. For example, orientation of an airplane is measured by roll, pitch, and yaw angles.
- Pose is the term used to describe orientation and position together.
- A link is a rigid piece of material used to connect joints of a robot.
- Joint is the device which is used to allow relative motion between two links of a robot.
- Kinematics is the study of motion with no regard to forces and torques.
- Dynamics is the study of motion with regard to forces and torques.

- An actuator is the device used to provide force and torque in a robot for motion.
- A sensor is the device used to read actual variables in robot motion for use in control.
- Haptic is a word from Greek language, which means “to touch”. Haptic interfaces are used in order to give human operators the sense of touch, either in virtual or real, remote environments with the devices used.

2.2 Application Areas of Robotics

The robots are used almost everywhere and in every application today. Basically, if there is a work to be done which is too dangerous, dull, or difficult to perform, or needs to be finished fast, then the robots are applied there [2]. There are three main application areas for robotics [2]:

- In production industry, robots are used in manufacturing, assembly, welding, spray painting, deburring, polishing, and machining.
- In remote operations, robots are used in undersea, nuclear environment, bomb disposal, law enforcement, and outer space.
- In service industry, robots are used as hospital helpmates, handicapped assistants, retails, household servants, and lawnmowers.

2.3 Common Robot Designs

2.3.1 Cartesian

Cartesian robots have three linear axes of motion possible and they have only prismatic joints. They are mostly used for pick and place tasks and to move heavy loads [2].

2.3.2 Cylindrical

The position of cylindrical robots are controlled a height, an angle, and a radius. They have two prismatic joints and one revolute joint. These robots are commonly used for assembly tasks [2].

2.3.3 Spherical

Spherical robots have two rotational joints and one prismatic joint.

2.3.4 Articulated

These robots are anthropomorphic and controlled by three revolute joints. They are the most versatile robots, but also the ones that are hardest to program [2].

2.3.5 SCARA (Selective Compliance Articulated Robot Arm)

These robots consist of three revolute joints and one prismatic joint. Providing the benefit of articulated and cylindrical robots, SCARA robots are a blend of them [2].

2.4 Technical Robotics Terms

In this section commonly used technical terms in robotics are explained [2]:

- Load bearing capacity is the maximum weight carrying capacity of the robot.
- Accuracy is the ability of going to the specified position without making a mistake. As it is impossible to position a machine exactly, accuracy in robotics is defined as the ability of a robot to position itself to the desired location with the minimal error.
- Repeatability is the ability to achieve the same positioning task multiple times with same results. Note that an accurate robot may not be repeatable and a repeatable robot may not be accurate for sure.
- Work envelope is the maximum reach or volume in which a robot can operate. Work envelope is usually specified by the combination of the limits of each of the robots' parts.
- Workcells are the group of machines, equipments, or robots used together with coordination in order to achieve useful works.

2.5 Robot Sensors

Robots need small electronic or electro-mechanical components that will allow them to react with their environment [2]. Only with the information gathered from these

sensors, robots can be controlled properly if interaction with environment is needed. Some common sensors are described below [2]:

- Vision data can be used with a computer controlled camera. It will allow robot to see and be aware of its environment, and act accordingly.
- Robots can be controlled with voice commands with voice systems. When robots need to be trained when trainer has to manipulate other object this system is useful.
- Force and torque sensors not only provide the robot to sense of force being applied on the arm but also the direction of the force. These sensors are one of the essential elements of haptic devices. These sensors are also used for tactile sensing.
- Proximity sensors are used in order to allow robots to detect the presence of objects that are very close to the arm, before the contact with the object is actually achieved. These sensors generally used for collision avoidance.
- Limit switches are used in "end-of-motion" areas in workspaces. They are generally used to inform robot in order to avoid collisions, change direction, or stop working.

2.6 Robotic Kinematics

Robotic mechanisms are systems of rigid bodies which are connected together with joints, unless exceptions about the system are stated. The term pose is used to collectively define the orientation and position of an object. Therefore, kinematics can be defined as the description of pose, velocity, acceleration, and all higher order derivatives of the pose of the bodies that compose the mechanism [1]. Since kinematics doesn't deal with forces and torques which induce the motion, this section will only focus on description of pose and velocity. Besides, instead of giving detailed information about the kinematics, only essentials used in thesis are emphasized on this section.

2.6.1 Position and orientation representation

As shown in Figure 2.1, the let reference frame be defined as O-x y z, where x, y, and z are the unit vectors of the frame axes. The position of the point O' on the rigid body with respect to the reference frame can be expressed by the relation [2]:

$$o' = o'_x x + o'_y y + o'_z z \quad \textbf{(2.1)}$$

And the position of O' can be written in compact form as a (3x1) vector [2]:

$$o' = \begin{bmatrix} o'_x \\ o'_y \\ o'_z \end{bmatrix} \quad \textbf{(2.2)}$$

Let x', y', and z' be the unit vectors of the frame O-x' y' z', then these unit vectors can be expressed by the relation below with respect to reference frame [2]:

$$x' = x'_x\, x + x'_y\, y + x'_z\, z \quad \textbf{(2.3)}$$

$$y' = y'_x\, x + y'_y\, y + y'_z\, z \quad \textbf{(2.4)}$$

$$z' = z'_x\, x + z'_y\, y + z'_z\, z \quad \textbf{(2.5)}$$

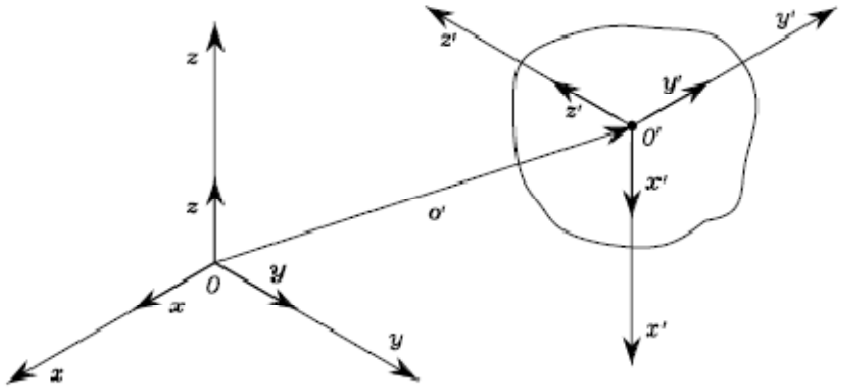

Figure 2.1 : Frame representation with respect to reference frame.

In order to represent the orientation of the body with respect to reference frame, the rotation matrix will be used [2]:

$$R = [x' \quad y' \quad z'] = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} = \begin{bmatrix} x'^T x & y'^T x & z'^T x \\ x'^T y & y'^T y & z'^T y \\ x'^T z & y'^T z & z'^T z \end{bmatrix} \quad \textbf{(2.6)}$$

A rotation matrix can also be used as the matrix operator allowing rotation of a vector by a given angle about an arbitrary axis in space [2].

To sum up, a rotation matrix can be used for three main reasons [2]:

- It can be used in order to describe the mutual orientation between two coordinate frames.
- It represents the coordinate transformation between the coordinates of a point expressed in two different frames.
- It is an operator for vectors which allows them to rotate in the same coordinate frame.

Inverse of the rotation matrix is equal to the transpose of it, that is:

$$R_i^j = (R_j^i)^T = (R_j^i)^{-1} \quad \textbf{(2.7)}$$

The rotation matrices can be composed with each other, that is:

$$R_3^0 = R_1^0 \, R_2^1 \, R_3^2 \quad \textbf{(2.8)}$$

As it has been defined how to represent orientation and translation of a body with respect to a reference frame, now it is time to define how to achieve both of these together:

$$p^0 = o_1^0 + R_1^0 p^1 \quad \textbf{(2.9)}$$

The Equation 2.9 represents the coordinate transformation (which includes orientation and translation) of a bound vector between two frames. In this equation "p" represents a (3x1) vector, which defines a point in space.

In order to achieve a compact representation of the relationships between the coordinates of the same point in two different frames, homogenous representation of a generic vector "p" can be introduced as the vector " $\tilde{p}$ ", which is obtained by adding another element to it [2]:

$$\tilde{p} = \begin{bmatrix} p \\ 1 \end{bmatrix} \quad \textbf{(2.10)}$$

By adopting this representation to the Equation 2.9, coordinate transformation can be written in terms of a (4x4) matrix [2]:

$$A_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix} \quad \textbf{(2.11)}$$

The matrix on Equation 2.11 is called the homogenous transformation matrix. Now using the Equation 2.9, it is easy to write the equation below:

$$\tilde{p}^0 = A_1^0 \, \tilde{p}^1 \quad \textbf{(2.12)}$$

Note that inverse of homogenous transformation matrix is equal to its transpose:

$$(A_j^i)^T = (A_j^i)^{-1} \quad \textbf{(2.13)}$$

Besides, the homogenous transformation matrix can be composed like rotation matrix:

$$A_3^0 = A_1^0 \, A_2^1 \, A_3^2 \quad \textbf{(2.14)}$$

2.6.2 Direct kinematics

The purpose of the direct kinematics is to compute the pose of the end-effector as a function of the joint variables [2].

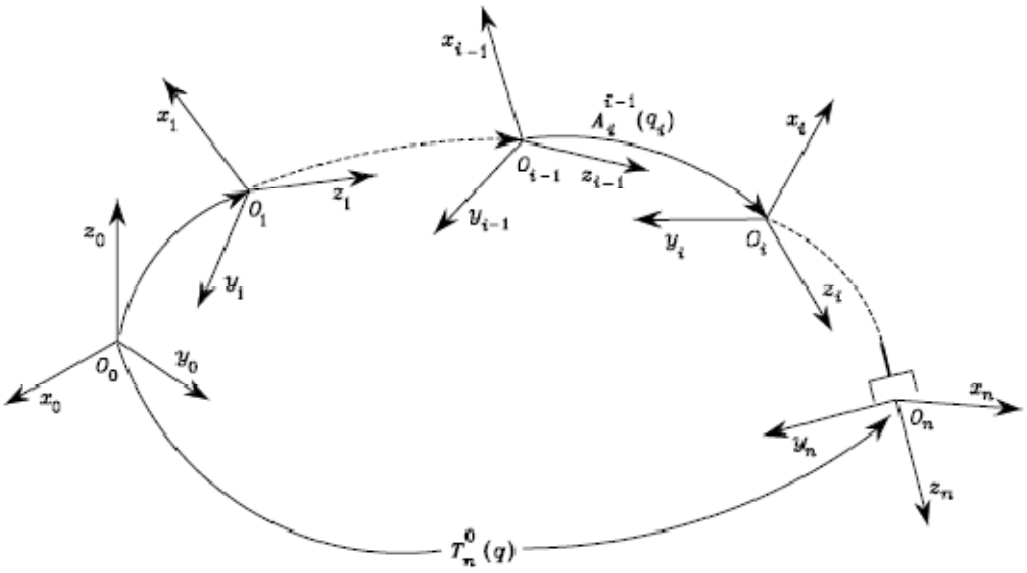

Figure 2.2 : Manipulator with n+1 links.

Assume that the manipulator on Figure 2.2 is being studied, and a coordinate frame for each link from "0" to "n" is defined. The coordinate transformation describing the position and orientation of Frame "n" with respect to Frame "0" (reference frame) is given by:

$$T_n^0(q) = A_1^0(q_1)\, A_2^1(q_2) \ldots A_n^{n-1}(q_n) \quad \textbf{(2.15)}$$

On Equation 2.15, the matrices which are named "A" are homogenous transformation matrices. And the transformation matrices are functions of "q", which are joint variables.

Assuming that the transformation matrix from base to link "0" is T_0^b, and the transformation matrix from link "n" to end-effector is T_e^n, the transformation matrix from base to end-effector can be obtained as:

$$T_e^b(q) = T_o^b\, T_n^0(q)\, T_e^n \quad \textbf{(2.16)}$$

2.6.2.1 Denavit-Hartenberg convention

Using a general method is within the reason in order to calculate the kinematic expression on Equation 2.15. The problem is to determine two frames attached to two links and calculate the coordinate transformations between them, in order to define the relative position and orientation of two consecutive links [2].

Denavit-Hartenberg convention (DH) is one of the most convenient methods in order to achieve this task. With reference to Figure 2.3, let us assume that link frame i is being defined.

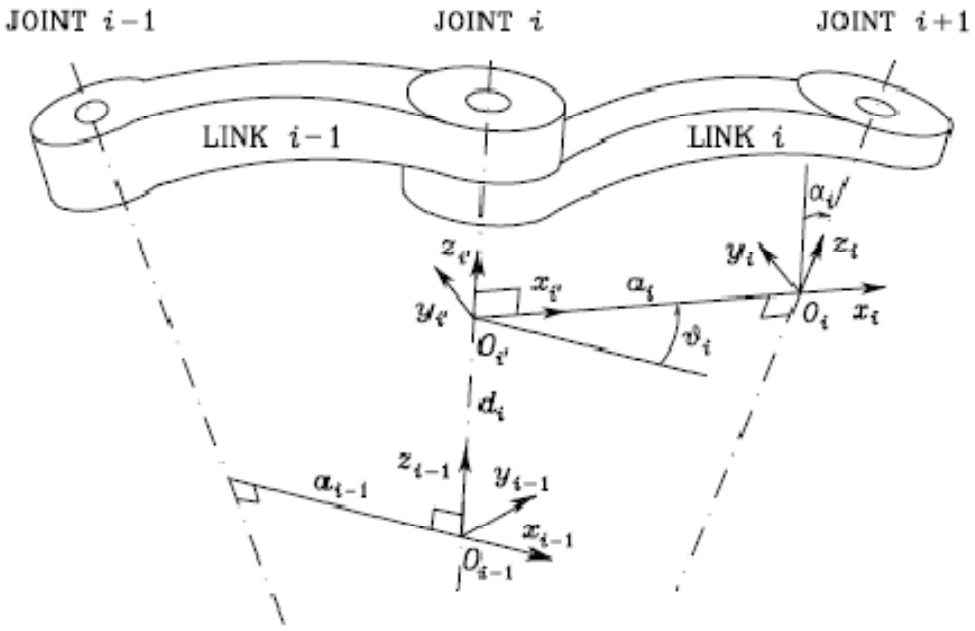

Figure 2.3 : DH method for assigning frames.

The steps, in order to Denavit-Hartenberg convention, are explained below [2]:

- Axis z_i must be chosen along the axis of Joint $i + 1$.
- The origin O_i must be located at the intersection of axis z_i with the common normal to axes z_i and z_{i-1}. Additionally, $O_{i'}$ must be located at the intersection of the common normal with axis z_{i-1}.
- Axis x_i must be chosen along the common normal to axes z_i and z_{i-1}, with the direction from Joint i to Joint $i + 1$.
- At last, axis y_i is chosen with the help of right hand rule, with the help of z_i and x_i .

After the link frames are defined, the position and orientation of frame i with respect to frame $i - 1$ can be completely specified using the parameters below [2]:

- a_i : The distance between O_i and $O_{i'}$.
- d_i : Coordinate of O_i along z_{i-1}.
- α_i : Angle between axes z_i and z_{i-1} about axis x_i to be taken positive when rotation is made counter-clockwise.

- ϑ_i: Angle between axes x_i and x_{i-1} about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

On next step, it is possible to express coordinate transformation between frame i and frame $i-1$ using the steps below [2]:

- A frame aligned with frame $i-1$ must be chosen.
- The chosen frame must be translated by d_i along axis z_{i-1} and rotated by ϑ_i about axis z_{i-1}. This sequence can be described by homogenous transformation matrix below:

$$A_{i\prime}^{i-1} = \begin{bmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) & 0 & 0 \\ \sin(\vartheta_i) & \cos(\vartheta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.17)}$$

- The frame aligned with frame i' must be translated by a_i along axis $x_{i\prime}$ and rotated by α_i about axis $x_{i\prime}$. This sequence can be described by homogenous transformation matrix on Equation 2.18:

$$A_i^{i\prime} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.18)}$$

- The composition of two matrices obtained above will be the transformation matrix given below:

$$A_i^{i-1}(q_i) = A_{i\prime}^{i-1}\, A_i^{i\prime} =$$

$$\begin{bmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i)\cos(\alpha_i) & \sin(\vartheta_i)\sin(\alpha_i) & a_i\cos(\vartheta_i) \\ \sin(\vartheta_i) & \cos(\vartheta_i)\cos(\alpha_i) & -\cos(\vartheta_i)\sin(\alpha_i) & a_i\sin(\vartheta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.19)}$$

Note that the transformation matrix from frame i to frame $i-1$ is a function of only the joint variables q_i, which is, ϑ_i for a revolute joint or d_i for a prismatic joint [2].

Direct kinematic equation of a manipulator can be obtained by applying the method above for finding transformation matrices between consecutive frames and then using these matrices with Equations 2.15 and 2.16. At the end of the procedure, the transformation matrix from base to end-effector, which is $T_e^b(q)$, is obtained.

2.6.3 Inverse kinematics

In contrast to direct kinematics problem, inverse kinematics problem consists of the determination of the joint variables corresponding to a given end-effector position and orientation. In order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions which allow the execution of the desired motion, the solution to this problem is fundamental [2].

2.6.4 Jacobian matrix

Assume a manipulator with "n" degrees of freedom is being studied. The direct kinematics can be expressed on the form given on Equation 2.20 [2].

$$T_e(q) = \begin{bmatrix} R_e(q) & p_e(q) \\ 0^T & 1 \end{bmatrix} \quad \textbf{(2.20)}$$

On this equation "q" represents the joint variables with the form given on Equation 2.21.

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \quad \textbf{(2.21)}$$

Differential kinematics' aim is to find the relationship between the joint velocities and end-effector linear and angular velocities. In other words, its aim is to calculate the linear velocity $\dot{p}_e$and angular velocity ω_e as a function of joint velocities $\dot{q}$.

Jacobian matrices are the operators that will give us the following relationships on Equation 2.22 and Equation 2.23 [2].

$$\dot{p}_e = J_p(q)\dot{q} \quad \textbf{(2.22)}$$

$$\omega_e = J_o(q)\dot{q} \quad \textbf{(2.23)}$$

In a more compact form Jacobian matrices on Equation 2.22 and Equation 2.23 can be written as on the Equation 2.24.

$$v_e = \begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = J(q)\ \dot{q} \quad \textbf{(2.24)}$$

On Equation 2.24, Jacobian matrix "J" with (6xn) dimension is called the geometric Jacobian.

$$J(q) = \begin{bmatrix} J_p(q) \\ J_o(q) \end{bmatrix} \quad \textbf{(2.25)}$$

2.6.4.1 Jacobian matrix calculation

In order to calculate Jacobian matrix, proceeding separately for linear and angular velocities is an eligible way [2].

At the first step, linear velocities will be inspected:

- If joint i is prismatic then the Jacobian can be expressed with Equation 2.26.

$$J_{p_i} = z_{i-1} \quad \textbf{(2.26)}$$

- If joint i is revolute then the Jacobian can be expressed with Equation 2.27.

$$J_{p_i} = z_{i-1} \times (p_e - p_{i-1}) \quad \textbf{(2.27)}$$

Next, angular velocities will be inspected:

- If joint i is prismatic then the Jacobian can be expressed with Equation 2.28.

$$J_{O_i} = 0 \quad \textbf{(2.28)}$$

- If joint i is revolute then the Jacobian can be expressed with Equation 2.29.

$$J_{O_i} = z_{i-1} \quad \textbf{(2.29)}$$

To sum up, the Jacobian which is defined on Equation 2.25 can be expressed in a partitioned form which is given on Equation 2.30.

$$J = \begin{bmatrix} J_{p_1} & \cdots & J_{p_n} \\ J_{O_1} & \cdots & J_n \end{bmatrix} \quad \textbf{(2.30)}$$

The partitioned parts of this Jacobian can be found by the expression given on Equation 2.31 [2].

$$\begin{bmatrix} J_{p_i} \\ J_{o_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & for\ a\ prismatic\ joint \\ \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} & for\ a\ revolute\ joint \end{cases} \quad \textbf{(2.31)}$$

The term z_{i-1} is calculated by the Equation 2.32.

$$z_{i-1} = R_1^0(q_1)\, R_2^1(q_2) \ldots R_{i-1}^{i-2}(q_{i-1})\, z_0 \quad \textbf{(2.32)}$$

$$z_0 = [0 \quad 0 \quad 1]^T \quad \textbf{(2.33)}$$

In the Equation 2.32, the matrices expressed with R are the rotational matrices. And the term z_0 is used to select the third column.

The terms p_e and p_{i-1} are calculated by the Equations 2.34 and 2.35.

$$p_e = A_1^0(q_1)\, A_2^1(q_2) \ldots A_n^{n-1}(q_n)\, p_0 \quad \textbf{(2.34)}$$

$$p_{i-1} = A_1^0(q_1)\, A_2^1(q_2) \ldots A_{i-1}^{i-2}(q_{i-1})\, p_{i-1} \quad \textbf{(2.35)}$$

$$p_0 = [0 \quad 0 \quad 0 \quad 1]^T \quad \textbf{(2.36)}$$

In the Equations 2.34 and 2.35, the matrices expressed with A are the transformation matrices. And the term p_0 is used to select the fourth column.

2.6.5 Statics

The aim of statics is to determine the relationship between the generalized forces applied to the end-effector and the forces applied to the joints [2].

Let τ, with a dimension of (nx1), be a vector indicating the torques on joints. And let γ, with a dimension of (6x1), be a vector indicating the forces on the end-effector being used, where n is the number of joints.

The Jacobian, which is mentioned on previous chapters, will help us setup the relationship desired for statics with Equation 2.37 [2].

$$\tau = J^T(q)\, \gamma_e \quad \textbf{(2.37)}$$

3. STAUBLI RX-160 INDUSTRIAL ROBOTIC ARM

On our applications STAUBLI RX-160, an industrial arm with six joints is used. In this section of the thesis, this arm will be described in order to reveal how applications are prepared and used.

Firstly, the technical properties and components of the arm including controller, manual control panel, industrial camera, and force-torque sensor will be introduced. On next step, working modes will be explained.

3.1 Technical Properties

As shown on Figure 3.1, RX-160 is composed of members interconnected by joints. An axis around which two members pivot is comprised by each joint [18].

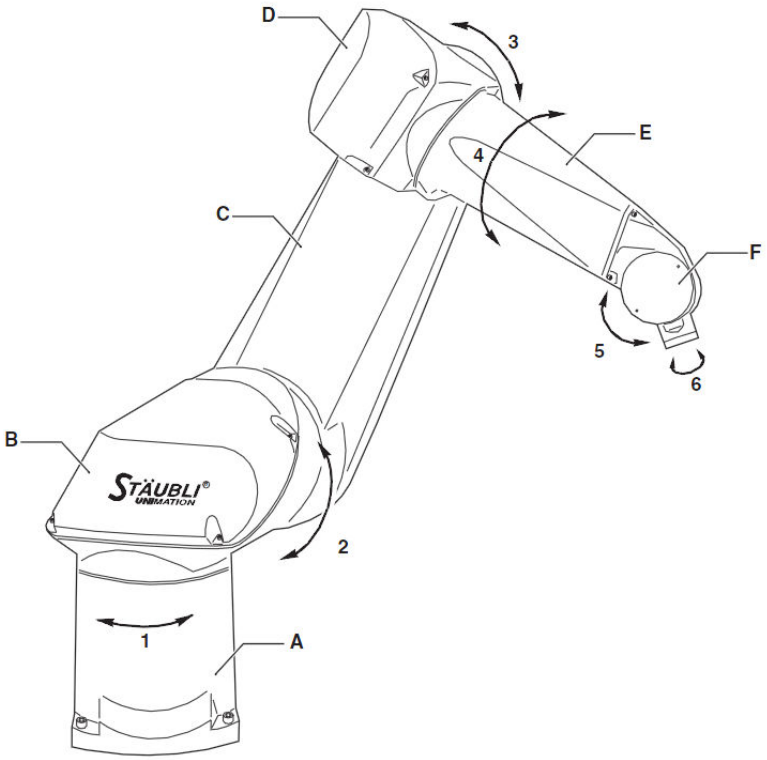

Figure 3.1 : RX-160 elements.

The motions of the robot's joints are generated by brushless motors coupled to resolvers. Each of these motors is also equipped with a parking brake. The arm is assembled in a flexible way which allows a wide range of applications can be performed [18].

With reference to Figure 3.1 the arm's main members are: the base (shown with "A"), the shoulder (shown with "B"), the arm (shown with "C"), the elbow (shown with "D"), the forearm (shown with "E"), and the wrist (shown with "F") [18].

The dimensions of the arm are shown on Figure 3.2 [18].

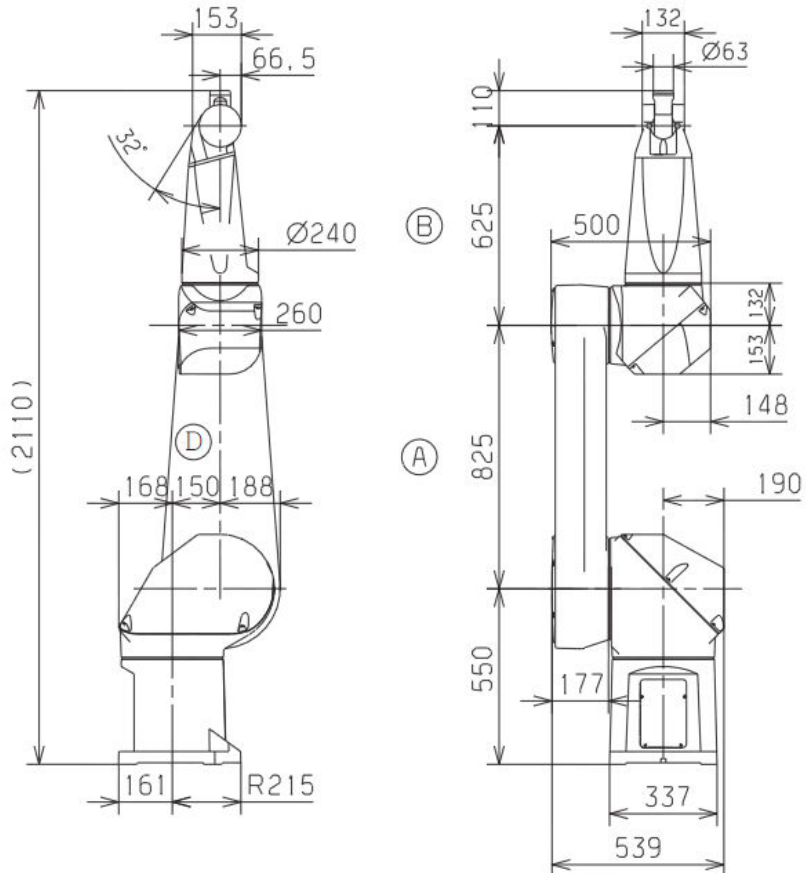

Figure 3.2 : RX-160 dimensions.

The arm can work in environments with temperatures from $+5^{o}C$ to $+40^{o}C$. The humidity of the working environment can be from 30% to 95%. The maximum working altitude of the robot is 2000 meters [18].

The weight of the robot arm is 248 kilograms [18].

The work envelope of the robot is shown on Figure 3.3 [18].

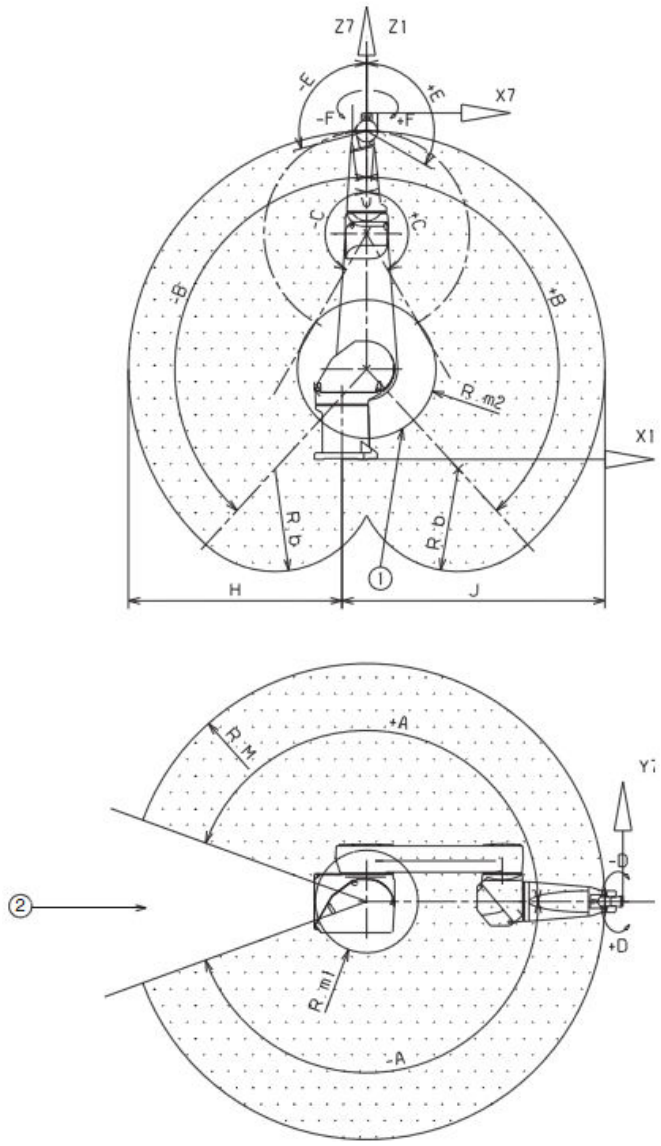

Figure 3.3 : RX-160 work envelope.

With reference to Figure 3.3, maximum reach between axis 1 and axis 5 (shown with R.M) is 1600 mm. Minimum reach between axis 1 and axis 5 (shown with R.m1) is 312 mm. Minimum reach between axis 2 and axis 5 (shown with R.m2) is 422 mm. The reach between axis 3 and axis 5 (shown with $R.b$) is 625 mm. The length shown with H is 1300 mm, and the length shown with J is 1600 mm [18].

The amplitude, speed and the resolution values of the axes are given on Table 3.1 [18].

Table 3.1 : Technical specifications of RX-160.

Axis	1	2	3	4	5	6
Amplitude (^{o})	320	275	300	540	225	540
Working Range Distribution (^{o})	± 160	± 137.5	± 150	± 270	$+120$ -105	± 270
Nominal Speed (^{o}/s)	165	150	190	295	260	440
Maximum Speed (^{o}/s)	200	200	255	315	390	870
Angular Resolution ($^o.10^{-3}$)	0.042	0.042	0.054	0.062	0.12	0.17

The load capacity of the arm is 20 kilograms with nominal speed, and 30 kilograms with reduced speed. The maximum load capacity of the arm is 34 kilograms [18].

3.2 Components

3.2.1 Controller

The controller which is used for RX-160 arm is called “CS8C”. It is located in the electrical cabinet where all electrical connections are collected. The controller is, in simple terms, the brain of the arm, including a processor (shown on Figure 3.4 with “5”). It controls the robot via digital power amplifiers (shown on Figure 3.4 with “1”) dedicated to each axis of the arm. The electrical power is converted b the power supply module (shown on Figure 3.4 with “7”). Robot power supply (shown on Figure 3.4 with “2”) is used in order to generate the DC voltage which is used for the amplifiers from a three phase AC voltage. The ARSP power supply (shown on Figure 3.4 with “3”) is the component which is used in order to generate DC voltage for the logic components of the controller. The components which are used in order to achieve electrical safety are grouped together on RSI board (shown on Figure 3.4 with “4”). And lastly, the main power switch is shown on Figure 3.4 with “8” [18].

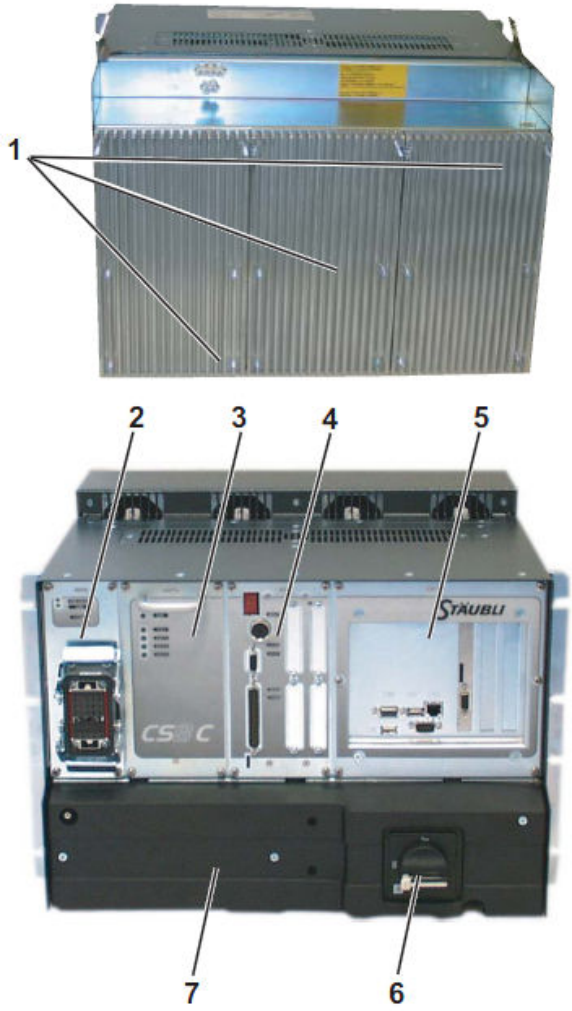

Figure 3.4 : Controller of the industrial arm.

3.2.2 Manual Control Pendant (MCP)

Manual control pendant can be used in order to enable arm power supply and control arm's motions. The use of this pendant is fundamental for applications which are studied; therefore the elements of pendant will be explained in detail. Pendant is shown on Figure 3.5 with elements grouped.

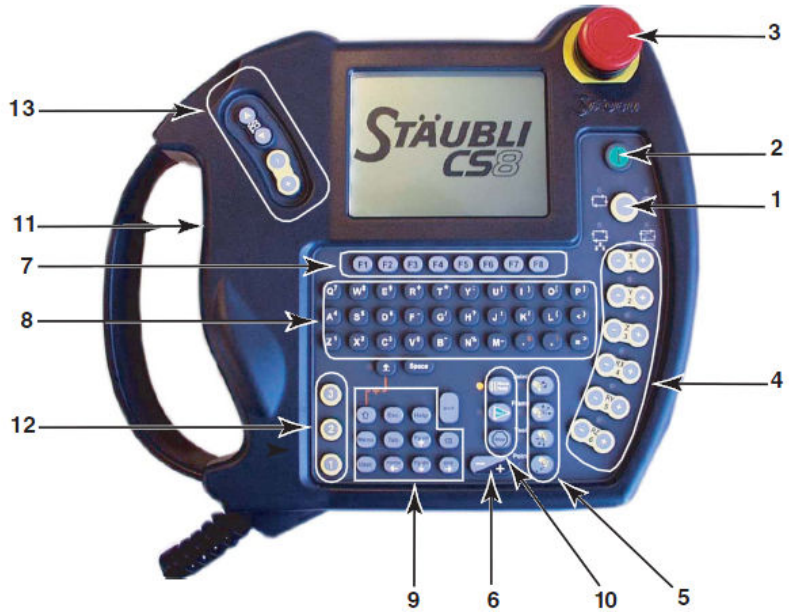

Figure 3.5 : Manual control pendant.

Working mode can be selected by the button shown on Figure 3.5 with “1”. The button indicated with “2” is the arm’s power button, if the green indicator light on this button becomes steady; it means arm power is on. The button shown with “3” is the emergency stop button. The motion keys are shown with “4”. They are used in manual mode, in order to generate motions depending on the motion mode selected. The keys shown with “5” are motion mode selectors. In manual mode they enable user to switch between joint, frame, tool, and point motion modes. The key shown with “6” allows the speed adjustment. The keys shown with “7” are the function keys, which are used to select menu above them. Alphanumerical keys are shown with “8”. They are used in order to enter data for applications. The keys shown with “9” are interface and navigation keys, which helps to use interfaces. The keys shown with “10” are application control keys. They are used in order to start, pause, and stop the applications which are created. The key shown with “11” is the enable button, which is also called dead man switch. It is used as a safety in manual mode when user needs to study close to the robot. This button has three positions; released, pushed down, and pushed down fully. User must keep this button in pushed down position in order to make robot move, otherwise the power of the arm will go off for safety. The keys shown with “12” are digital output activation keys. And lastly, the

key shown with “13” are jog keys which are activated in manual mode and enable user to generate arm motions [18].

With manual control pendant, user can not only move the arm manually but also can run the programs written; debug, change, and save them if needed, monitor the data gathered from inputs and outputs, and change system variables.

3.2.3 ATI Force-Torque sensor

In order to achieve applications which require force control, ATI Force-Torque sensor is used. This sensor’s transducer is mounted on the wrist of the arm, just before the end-effector as shown on Figure 3.6.

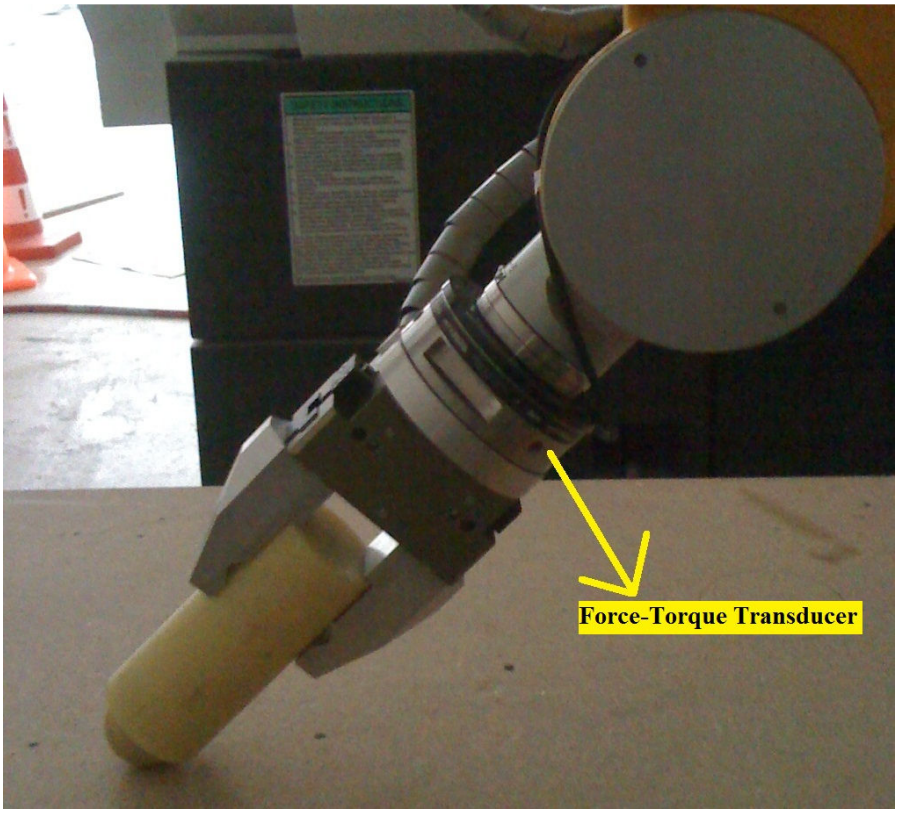

Figure 3.6 : F/T transducer on wrist of RX-160.

There are two main components of this force torque sensor system; the transducer and the controller (shown on Figure 3.7) [19]. The controller of our system is mounted on one side of the conveyor.

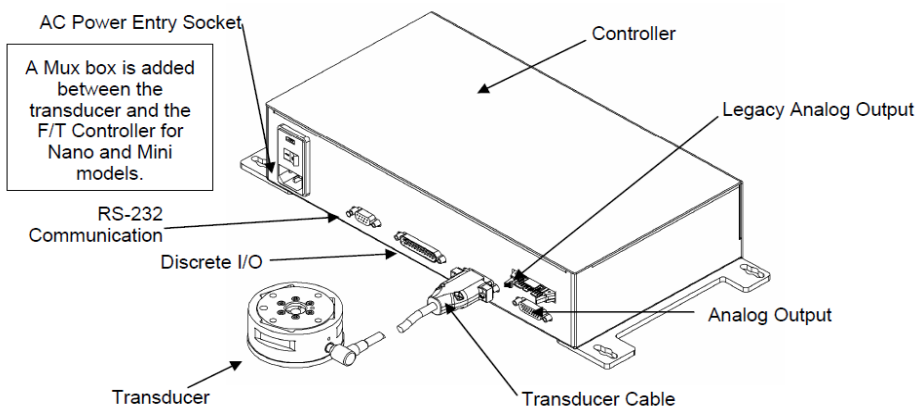

Figure 3.7 : F/T sensor system components.

3.2.3.1 Force-Torque transducer

The transducer is a compact, rugged, and monolithic structure which is used in order to convert force and torque measurements into analog strain gage signals for force-torque controller [19]. As mentioned before the transducer of our system is mounted on the wrist of the RX-160 arm, and it acquires overload pins which protects it from inadvertent overloads. The transducer is connected to the controller with the transducer cable as shown on Figure 3.7. The way how transducer is attached to the arm's wrist is shown on Figure 3.8.

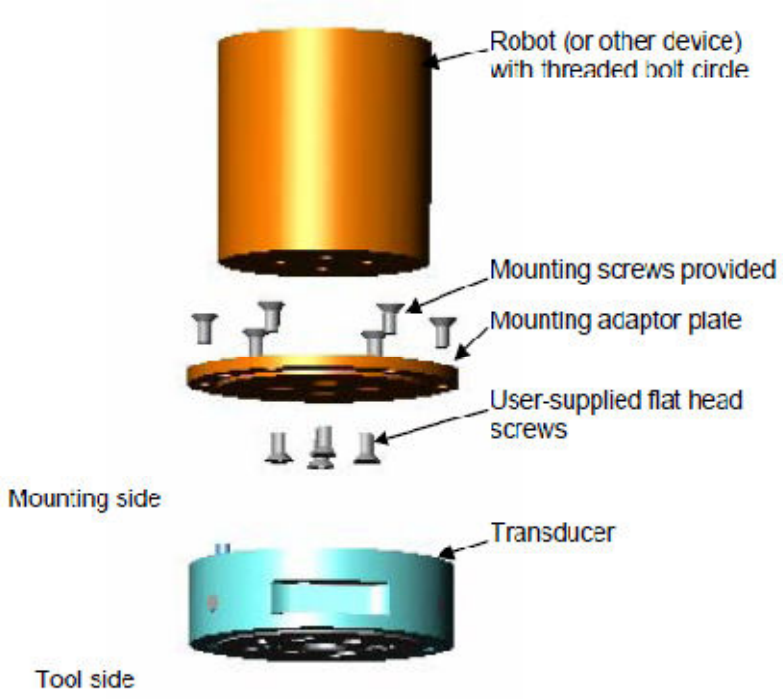

Figure 3.8 : F/T transducer.

The applied force and torque frame on transducer is given on Figure 3.9 [19]. The transducer should be placed on the arm's wrist in a way that the wrists frame and sensors frame coincides. Otherwise the measurements gathered from the sensor will be hard to interpret.

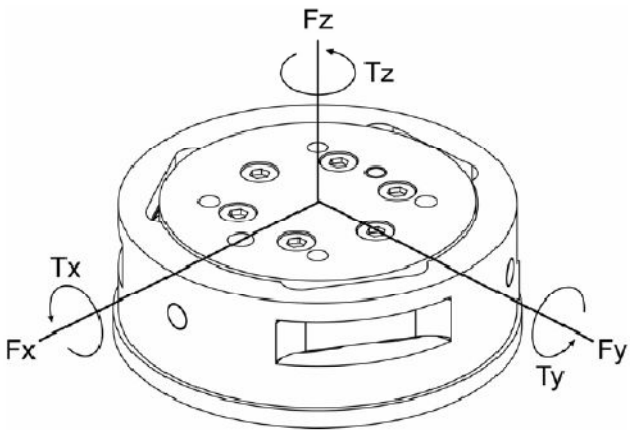

Figure 3.9 : Axes of F/T transducer.

The model of transducer used in our system is Delta. The calibration ranges in which Delta transducers can be used are given on Table 3.2 [19]. Our transducer is calibrated within SI-660-60 range.

Table 3.2 : Technical specifications of F/T transducer.

Calibration	Sensing Ranges				Resolution			
	Fx, Fy (N)	Fz (N)	Tx, Ty (Nm)	Tz (Nm)	Fx, Fy (N)	Fz (N)	Tx, Ty (Nm)	Tz (Nm)
SI-165–15	165	495	15	15	1/32	1/16	1/528	1/528
SI-330–30	330	990	30	30	1/16	1/8	5/1333	5/1333
SI-660–60	660	1980	60	60	1/8	1/4	10/1333	10/1333

3.2.3.2 Force-Torque controller

The primary function of the force-torque controller is to convert strain gage data to Cartesian force-torque components. The communication with controller can be done using serial I/O, the discrete I/O, or analog output [19].

The force-torque controller in our system is connected to the arm's controller with RS-232 serial I/O connection. The serial cable is connected to a WAGO terminal which allows connection through TCP/IP protocol with the robot's controller.

Using the serial connection with a computer, it is also possible to monitor measured data from the controller.

3.3 RX-160 Manipulator Kinematics

Until this section all the information that is needed in order to obtain the kinematic equations of the manipulator, which we will use for our applications, is gathered. The technical information including the dimensions of the arm is stated on section 3.1. Besides, the calculation methods of manipulator kinematics are stated on section 2.6. With all this information gathered, kinematic equations will be obtained by using the following steps.

At first, the frames will be attached to the joints of the arm. In order to achieve this, Denavit-Hartenberg convention method, which is mentioned on section 2.6.2.1, will be used. After frames are attached, the transition, rotation, and transformation

matrices will be calculated. These matrices will be defined as they give us the relationships between frames from base to end-effector. At last step, the Jacobian matrix of the arm will be calculated using the method mentioned on section 2.6.4.1.

The Jacobian matrix that is found after the end of the procedure is tested using the measurement data obtained from the arm. Results obtained are interpreted in order to confirm validity of Jacobian matrix.

3.3.1 Attaching frames to the joints

In order to attach frames to the joints of the arm the Denavit-Hartenberg convention is used. The exact locations of the frames placed are found with the help of the AutoCAD files of the arm. The AutoCAD file can be found in Appendix A.1, saved with the name “Frames.dwg”. Note that this file can only be opened with AutoCAD 2010 or later versions.

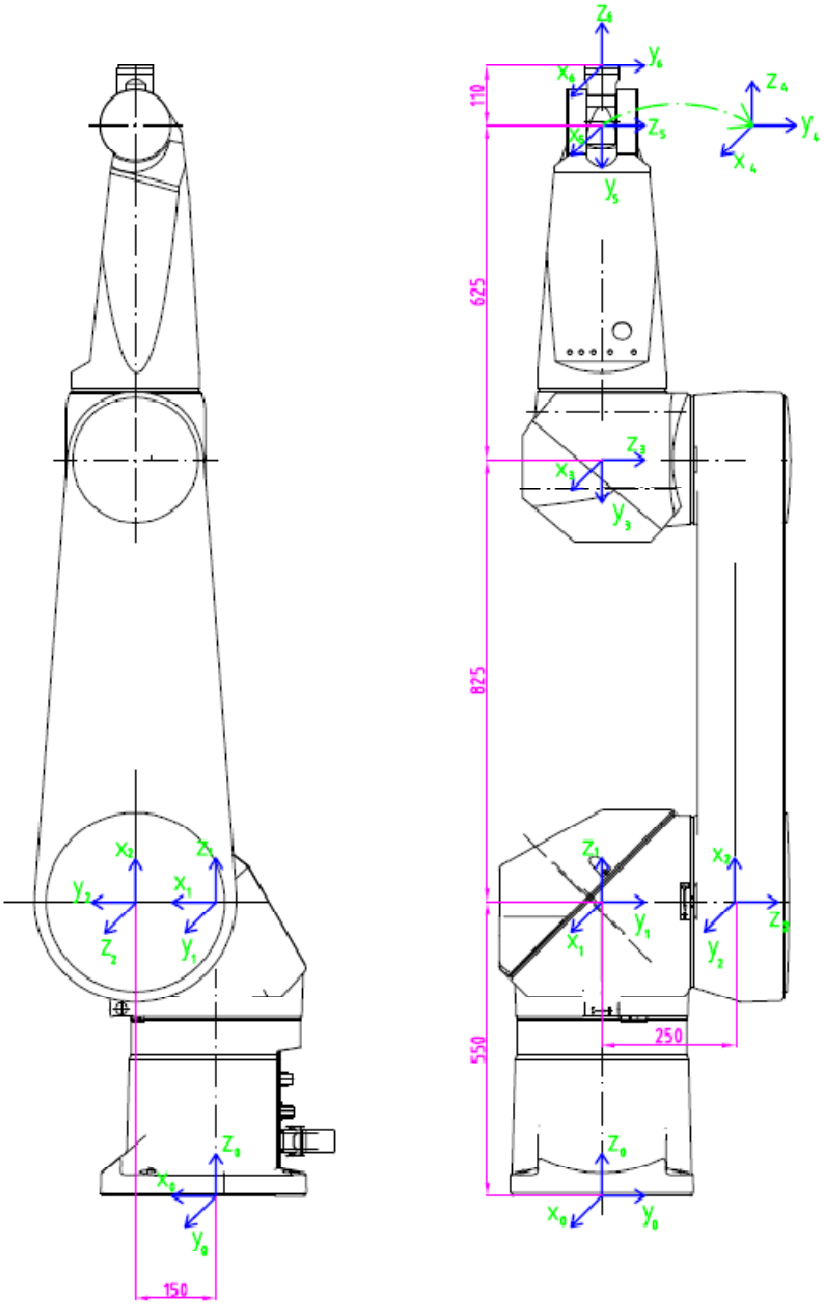

Figure 3.10 : RX-160 frame assignment.

Frames are started to be defined from the base, which is defined as frame 0. Step by step, through the base to end-effector frame, which is defined as frame 6, they are assigned. Placement of frames is shown on Figure 3.10.

At first, the Z axes of all frames are chosen along the axes of joints. The origins are placed on points where previous frame's Z axis intersects with common normal

defined on section 2.6.2.1. The X axes are chosen along the common normal to Z axis and previous joint's Z axis, from frame's origin to the next joint. And at last, the Y axes are chosen with the help of the right hand rule, using X and Z axes.

According to the frame placement, the Denavit-Hartenberg parameters are calculated. The Table 3.3 shows the parameters which will be used. On this table, the term "I" indicates the joint number. The term "a_i" indicates the distances between frame's origin and next frame's origin. The term "α_i" indicates the angle between axes z_i and z_{i-1} about axis x_i to be taken positive when rotation is made counter clockwise. The term "ϑ_i" indicates the angle between axes x_i and x_{i-1} about axis z_{i-1} to be taken positive when rotation is made counter-clockwise. And lastly, the term "d_i" indicates the coordinate of O_i along z_{i-1}.

Table 3.3 : DH parameters.

$\boldsymbol{i}$	$\boldsymbol{\alpha_{i-1}}$ (°)	$\boldsymbol{a_{i-1}}$ (mm)	$\boldsymbol{d_i}$ (mm)	$\boldsymbol{\vartheta_i}$(°)
Joint 1	0	0	550	θ_1
Joint 2	-90	150	250	$\theta_2 - 90$
Joint 3	0	825	-250	$\theta_3 + 90$
Joint 4	90	0	625	θ_4
Joint 5	-90	0	0	θ_5
Joint 6	90	0	110	θ_6

3.3.2 Calculating rotation and transformation matrices

On this section the rotation and transformation matrices will be calculated. The method which is illustrated on section 2.6.2.1 will be used again, with a slight difference. The transformation and the rotation matrices will be different this time. These matrices will be calculated using screw notation as our arm consists of only rotational joints, as using screw notation is more convenient for these kinds of arms [20].

The rotation operator which will be used is given on Equation 3.1.

$$R_i^{i-1} = \begin{bmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) & 0 \\ \sin(\vartheta_i)\cos(\alpha_{i-1}) & \cos(\vartheta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) \\ \sin(\vartheta_i)\sin(\alpha_{i-1}) & \cos(\vartheta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) \end{bmatrix} \quad \textbf{(3.1)}$$

The transformation operator which will be used is given on Equation 3.2.

$$T_i^{i-1} = \begin{bmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) & 0 & a_{i-1} \\ \sin(\vartheta_i)\cos(\alpha_{i-1}) & \cos(\vartheta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})\, d_i \\ \sin(\vartheta_i)\sin(\alpha_{i-1}) & \cos(\vartheta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.2)}$$

Because of the reason that calculations are too complicated and matrices are long, a MATLAB programs are prepared for an easy solution. These programs gave us the advantage of time and let us change our parameters if a mistake is done.

Because of the fact that matrices are calculated by MATLAB and they are very long, they will not be written here. Instead of that, the MATLAB program prepared for this reason will be explained step by step. The code itself can be found on Appendix A.1 saved with the name “Transformation .m”.

With reference to MATLAB file created, lines from 9 to 14 are used for assigning program variables as symbols. On line 17, the rotation operator matrix is defined which is given on Equation 3.1. The lines from 19 to 25 are used to substitute symbols with our parameters found with Denavit-Hartenberg convention. From line 27 to 31 the calculation of rotation matrix is given from base to end-effector as it is explained with Equation 2.8. On next step, the transformation operator matrix is defined on line 34. From line 36 to 42 the symbols are substituted with our parameters found with Denavit-Hartenberg convention. And lastly, on lines from 44 to 48 the transformation matrix is calculated as it is explained with Equation 2.15.

The results are saved on variable T $(:,:,1)$ for transformation matrix and R $(:,:,1)$ for rotation matrix.

3.3.3 Calculating Jacobian matrix

On this section Jacobian matrix will be calculated. Like in Section 3.3.2, the Jacobian matrix will be found with the help of MATLAB. The method which is told on

section 2.6.4 will be applied using transformation operator on Equation 3.2 and the rotation operator on Equation 3.1.

Because of the fact that matrices are calculated by MATLAB and they are very long, they will not be written here. Instead of that, the MATLAB program prepared for this reason will be explained step by step. The code itself can be found on Appendix A.1 saved with the name “Jacobian .m”.

With reference to MATLAB file created, lines from 9 to 14 are used for assigning program variables as symbols. On line 17, the rotation operator matrix is defined which is given on Equation 3.1. The lines from 19 to 25 are used to substitute symbols with our parameters found with Denavit-Hartenberg convention. From line 27 to 35, the unit vectors which indicate axes’ of joints are calculated. The reason why these vectors are calculated is that they are needed in Equation 2.31. The calculation method is given with the Equation 2.32 and Equation 2.33. On line 38 transformation operator is defined. From line 40 to 46 the symbols are substituted with our parameters found with Denavit-Hartenberg convention. From line 48 to 60 the vectors which indicate frames’ origins are calculated. The reason why these vectors are calculated is that they are needed in Equation 2.31. The calculation method is given with the Equation 2.34, Equation 2.35, and Equation 2.36. The Jacobian matrix is calculated in lines from 63 to 69. Line 71 can be used in order to simplify the Jacobian Matrix. And lastly the line 75 can be used in order to simplify the Jacobian Matrix using the angle values of joints.

The Jacobian matrix is saved on variable J after the program is run. And the matrix, in which the joint angle parameters are substituted, can be found in variable J_i.

3.3.4 Verifying Jacobian matrix

On Section 3.3.3 the Jacobian matrix is obtained using the methods given on Section 2.6.4 and with the Denavit-Hartenberg parameters found on Section 3.1.

On this section, the validity of the calculations which are done so far from attaching frames to the joints to the Jacobian calculation will be inspected and confirmed.

One of the easiest ways, to confirm our calculations, is to collect data from the robotic arm itself and compare it with the results which is found with the equations obtained In Section 3. In order to do so, the relationship between the end-effector angular and linear velocities and joint velocities will be used. As it is mentioned on Section 2.6.4 with Equation 2.24, the end-effector velocity can be calculated by multiplication of Jacobian matrix and joint velocities.

In order to achieve this, a program function which gathers velocity data of joints and the end-effector is prepared. This function can be run as a task while arm is commanded to do various motions on various speeds. The data gathered is saved to a file on arms controller with an extension ".log". This file can be downloaded to a computer and used with MATLAB by importing variables.

Using the program function mentioned above, a set of data is gathered, saved, downloaded to computer, and imported to MATLAB. The MATLAB program, which is used in order to verify our Jacobian Matrix, is given on Appendix A.1 saved with the name "Jacobian Check .m". This program is different from the program which is used in order to calculate Jacobian matrix with some lines. Only the addition lines will be explained here as the other lines are explained on Section 3.3.3.

With reference to the file mentioned above, the line 75 is used in order to define which element will be used from the data collected. The lines 81 and 82 are used to calculate the end-effector speed using the Jacobian matrix found. And lastly, the line 84 is used to define the actual end-effector speed which is obtained from the arm itself. The Jacobian can be tested with different sets of data gathered in different times by changing the value "a". The actual and the calculated velocities can be compared after the program is run. Note that the measurement data which is gathered with the execution of gathering program should be imported to MATLAB first. Otherwise, after the program is run MATLAB will return an error that there are some variables missing.

The data gathered from the robot can be found in Appendix A.1 with the file name "Jacobian Check .log". For easy access data is also saved as MATLAB workspace with the file name "Jacobian Check .mat". The program which is created in order to gather data from robot is explained on following chapters.

Now let us inspect some of the data collected for verification. Table 3.4 shows six different data sets gathered on different times.

Table 3.4 : Error in percent for calculated Jacobian.

$Time\ (s)$	Error In Percent					
	$\dot{p}_{ex}$	$\dot{p}_{ey}$	$\dot{p}_{ez}$	ω_{ex}	ω_{ey}	ω_{ez}
10 (a = 250)	0.0291	0.0645	0.0265	0.1232	0.0087	0.1509
19.6 (a = 490)	0.7271	0.7286	0.7276	0.7389	0.8609	0.6832
26 (a = 650)	0.6368	0.6371	0.6362	0.2063	0.6379	0.2638
31.6 (a = 790)	0.5174	0.5051	0.5125	5.2238	0.2691	0.9245
36.2 (a = 905)	0.0360	0.2325	0.6043	0.8695	0.0193	0.1633
42 (a = 1050)	0.0901	0.0407	0.1397	1.3394	0.0213	0.2384

The error in percent, which is shown on Table 3.4, is calculated using the Equation 3.3.

$$Error\ In\ Percent = 100 \times \frac{|\ Calculated\ Tool\ Speed - Actual\ Tool\ Speed\ |}{Actual\ Tool\ Speed} \qquad \textbf{(3.3)}$$

The data shown with $\dot{p}_e$ are the end-effector linear velocity errors, and the data shown with ω_e are the end-effector angular velocity errors.

With the help of the table, it ends up with the conclusion that the Jacobian, rotation, and the transformation matrices found are valid. The error which is observed is mostly because of the round-off error which is done on mathematical calculations.

3.4 Programming with VAL3 Language

VAL3 is a high level programming language designed to control Staubli Robots in all kinds of applications [18].

VAL3 language is created in order to combine the basic features of a real time high level computer language with the features which are needed in order control industrial robot cells [18]. These features are:

- Input and output control tools
- Robot control tools
- Geometrical control tools

In order to explain the programs which are used in applications, explaining the concept of using VAL3 language is fundamental.

In this section, the basic information which is needed in order to create programs, like language elements, simple types, user interface, and libraries will be explained briefly. On the other hand, the information which is needed in order to generate motion and achieve control functions will be explained on detail. These important subjects are tasks, robot control, arm positions, and motion control.

3.4.1 VAL3 language elements

VAL3 consists of the following elements [18]:

- Applications
- Programs
- Libraries
- Data Types
- Tasks

3.4.1.1 Applications

A VAL3 application is a self-contained software package which is designed for controlling the robots and inputs-outputs associated with the controller [18].

An application consists of the following elements [18]:

- A set of libraries
- A set of programs
- A set of global data

Additionally, when an application is running it can also contain a set of tasks, if programmed so.

3.4.1.2 Programs

A program is a sequence of VAL3 instructions to be executed. It is basically the essential part of the programming interface. Considering languages like C or C++, it can be said that programs are like the functions on VAL3 language.

A program consists of the following elements:

- The sequence of instructions
- A set of local variables
- A set of parameters

The set of local variables are used at the execution of the program and deleted after the program execution is finished. It can be said that they are temporary elements. On the other hand, the set of parameters are the input variables of the program, which can also be used as output variables.

In VAL3 there are two main programs which must be included mandatorily in our applications. These programs are start () and stop (). Start () program is initiated as an application is run. Stop () program is initiated if an application is stopped and will be closed

3.4.1.3 Data types

A VAL3 variable or constant type is characteristic which allow the system to control the applications and programs. All the variables and constants have a type. There are two main types, simple and structured.

VAL3 includes the following simple types:

- Bool type is used to define Boolean values.
- Num type is used to define numeric values.

- String type is used to define character strings.
- Dio type is used to define digital inputs and outputs.
- Aio type is used to define analog inputs and outputs.
- Sio type is used to define the inputs and outputs from serial port.

Structured includes fields which are used to combine many typed data. These fields can be accessed individually typing their names. VAL3 includes the following simple types:

- Trsf type is used to define Cartesian geometrical transformations.
- Frame type is used to define Cartesian geometrical frames.
- Tool type is used to define tools mounted on robot.
- Point type is used to define Cartesian positions of tool.
- Joint type is used to define the robots joint angles.
- Config type is used to define robot configurations.
- Mdesc type is used to define the robot motion parameters.

3.4.1.4 Libraries

Libraries are actually the applications which include the variables and programs that can be used occasionally.

3.4.1.5 Tasks

Tasks are programs that can run in background, while other processes are ongoing. Tasks play an important role in real-time processes; therefore they will be explained in detail on following sections.

3.4.2 User interface

In this section the user interface of the arm will be explained. Using the interface efficiently is a very important issue as it is the fastest way to interfere and monitor processes.

The interface itself is monitored from the screen located on the manual control pendant, which is shown on Figure 3.5.

In VAL3 language, the user interface instructions are used in order to [18]:

- Display messages on a page of the manual control pendant reserved for the application (User interface page).
- Acquire the keystrokes from the manual control pendants keyboard.

The user page on manual control pendant consists of fourteen rows and forty columns which are shown on Figure 3.11. The last row or line is generally used as the menu selection area which is mentioned on Section 3.2.2.

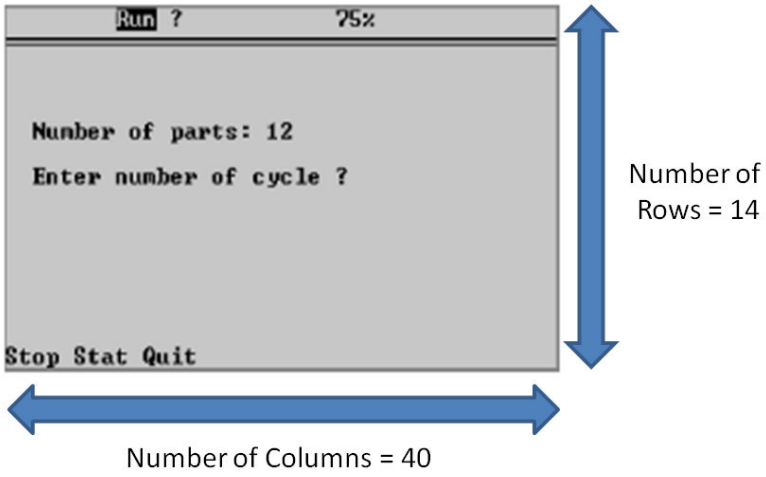

Figure 3.11 : User interface page dimensions.

The instructions which are related to the user interface are given below:

- Void $userPage()$ instruction is used in order to display the user page on manual control pendant.
- Void $gotoxy(num\ nX, num\ nY)$ instruction is used in order to position the cursor at the coordinate $(nX,\ nY)$. Coordinate of the top left corner is (0, 0), and the coordinate of the bottom right hand corner is (39, 13).
- Void $cls()$ instruction is used in order to clear the screen, and go to coordinate (0, 0).
- Num $getDisplayLen(string\ sText)$ instruction is used in order to obtain the length of string written on parameter section.
- Void $put(string\ sText)$ and Void $putln(string\ sText)$ instructions are used in order to display the specified parameters at the cursor position. The second instruction is used in order to set the cursors location to the next lines

first column after the text is written. On the other hand first instruction sets the cursor just after the text written.

- Num $getKey()$ instruction acquires a key strike from the control panel keyboard. If a key is pressed returns the code of the key. Otherwise returns -1.
- Void $popUpMsg(string\ sText)$ instruction is used in order to display pop up messages. The string given as the parameter is displayed on message box.

3.4.3 Tasks

In this section tasks will be inspected. In this thesis, most of the applications essential elements are tasks. In real time control and data gathering usage of tasks are mandatory.

As mentioned before a task in VAL3 language is actually a program, which can run in background synchronized with other tasks while other programs keep on running.

A task is defined by the following elements:

- A unique name which will be used as identifier.
- A priority or a period which will allow task sequencing possible.
- A program in which what task will do is defined.
- A status which is shows if task is running or stopped.
- The next instruction to be executed.

After a task is started and running in an application, the status of the task can be monitored from the task manager page which is located on the main page of the manual control pendant. If the task is stopped the status disappears from this page.

3.4.3.1 Task sequencing

When several tasks are running on an application, they appear to run concurrently and independently. If the application is observed over a sufficiently long period of time, it can be said this is practically true. On the other hand, if application is examined closely over a short period of time the facts change [18].

In fact, the system can only execute one task at a time, because of the fact that it has only one processor. Concurrent execution is simulated by very fast sequencing of the tasks that execute a few instructions in turn before the system moves on to the next task, as it is shown on Figure 3.12 [18].

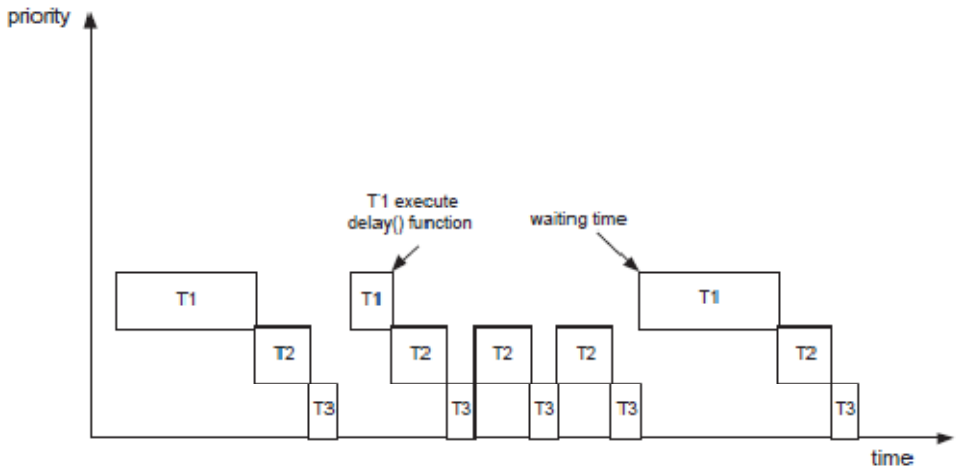

Figure 3.12 : Task execution times.

On Figure 3.12 there are three tasks, T1, T2, and T3. The tasks' instructions are executed for a long or short time depending on their priority levels. In other words, high priority means more instructions are executed.

While a task is being executed, if system meets a time delay or condition sequencing instruction, it can delay the execution of this program until desired conditions are met. Figure 3.12 shows us how system reacts if it meets delay instruction, it waits for the desired time given as parameter, and starts only after this time has passed.

The VAL3 instructions, which can cause a task to be sequenced immediately, are as follows [18]:

- The instruction $watch()$ which holds task for condition-wait timeout.
- The instruction $delay()$ which holds task for timeout.
- The instruction $wait()$ which holds task for condition waiting time.
- The instruction $waitEndMove()$ which holds task for waiting arm stop time.
- The instructions $open()$ and $close()$ which holds task for opening and closing tool.
- The instruction $get()$ which holds task for keystroke waiting time.

- The instruction $taskResume()$ which holds the system until the task is ready for restart.
- The instruction $taskKill()$ which holds the system until task is actually terminated.
- The instruction $disablePower()$ which holds the system until power is actually gone off.
- The instructions accessing the contents of the disk ($libLoad()$, $libSave()$, $libDelete()$, $libList()$, $setProfile()$)
- The serial reading and writing instructions ($sioGet()$, $sioSet()$)
- The instructions $setMutex()$ which holds task for the Boolean mutex to be false.

3.4.3.2 Synchronous tasks

Sometimes tasks are needed to be scheduled so that they are run in regular periods of time. This need comes up with the applications like data acquisition, real time control, or external device control [18].

The synchronous tasks are executed in the sequencing cycle by interrupting the current asynchronous task between two VAL3 lines [18]. Unlike most of the programmable logic controller languages or high level languages, the interrupt doesn't need to be done using a parameter, but done in time defined.

The sequencing of the VAL3 synchronous tasks obeys the following rules [18]:

- Each synchronous task is sequenced exactly once per period of time specified at the task creation.
- At each sequence, the system can execute up to 3000 VAL3 instruction lines. If sequence couldn't be completed after 3000 lines, it shifts to the next task.
- The synchronous tasks which are defined with the same period are sequenced in order which they are created.

If a task is being used which has the possibility of going over 3000 lines, it is a necessity to use delay instruction in order to achieve operations with success in time.

Actually, the tasks are generally ended with the instruction “delay (0)” in order to avoid overruns.

3.4.4 Arm positions

In this section the data types and instructions which are used in order to program the arm positions in VAL3 applications are explained.

3.4.4.1 Arm position types

To start with, the types which are used for arm position operations will be explained:

- In VAL3 there are two position types defined. These are joint positions which are used in order to give the angular information about joints, and Cartesian points which are used in order to give the Cartesian position of the tools center relative to a reference frame [18].
- The tool type is used in order to describe the tool and its geometry. Additionally, it gives information how tool can be activated [18].
- The frame type is used in order to describe geometrical reference frames. Geometrical point manipulation is more simple and heuristic with the use of frames [18].
- The trsf type is used in order to define geometric transformations. Frame, point, and tool types use these transformations [18].
- The Config type is used in order to describe advanced configurations of the arm during motions.

3.4.4.2 Arm position instructions

Besides, let us explain the instructions which are used for arm position operations:

- The instruction $joint\ herej()$ gives the angular information of the current arm position.
- The instruction $point\ here()$ gives the Cartesian information of the current arm position.
- The instruction $num\ distance(trsf\ trPos1, trsf\ trPos2)$ returns the distance between two given parameters.

- The instruction $trsf\ interpolateL(tStart, trsf\ tEnd,\ num\ nPosition)$ is used to return an intermediate position aligned with a start position and a target position which are given as the parameters of instruction. The last numerical parameter is used in order to define how close the point will be to start point. If it is defined as 0 the point is at start, on the other hand if it is defined as 1 the point is at end.
- $trsf\ interpolateC(trsf\ tStart, trsf\ tIntermediate, trsf\ tEnd,$ $num\ nPosition)$ Instruction is used to return an intermediate position on the arc of a circle whose center, start, and end points are given as parameters. The last numerical parameter is used in order to define how close the point will be to start point. If it is defined as 0 the point is at start, on the other hand if it is defined as 1 the point is at end.
- $num\ setFrame(pOrigin, pAxisOx, point\ pPlaneOxy, frame\&\ fResult)$ Instruction is used to calculate the new frame with reference to the given parameters. This function returns "0" if there are no errors, "-1" if the x point is chosen wrong, "-2" if the plane is chosen wrong.
- The instruction $trsf\ position(frame\ fFrame, frame\ fReference)$ returns the coordinates of the target frame which is defined as first parameter, with reference to reference frame which is defined as second frame.
- The instruction $void\ open(tool\ tTool)$ is used to activate the tool attached to the wrist, with setting the digital output which is related to the tool to true.
- The instruction $void\ close(tool\ tTool)$ is used to deactivate the tool attached to the wrist, with setting the digital output which is related to the tool to false.
- The instruction $trsf\ position(tool\ tTool, tool\ tReference)$ is used to return coordinates of the target tool which is given with first input parameter, with reference to the frame where reference tool is defined.
- $point\ compose(pPosition, fReference, trsf\ trTransformation)$ Instruction is used to return a position, which found by applying geometrical transformation to target position relative to reference frame.

- The instruction $point\ appro(point\ pPosition,\ trsf\ trTransformation)$ is used in order to obtain the point which is found by applying geometrical transformation to target position relative to the frame which target point is defined.
- $point\ jointToPoint(tool\ tTool, frame\ fRef, joint\ jPosition)$ Instruction is used in order to return the position of the tool when the arm is in the joint position defined with the last parameter of instruction.
- $bool\ jointToPoint(tool\ tTool,\ joint\ jInitial,\ point\ pPos,$ $joint\&\ jResult)$ Instruction is used in order to obtain the joint position of the arm corresponding to the Cartesian position of the point using the given parameters.

3.4.4.3 Arm configurations

Generally, there exists several ways for the robot to reach to a desired point. These possibilities are called as configurations [18]. In this section configurations will be explained.

In VAL3 the configurations for given Cartesian points are defined using the Config type. This type consists of data fields which are shoulder, elbow, and wrist. The shoulder, elbow, and elbow fields can have the following values given on Tables 3.5, 3.6, and 3.7 [18].

Table 3.5 : Shoulder configurations.

Field	Configuration	Explanation
Shoulder	$righty$	$righty$ shoulder configuration imposed
	$lefty$	$lefty$ shoulder configuration imposed
	$ssame$	Shoulder configuration change is not allowed
	$sfree$	Free shoulder configuration

Table 3.6 : Elbow configurations.

Field	Configuration	Explanation

Elbow	$epositive$	$epositive$ elbow configuration imposed
	$enegative$	$enegative$ elbow configuration imposed
	$esame$	Elbow configuration change is not allowed
	$efree$	Free elbow configuration

Table 3.7 : Wrist configurations.

Field	Configuration	Explanation
Wrist	$wpositive$	$wpositive$ wrist configuration imposed
	$wnegative$	$wnegative$ wrist configuration imposed
	$wsame$	Wrist configuration change is not allowed
	$wfree$	Free wrist configuration

The configuration types which are explained on previous tables can be understood more easily by inspecting Figure 3.13, 3.14, and 3.15 [18].

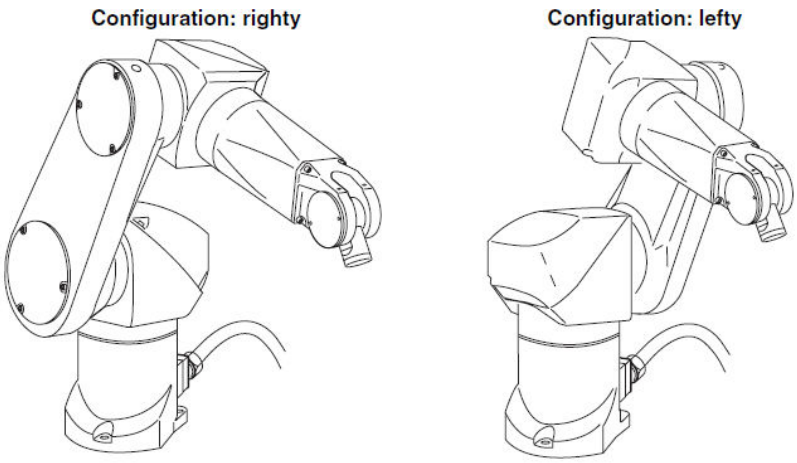

Figure 3.13 : Shoulder configurations.

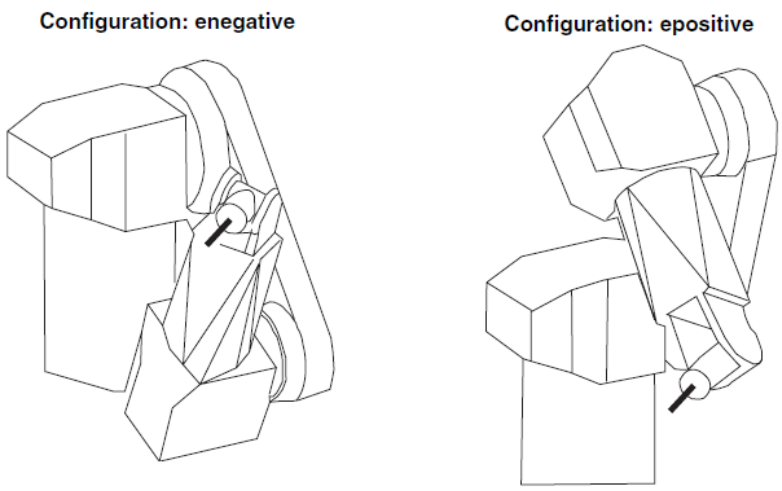

Figure 3.14 : Elbow configurations.

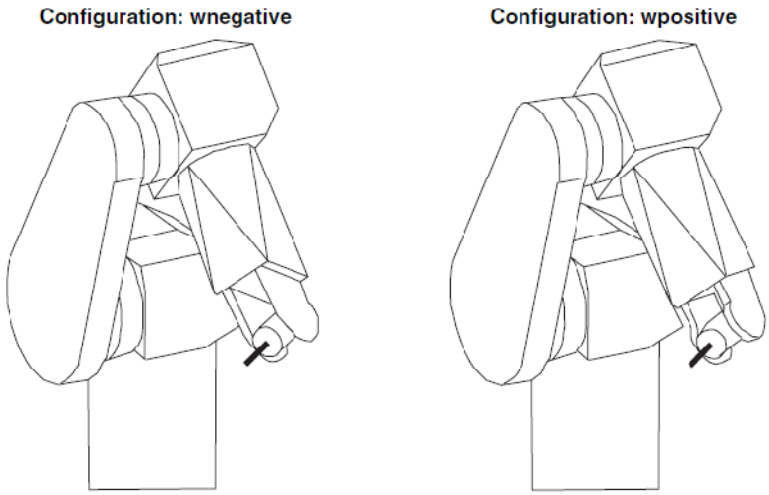

Figure 3.15 : Wrist configurations.

3.4.5 Motion control

In this section of the thesis the way how motion control is achieved is explained. For this reason, the motion descriptor type which will be used almost for all instructions will be inspected at first. Next, the instructions which allow motion control will be explained step by step.

3.4.5.1 Motion descriptor type

The essential data type which is used for motion control is motion descriptor. The motion descriptor is indicated by $mdesc$ in VAL3 language.

The $mdesc$ type is used in order to define motion parameters (speed, acceleration, blending) [18].

The $mdesc$ type is a structured data type, which consists of the fields shown on Table 3.8 [18].

Table 3.8 : Motion descriptor fields.

Data Field Name	Explanation
$num\ accel$	Maximum permitted joint acceleration
$num\ vel$	Maximum permitted joint speed
$num\ decel$	Maximum permitted joint deceleration
$num\ tvel$	Maximum permitted transitional speed of tool center point
$num\ rvel$	Maximum permitted rotational speed of the tool center point
$blend$	Blend mode (on or off)
$num\ leave$	Leaving distance in blend mode
$num\ reach$	Reaching distance in blend mode

3.4.5.2 Motion instructions

In VAL3 language $void\ movej(point\ pPos, tool\ tTool, mdesc\ mDesc)$ instruction can be used in order to execute joint motions. The motion is done according to the parameters given which are target position, tool that will be used, and lastly the motion descriptor.

The target position can be defined as Cartesian space point or joint point. A runtime error will be generated if a motion that cannot be achieved is ordered.

In VAL3 language $void\ movel(point\ pPos, tool\ tTool, mdesc\ mDesc)$ instruction can be used in order to execute linear motions. The motion is done according to the parameters given which are target position, tool that will be used, and lastly the motion descriptor.

On contrast to joint motions, linear motions can only take Cartesian points as target, but not joint positions. A runtime is generated in case the motion cannot be achieved.

In VAL3 language $void\ movec(pInt, point\ pTar, tool\ tTool, mdesc\ mDesc)$ instruction can be used in order to execute circular motions. The motion is done according to the parameters given which are circular motion's center point, target position, tool that will be used, and lastly the motion descriptor.

The target point's position can be defined only in Cartesian space. A runtime error will be generated if a motion that cannot be achieved is ordered.

In VAL3 language $void\ stopMove()$ instruction can be used in order to stop the arm on the trajectory and suspend authorization of the programmed motion.

The motion can only be resumed after $restartMove()$ or $resetMotion()$ instructions. On the other hand, non-programmed motions are still possible.

In VAL3 language $void\ resetMotion()$ instruction can be used in order to stop the arm on the trajectory and cancel all the stored motion commands.

In VAL3 language $void\ restartMove()$ instruction can be used in order to restart the motion on trajectory which is interrupted by $void\ stopMove()$ instruction.

Note that the motions which are interrupted by $resetMotion()$ cannot be restarted by this command.

In VAL3 language $void\ waitEndMove()$ instruction can be used in order to wait for the end of the motion which is ordered previously.

In a case which $waitEndMove()$ is used, the blending mode which is defined using motion descriptor will be avoided.

In VAL3 language $bool\ isSettled()$ instruction can be used in order to check if robot is moving or not. Instruction will return true if robot is stopped, otherwise it will return false.

In VAL3 language $num\ getSpeed(tTool)$ instruction can be used in order to obtain the current Cartesian transitional speed at the end of the tool specified with parameter.

This instruction computes speed from the joint velocity command but not from the joint velocity feedback.

In VAL3 language $num\ getPositionError()$ instruction can be used in order to obtain the current joint position error of the arm.

The joint position error is the difference between the joint position command and the joint position feedback measured by the encoders.

In VAL3 language $void\ getJointForce(num\&\ nForce)$ instruction can be used in order to obtain the forces on the robot's joints. Results are saved in parameter given with instruction.

3.4.6 Real-Time motion control

3.4.6.1 Compliant motions with force control

The motion commands which are previously explained only allow robot motions to reach a desired position at a previously programmed acceleration and speed. In these motions, if it is not possible for arm to follow the command, in order to reach the desired position, additional force will be requested from the motors. A system error, which also cuts off the power to the arm, is generated if the deviation between the position set by the command and the true position is too great [18].

When the robot accepts certain deviation between the position set by command and the actual position, it is said to be "compliant". By controlling the force applied by the arm, the controller can be programmed to be trajectory compliant which will allow it to accept a delay or advance in relation to the programmed trajectory [18].

Practically speaking, the arm can follow a trajectory while being pushed or pulled by an outside force, or come into contact with an object, with a check made on the force applied by the arm on the object, using the VAL3's compliant motions [18].

While using force control, if the specified force parameter is null, the arm is passive. In other words it only moves when actuated by outside forces [18].

When the force parameter is positive, everything operates as though an outside force are pushing the arm to the position ordered. In other words, the arm moves on its own, but it can be held back or accelerated by outside action which is added to the force commanded [18].

When the force parameter is negative, everything operates as though an outside force are pushing the arm towards its initial position. In other words, to move the arm towards the position commanded, it is thus necessary to apply an outside force that is greater than the force commanded [18].

Compliant motions have the following limitations [18]:

- The blending is not possible at the end or start of a compliant motion. It is mandatory for arm to stop at the start and end of every compliant motion.
- When a compliant motion is being executed, the arm cannot move back any further than the point it started. The arm will stop suddenly if it is moved to the starting point.
- The force parameter on the arm cannot exceed 1000%.
- A lot of internal memory capacity is required for long compliant motions. A runtime error is generated if the system doesn't have enough memory to fully process the motion.

The instructions which are used in order to achieve compliant motions are not very different from the standard motion instructions.

The instructions which allow compliant motions are given below:

- $void\ movejf(joint\ jPos,\ tool\ tTool,\ mdesc\ mDesc,\ num\ nForce)$. This instruction allows a compliant joint motion towards the target joint position with tool and motion descriptor defined in parameter section. The last parameter defines the force command which is a numerical value representing the arm force and cannot exceed ± 1000. A value of 100 closely matches the weight of the nominal mass of the arm.

- $void\ movelf(point\ pPos,\ tool\ tTool,\ mdesc\ mDesc,\ num\ nForce)$.
  This instruction allows a compliant linear motion towards the target position with tool and motion descriptor defined in parameter section. The last parameter defines the force command which is a numerical value representing the arm force and cannot exceed ± 1000. A value of 100 closely matches the weight of the nominal mass of the arm.
- $bool\ isCompliant()$. This instruction is used in order to check if robot is working in compliant mode. If the robot is in compliant mode instruction returns true, else it returns false.

3.4.6.2 Alterable motions

Geometrical transformations (translation, rotation, rotation at the tool centre point) on a path, which are immediately effective, are possible with the use of Cartesian alterations [18].

Using alterable moves, it is possible to modify a nominal path with the measurements gathered from an external sensor [18]. This feature of VAL3's alterable motion option is a key point for our applications.

In order to program alterable motions, firstly the nominal path must be defined, then, in real time, a deviation must be specified for it. Several alterable moves may succeed, or some alterable moves may alternate with not alterable moves. Therefore, the alterable path is defined as the successive alterable move commands between two not alterable move commands [18].

In order to achieve alterable motions, some principals must be taken into consideration. These are principals are given below [18]:

- The change in alteration must be controlled so that the resulting arm path remains without discontinuity or noise; this is because of the fact that alteration commands are applied immediately. Large changes in alter deviation may cause robot to stop with a system error as it cannot move that fast. Besides, at the end of altering a null alteration speed is required in order to stop robot properly. The input from the sensor which will be used as

control feedback can be filtered in order to avoid noise which will cause immediate changes in motions.

- Because of the fact that controller sends position and velocity commands to the amplifiers every 4 ms, the alter command must be synchronized with this communication period so that alteration speed remains under control. As mentioned on Section 3.4.3.2, this can be achieved by synchronous VAL3 tasks.
- The first non-alterable move following an alterable move can be computed only after alter motion is ended. Besides, while in alter mode robot can only work in configuration which nominal alterable path started. Because of this reason singularities which may occur during alterable motions must be investigated in detail.

Besides, while using alterable motion options in VAL3 a null motion is ignored by the system. Therefore, a move that is not null is required for the robot to enter alter mode.

In this section, the instructions which are needed in order to use alterable motion option in VAL3 are explained.

The instruction $void\ alterMovej(joint\ jPos, tool\ tTool, mdesc\ mDesc)$ is used in order to register an alterable joint move command. The arm is programmed to make a joint motion to the desired joint point, with desired tool and motion descriptor defined in parameters.

The instruction $void\ alterMovel(joint\ jPos, tool\ tTool, mdesc\ mDesc)$ is used in order to register an alterable linear move command. The arm is programmed to make a linear motion to the desired point defined in Cartesian space, with desired tool and motion descriptor defined in parameters.

The instruction $void\ alterMovec(pInt, point\ pTar, tool\ tTool, mdesc\ mDesc)$ is used in order to register an alterable circular move command. The arm is programmed to make a circular motion to the desired point defined in Cartesian space, with desired tool and motion descriptor defined in parameters. The center of the circle must be defined in first parameter.

The instruction $num\ alterBegin(frame\ fAlter, mdesc\ mMaxVel)$ is used in order to initialize the alter mode for the alterable path which is being executed. This instruction returns a numerical value depending on the result of execution.

The instruction $num\ alterEnd()$ is used in order to exit the alter mode and make the current move not alterable anymore.

The instruction $num\ alter(trsf\ trAlteration)$ is used to specify a new alteration of the alterable path. Transformation parameter is used in order to define the alteration deviation to apply until the next alter instruction.

The alter mode is activated with the $alterBegin$ instruction and it is only terminated with $alterEnd$ or $resetMotion$ instructions. When the end of an alterable path is reached, the alter mode remains active until $alterEnd$ is executed.

The alteration is applied by the system every 4ms. If several instructions are executed less than 4ms, the last one will be applied. Actually, it is the most convenient way to create a synchronous task with 4ms cycle time and refresh alter commands in this time even a null move is required.

4. MOTION AND FORCE CONTROL

In this section the control methods which are used for our applications will be explained. On the first step, motion control with the use of CAD data will be explained. Then, force control will be introduced. And lastly, hybrid control which combines CAD tracking and force control will be explained.

4.1 Motion Control with CAD Data

4.1.1 Introduction

If a robotic manipulator is to be used performing useful work, it must be programmed to accomplish the desired task or motion cycle [21]. Conventional robot programming in industry, using the robot manual teach control pendant, is a laboring and time consuming task which also requires a programmer with recent experience [17]. In order to avoid this problem, new ways are being inspected for making robot programming easier and faster. Actually, the aim is to create an interface which even a programmer, who has no experience on robotics, can interact with robot very easily and fast while the program created handles the process on a perfect way.

In this section of the thesis, our aim will be to introduce an interface which will be working with the help of computer aided design (CAD) data. The purpose of this interface will be to allow a user, who has the basic CAD skills, program the robot in an effective way.

This interface will not only help the programmer but will also allow flexible manufacturing processes. As the production lines today are built up to produce a wide variety of products, the robots should also be able to adapt changes in production in a fast way. Using CAD data to solve this problem is one of the efficient ways since almost all products have CAD data today.

Imagine that you want your robot to follow the surface of a complex object. Using the conventional methods, programming would be very hard and long, even impossible sometimes, many points on the target object would have to be defined. CAD data tracking method will be used in order to overcome this programming problem.

In this section, firstly how to create the CAD data files, including 2D and 3D drawings, will be explained. Next, importing the CAD files to the robotic arm's controller will be explained. On next step, the inspection and point extraction from imported data will be explained. Lastly, converting points gathered to robotic transformations and achieving the motions using these transformations will be explained.

4.1.2 Creating CAD files

One of the most popular software which is used in order to deal with CAD files is AutoCAD. It is developed and sold by Autodesk, Inc. It was firstly released in 1982 and it was one of the first CAD programs which can run on personal computers. AutoCAD will be used for creating CAD files for applications as it is used widely in industry and as it provides powerful tools to modify our drawings easily.

The AutoCAD file format that will be used is DXF, the drawing exchange format. This file format is widely used because of the fact that it provides an effective and easy data exchange environment. Other native file formats like DWG, DWS, and DWT can be converted easily to DXF format.

The DXF files consist of tagged data. Tagged data means each data element in file is preceded by an integer number that is called a group code. A group code's value indicates what type of data element follows. This value also indicates the meaning of a data element for a given object (or record) type.

DXF files can be read and modified by text editors. The basic organization of a DXF file is given with the Table 4.1.

Table 4.1 : DXF file section explanations.

Section Name	Explanation
HEADER	General information about the drawing. Each parameter has a variable name and an associated value.
CLASSES	Holds the information for application-defined classes whose instances appear in the Blocks, Entities, and Objects sections of the database.
TABLES	This section contains definitions of named items.
BLOCKS	This section contains Block Definition entities describing the entities comprising each Block in the drawing.
ENTITIES	This section contains the drawing entities, including any Block References.
OBJECTS	Contains the data that apply to non-graphical objects, used by Auto LISP and Object ARX applications.
Thumbnail Image	Contains the preview image for the DXF file.
End of File (EOF)	Defines the end of file.

The DXF file format will be inspected in order to acquire point data in the following sections.

4.1.2.1 Creating 2D drawings

Let us start creating CAD files which will contain the tracks for robot with 2D drawings. In this section the key points of creating files for tracks will be underlined, but the following sections will give more detailed information depending on applications studied.

The classical AutoCAD interface opens with 2D drawing tools in the first place. Therefore, the interface can be used with no changes required in order to create tracking files.

Let us assume that it is wanted to draw a triangular track whose sides are on points (0, 0), (200, 0), and (100, 100) in Cartesian space. Note that all the dimensions

mentioned here will be in millimeters. Using the line or poly line commands the triangle will be created at first. The origin defined with the point (0, 0) in Cartesian space will always be start position of tracking position of our applications.

After drawing the whole track to follow, the track must be divided into smaller parts. This is because of the fact that the robot will be performing motions with the combination of many small motions. The points will be used in order to divide the track. Actually the robot will move to the points; one after another in order to achieve desired motion in track. The points can be created by the divide or measure drawing commands. The track will be divided into smaller parts with the length of 10mm in this application.

After drawing is completed the file must be saved as DXF file. In our applications, DXF files are always saved in AutoCAD 2007 compatible mode in order to increase compatibility with older versions of AutoCAD.

The AutoCAD file saved with the name “01triangle.dxf” can be found on Appendix A.1. This file is added in appendix, in order to demonstrate the drawing which is meant to be created. The drawing is also shown on Figure 4.1.

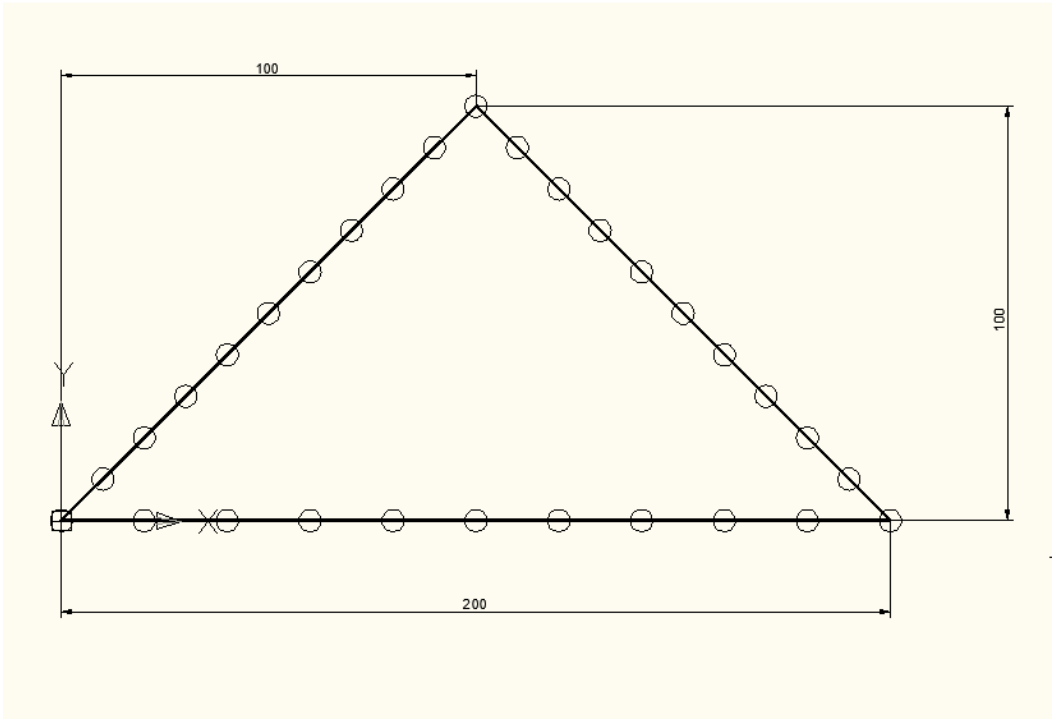

Figure 4.1 : 2D triangle drawing.

4.1.2.2 Creating 3D drawings

Like the previous section, the key points of creating files for tracks will be underlined, but the following sections will give more detailed information depending on applications studied.

If the classical AutoCAD interface is being used then the 3D tools must be enabled in order to work with three dimensions. Switching to 3D modeling workspace is the fastest way to draw in 3D.

Let us assume that it is wanted to draw a helix as a path for robot to follow. First a line from origin (0, 0, 0) to (70.7, 70.7, 0) will be drawn which will create a line 100mm long. The point (70.7, 70.7, 0) will be our helix's starting point. Next, a helix starting at this point, with a radius of 80mm and height of 300mm is drawn. The helix's endpoint should be at point (127.28, 127.28, -300) after drawing is completed. The helix which is expected to be drawn is shown on Figure 4.2.

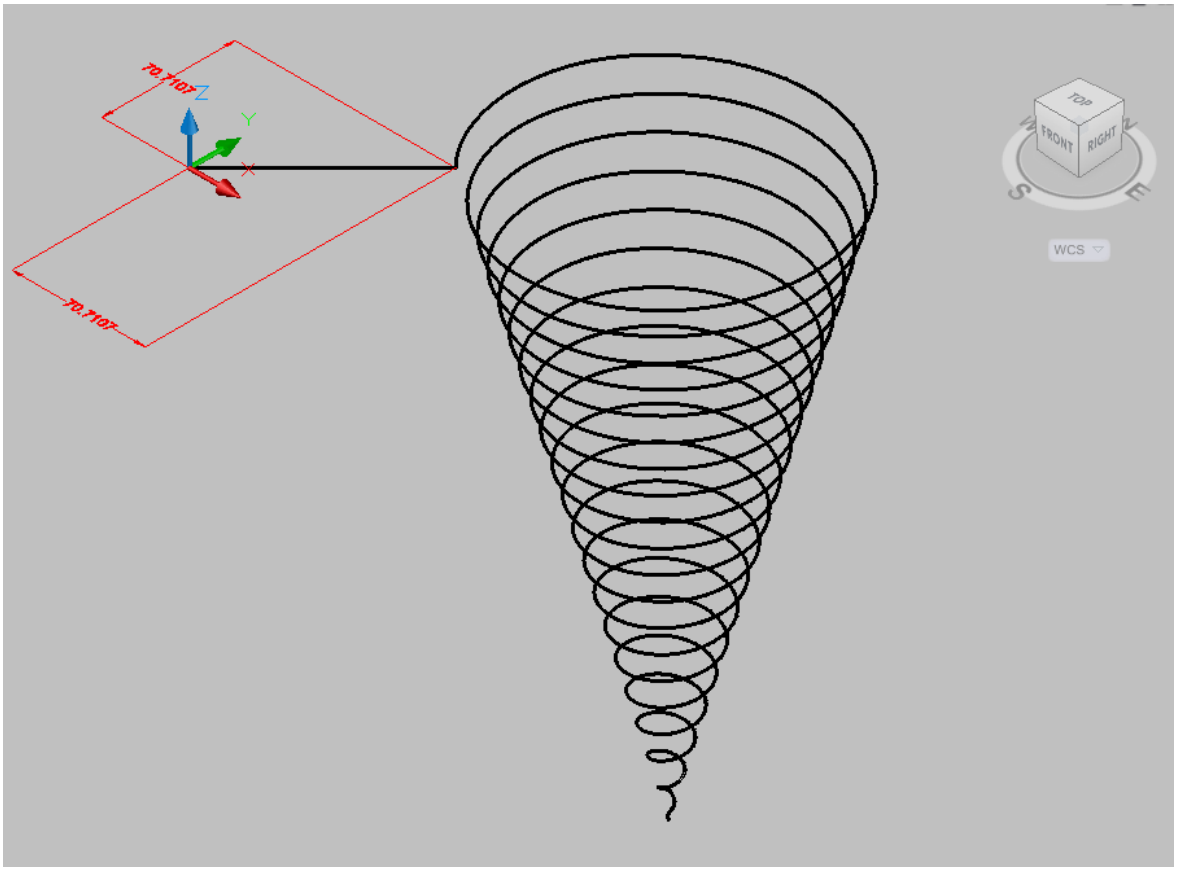

Figure 4.2 : 3D helix drawing.

After the helix path to be followed is drawn, it will be divided into smaller paths. As mentioned on previous section, this is because of the fact that robot will be performing motions with the combination of many small motions. Like before division process will be handled using points. The points can be created by the divide or measure drawing commands of AutoCAD. Let us divide the track with the length of 20mm like the previous section. The result obtained is given on Figure 4.3.

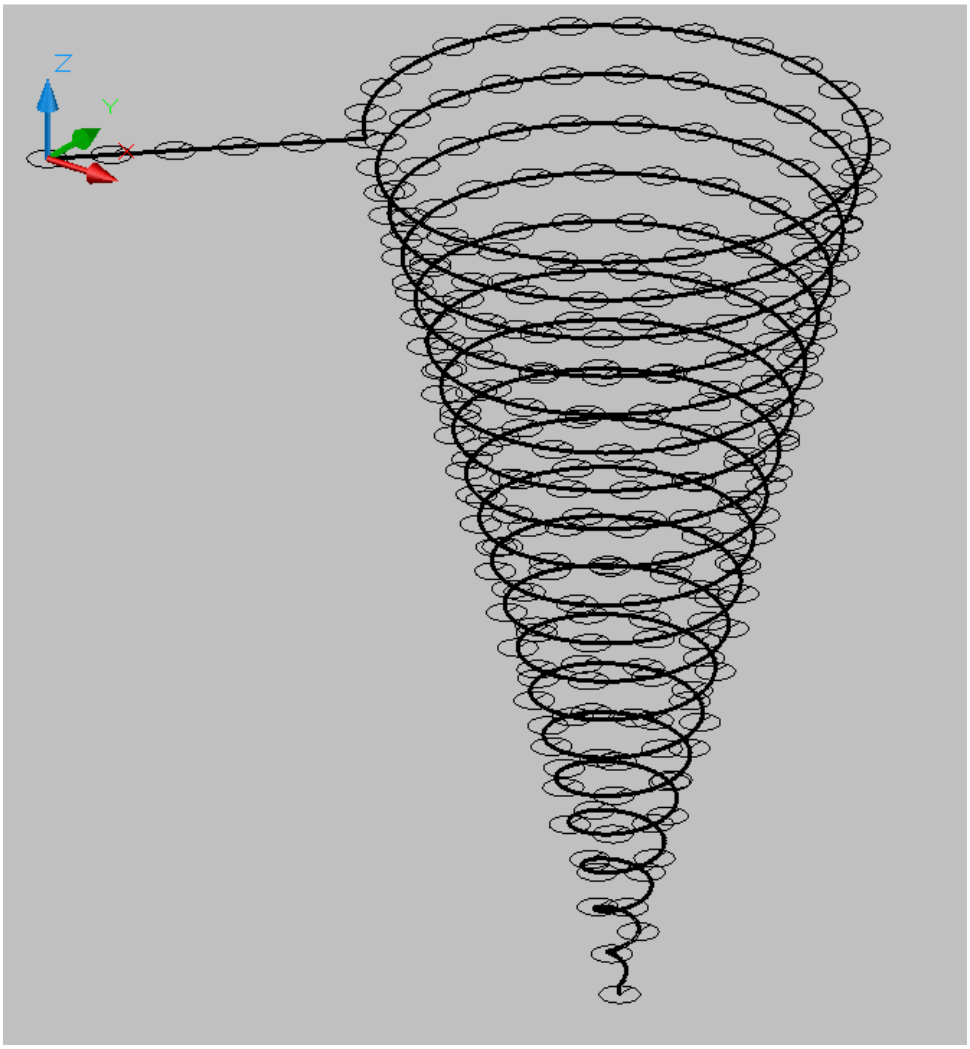

Figure 4.3 : 3D helix drawing with points.

Inspecting the drawing created, it is easy to see that the points which are created at the tail section of the helix are simply not enough to successfully follow the path desired. There are two options to choose in this situation. First option is to add more points to tail section of the helix where the points are not enough in order to follow path properly. If this option is chosen robot will move slower in these points as the

deviation will be smaller than the previous intervals. Second option is to divide the path into smaller parts with the length of 10mm. This option will allow the robot to move in a constant velocity but will increase the number of points greatly making the controller of the robot busy and strains it.

Choosing the deviation of small paths is explained in a more detailed way in the following sections, where alterable motion applications and 3D CAD data tracking applications are studied.

The helix drawn in AutoCAD can be found in Appendix A.1 with the filename "02helix.dxf". This file is added in appendix, in order to demonstrate the drawing which is meant to be created.

4.1.3 Importing CAD files

The exchange of CAD data between the robot's controller and software where data is created is one of the most important issues in CAD data tracking process. In this thesis, CAD files will be created using AutoCAD in personal computers. The drawing created will be saved in AutoCAD's 2007 DXF file format. The saved file will be transferred to the robot's controller using TCP/IP connection and saved under "USRAPP" directory of controller. This directory is actually where the user application codes are saved. After this procedure the CAD file can be used with user created applications.

In order to transfer files between robot's controller and the personal computers, a FTP (File Transfer Protocol) Software should be used. The STAUBLI itself recommends the program called "FTP SURFER". In our applications this program is used for file transfers. Note that deleting or changing the controller's system files is also possible with the FTP connection. Therefore file transfer process should be done carefully.

4.1.4 Extracting points from CAD file to transformation data

In our applications, the points that are created on desired path will be used as the motion's essential elements. Therefore our main aim will be to obtain the data which contains the information of points created.

The information about the points mentioned above is saved in ENTITIES section of the DXF file. The point information is defined using the group codes as it is mentioned on Section 4.1.1. The group codes which are applied to point entities are shown on Table 4.2.

Table 4.2 : DXF file point group codes.

Point Group Codes	
Code Number	Description of the Code
100	Subclass marker (AcDbPoint).
10	The X value of point location (position) in Cartesian Space.
20	The Y value of point location (position) in Cartesian Space.
30	The Z value of point location (position) in Cartesian Space.
39	Thickness (optional; default = 0).
210	X value of extrusion direction (optional; default = 0, 0, 1).
220	Y value of extrusion direction (optional; default = 0, 0, 1).
230	Z value of extrusion direction (optional; default = 0, 0, 1).
50	Angle of the X axis for the UCS (user defined coordinate system) in effect when the point is drawn (optional, default = 0).

In order to get point information out of the DXF codes the following steps will be applied.

Firstly, the DXF file, which will be inspected, is opened using the expansion pack instructions of VAL3. It will be checked if the file is corrupted or valid. Next, the entities section of the file will be searched in file. If entities section exists, the lines which contain points are searched in entities section. Points are defined under groups indicated with “POINT”. The point data groups are defined with the group code “100” following with the string “AcDbPoint” as mentioned in Table 7.2. The X coordinates of the point found is given under code number “10”, the Y coordinates of the point found is given under code number “20”, and the z coordinates of the point

found is given under code number “30”. The coordinate information found from this procedure is saved in transformation types of VAL3 as they will be used as motion indicators. The transformation variable is an array of n-elements where the number “n” represents the number of points defined in CAD file. Lastly, file is closed if end of file (EOF) is reached.

Using the information explained above, let us inspect the file which is created for 2D drawing explanations, “01triangle.dxf” which is given in Appendix A.1. After the file is opened using a text editor, the ENTITIES section can be found in lines between 3602 and 4438. This section contains the POINT groups which start with line 3898. It can be observed that the coordinate information is saved in these groups as explained above. 32 points which have been drawn can be inspected with the coordinates defined to follow the triangle.

In order to process the CAD data in robot’s controller, application named “CADTracking” is used. This application can also be found in Appendix A.1 and can be opened by Staubli Robotics Studio. Let us inspect the $void\ cadGetPoints(string\ sFileName, trsf\&\ trTrsf, num\&\ nError)$ program’s code lines step by step (Note that if program is being read using text editor, line numbers may differ from SRS lines.). This program is created in order to achieve data gathering process mentioned on this section. The first parameter defined as string is used for entering the name of the CAD file that will be used. The second parameter defined as transformation array is used for storing the CAD point data that will be gathered during process. The last parameter defined as num is used in order to return the error code if any. From lines 7 to 12 the CAD file that will be used is opened. An error code with the number -2 is returned if there exists no file. From lines 13 to 21 the CAD file is inspected as if it is corrupted or valid, if file is corrupted the error code with the number -3 is returned. From lines 22 to 26, end of file check is done. If there are no points defined in the error code with the number -4 is returned. From lines 27 to 31 the application is programmed to search for the ENTITIES section. In entities section if no points are found an error with the number -4 returned as programmed in lines from 32 to 35. From lines 36 to 77 a search in order to find the point data is processed until the end of the section is reached. On

this process the transformation array's size is increased depending on the number of points found. An error with the number -1 is returned if the file specified cannot be opened.

The program mentioned above is defined as public in application with two of its parameters passed by reference. This is done in order to allow reachability of this program from other applications if the "CADTracking" application will be used as a library.

The usage details of the program will be given on following sections with the application that will be explained.

4.1.5 Converting transformations to points

On this section, conversion process of transformation array, which is obtained from CAD file, to point array that will be used for motion is explained. The question why the transformation array is used but not the points obtained from CAD data at the first step might be asked at this point. The answer to this question is flexibility. Using transformations gives us the advantage of defining the motions in desired (in other words user created) frames. Otherwise the motions could only be achieved in world frame.

$void\ trsfToPoints(trsf\&\ trTrsf, pPoints, frame\ fFrame, point\ pRefPoint)$

Program which is located in application named "CADTracking" (Located in Appendix A.1), is used for the conversion process. The first parameter of the program is the transformation data array which is wanted to be converted. The second parameter is the point data array, in which the resulting point data will be stored. The third parameter is the reference frame in the Cartesian space, in which the motions will be done. And the last parameter is the reference point defined in reference frame which will be used in order to point the origin.

On next step, let us investigate the program line by line. The program starts with clearing the contents of previously used point array data with the lines from 3 to 6. From lines 7 to 11, the size of the point array is adjusted according to the transformation array size given as parameter. In the line 13 new point data is assigned using the compose instruction that is explained on Section 4.4.2. Note that

new size of the point data array will be equal to the number of points defined in CAD file.

After all the procedure explained above including creating the CAD file, importing the file to controller, extracting points from the file, and conversion of new points in reference frame, the user will be ready to achieve motions with the CAD data.

4.1.6 Moving the arm through gathered points

On this section how to generate the motion, which will be used for following the path defined in CAD file, will be explained using the point data array mentioned on previous chapters.

As mentioned before, the path will be followed by the combination of many small motions. Therefore, in order to achieve a smooth follow through the desired path blending option of VAL3 is mandatory. Besides, the number of points on the CAD data should be chosen carefully. Choosing the number of points insufficiently will end up with poor tracking results. Besides, using an exceeding number of points will end up with overloading the processor pointlessly or even cause the system to stop with an error turning the arm's power off.

The program $void\ moveLinPath(\ pArray, tTool, mdesc\ mSpeed, num\&\ nErr)$ which is located in application named "CADTracking" (Located in Appendix A.1) is used in order to generate motions. The first parameter of the program is the point data array which is obtained on previous sections. The second parameter is used to define the tool which will be used while motions are achieved. The third parameter is used to define the motion description of the motions. And the last parameter is used in order to return the point number in which an error occurred.

The program starts with recording the current position of the arm as joint variables in line 7, this information will be used in order to check if target points can be reached. The motion indicator is set to 0 in line 9, which will allow us to keep track of the motions easily. On line 12 the reachability check is done using the point to joint command which is explained on Section 3.4.4.2. On line 14 the motion through the points is achieved only if reachability check returned true. Motions are achieved with the linear move command which is explained on Section 3.4.5.2. If the motion is not

possible to a point, the number of this point in array is returned as an error with the last parameter of the program. If the process is successful with no errors, 0 is returned.

On the following sections, the applications of the programs, studied on this section, will be explained and inspected in detail.

4.2 Force Control

4.2.1 Introduction

The capability to handle the physical contact between a robot and the environment is a fundamental requirement for success of a manipulation task. Because of unavoidable modeling errors and uncertainties which arise during the contact, pure motion control turns out to be inadequate, leading to an unstable behavior during the interaction. This problem occurs especially in the presence of rigid environments [1]. Assume that a manipulator is used to wash a window with a sponge. The compliance of the sponge will make it possible to regulate the force applied to the window by just controlling the position of end-effector relative to glass. However, in many applications the stiffness of the end-effector, tool, or environment is high. Therefore, it becomes increasingly difficult to perform operations in which the manipulator presses against the surface. Now assume that instead of the washing with sponge, the manipulator will be used for scrapping paint off a glass surface, using a rigid scrapping tool. If there is any uncertainty in the position of the glass surface or any error in the position of the manipulator, this task would become impossible. Either the glass would be broken, or the manipulator would move the scrapping tool over the glass with no contact taking place. In tasks like the examples given above, it would be more reasonable to specify a force that is to be maintained normal to the surface, rather than specifying the position of the plane to be tracked [20].

This section will start from the basic force control strategies, including stiffness control, which is essentially based on mass spring system. Then, the problem of interaction tasks will be analyzed with the Hybrid (Force/Motion) Control. And lastly, the control strategy which will be used in applications will be explained.

4.2.2 Stiffness control

While considering forces of contact, some model of the environment upon which is being acted must be chosen. For the purposes of conceptual development, a simple model should be used for ease of use. On this section the contact environment will be modeled as a spring. It will be assumed that there is stiffness between the probe of robot and the environment [20].

Assume that a simple mass spring system shown on Figure 4.4 is being studied [20].

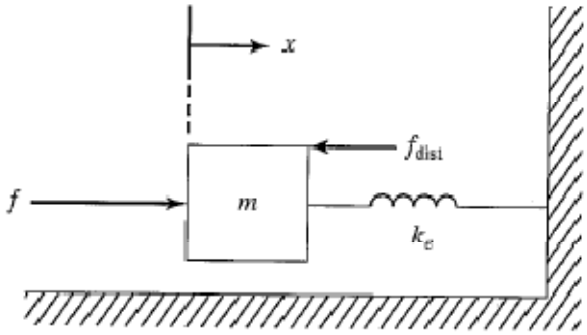

Figure 4.4 : Mass spring system.

The variable that is wanted to be controlled is the force acting on the environment f_e, which is also the force acting on the spring, given on Equation 4.1 [20].

$$f_e = k_e x \tag{4.1}$$

The equation describing the physical system is given with the Equation 4.2 [20].

$$f = m\ddot{x} + k_e x + f_{dist} \tag{4.2}$$

These equations can be rewritten as on Equation 4.3.

$$f = \frac{m}{k_e}\ddot{f}_e + f_e + f_{dist} \tag{4.3}$$

In these equation f_{dist} is the disturbance force acting on our system.

In order to control the system defined above, a PD controller can be used [20]. The control law given on Equation 4.4 will be used as an example.

$$f = \frac{m}{k_e}\left[\ddot{f}_d + k_{vf}\dot{e}_f + k_{pf}e_f\right] + f_e + f_{dist} \tag{4.4}$$

In Equation 4.4 the variable $\dot{e_f}$ is the difference between the desired force and actual environment force defined with $e_f = f_d - f_e$. Assuming that that the environment is stiff enough, the disturbance can be taken as $e_f = f_e$. Therefore our control law will take the form given on Equation 4.5 [20].

$$f = \frac{m}{k_e}\left[\ddot{f_d} + k_{vf}\dot{e_f} + k_{pf}e_f\right] + f_d \qquad \textbf{(4.5)}$$

The block diagram of the force control system explained above is given on Figure 4.5 [20].

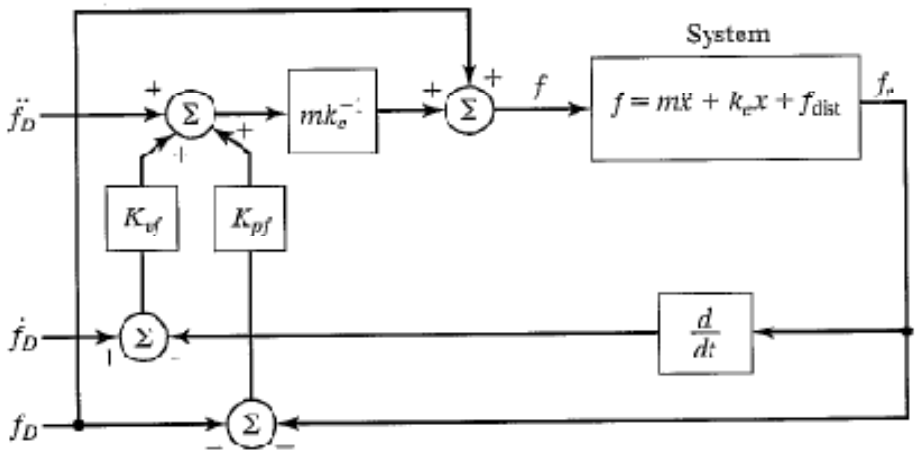

Figure 4.5 : Block diagram of mass spring control system.

An additional integral term can be included in controller in order to improve steady state error.

An important problem is that the term k_e which indicates the stiffness of the environment appears in our control law. The difficulty that will be come across is, this parameter is generally unknown and changes from time to time. However, often an assembly robot is dealing with rigid parts, and k_e can be guessed to be quite high. Generally the proportional and derivative factors are chosen so that the system is robust with respect to variations in k_e [20].

The stiffness control suggests that, the position gains in a joint-based servo system are modified in such a way that the end effector appears to have certain stiffness along Cartesian degrees of freedom [20]. The relationship between forces on end

effector and the Cartesian position of the end effector can be given with the Equation 4.6 [2].

$$\boldsymbol{\gamma = K\, dx} \tag{4.6}$$

On this equation, let γ, with a dimension of (6x1), be a vector indicating the forces on the end-effector being used. K is a (6x6) matrix which defines the stiffness. It is symmetric and positive semi-definite and consists of the elements given with the Equation 4.7. The variable given with x is the elementary displacement of end-effector [2].

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K_f} & 0 \\ 0 & \boldsymbol{K_\mu} \end{bmatrix} \tag{4.7}$$

In Equation 4.7, $\boldsymbol{K_f}$ is a (3x3) matrix and defined as the translational stiffness matrix. $\boldsymbol{K_\mu}$ Is a (3x3) matrix and defined as the rotational stiffness matrix. The zeros on opposite diagonal are there because of the fact that coupling stiffness is assumed to be zero [2].

4.2.3 The hybrid (position/force) control

In this section, an architecture design for a control system that implements the hybrid position/force controller will be introduced.

As a the first step, a simple case of a manipulator having three degrees of freedom with prismatic joints will be considered. The arm imagined is given on Figure 4.6. The end-effector of the manipulator is in contact with a surface with the stiffness k_e that is oriented with its normal in the $-Y$ direction. Therefore, force control is required in the Y direction while position control is needed in X and Z directions [20].

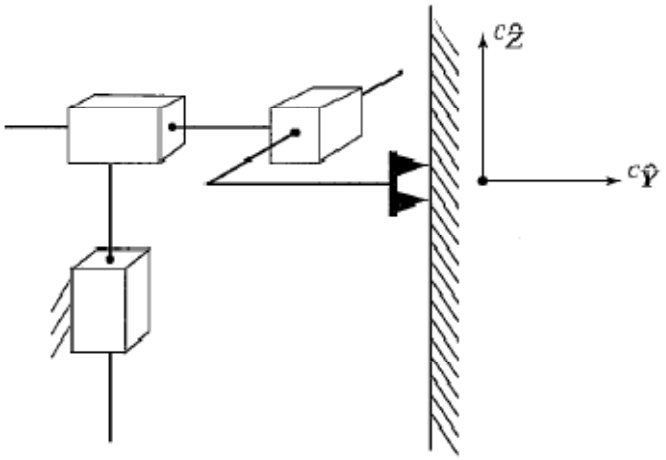

Figure 4.6 : Simplified manipulator with 3-DOF.

In this case a hybrid (position/force) controller is needed in order to solve the problem [20]. It is desired that joints 1 and 3 will be controlled with position controller while the joint 2 will be controlled with the force controller.

The Cartesian arm control system which will be designed can be generalized as follows: The structure of the controller will be built in a way that a complete position trajectory in all three degrees of freedom and also a force trajectory in all three degrees of freedom will be represented. Of course, the control with these six constraints cannot be achieved all together at any given time. Instead of using them all together, a control design which is shown on Figure 4.7 will be used [20].

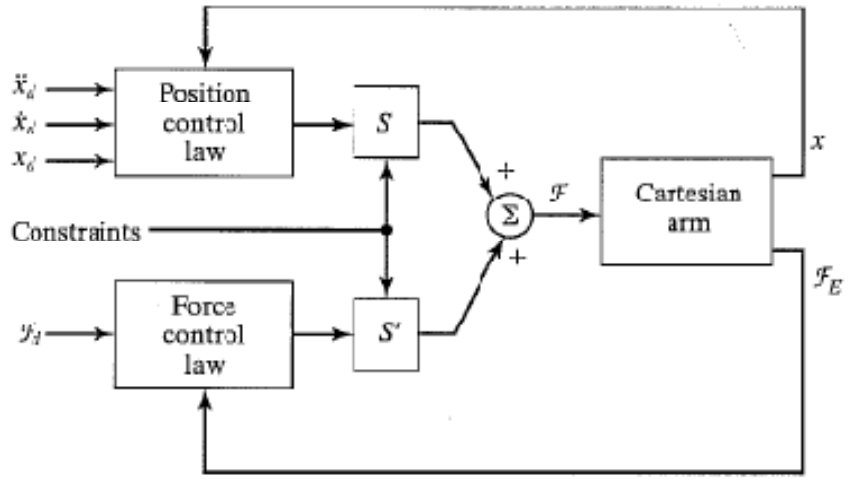

Figure 4.7 : Simplified block diagram of hybrid control.

Here, the control of all three joints of our simple Cartesian arm is indicated in a single diagram by showing both the position controller and the force controller. The matrices S and S' are introduced to control which mode, position or force, is used in order to control each joint of the Cartesian arm. The S matrix is diagonal, consisted with ones and zeros on the diagonal. Where a one is present in S, a zero is present in S' and position control is in effect. Where a zero is present in S, a one present in S' and force control is in effect. Therefore, the matrices S and S' simply switches that set the mode of control to be used with each degree of freedom in manipulator [20].

Considering the example manipulator given on Figure 4.6, the S and S' matrices can be given according to the Equation 4.8.

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad and \quad S' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

The generalized control scheme which combines kinematics, dynamics and hybrid control is drawn for a Cartesian based general manipulator on Figure 4.8 [20].

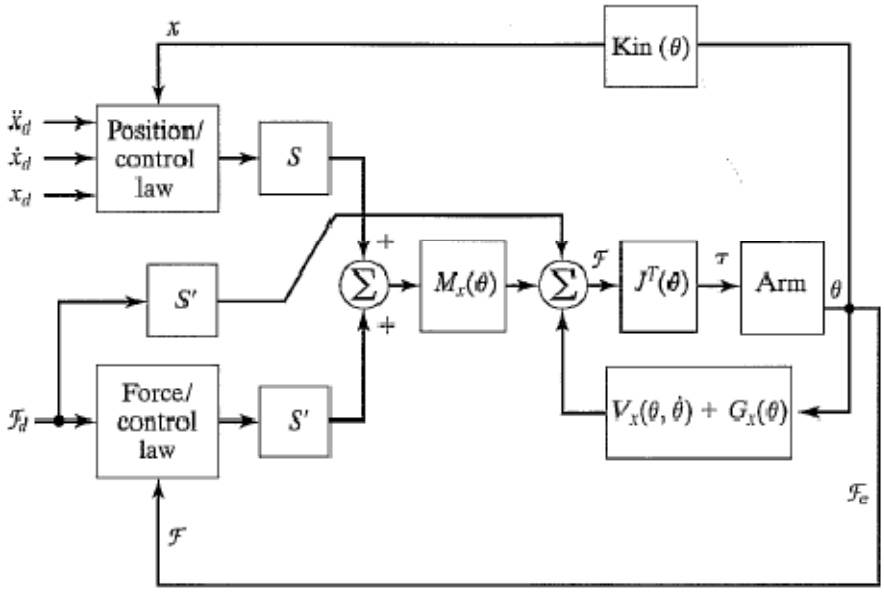

Figure 4.8 : Block diagram of hybrid control.

In Figure 4.8 the term $M_x(\theta)$ is a (6x6) matrix which is used in order to define the mass matrix of the arm. The term $V_x(\theta, \dot{\theta})$ is a (6x1) vector which is used in order to

define the centrifugal and Coriolis terms. And lastly the term $G_x(\theta)$ is a (6x1) vector which is used in order to define the gravity terms [20].

4.2.4 Impedance control

Impedance controller is used in order to impose, along each direction of the task space, a dynamic relation between the manipulator end-effector position error and the force of interaction with the environment, the desired impedance [7]. Note that it is designed to maintain a desired position, velocity, and acceleration with the help of force data. It cannot be used in order to maintain a desired force, therefore will not be used in our application but will only be explained here briefly. Usually the desired impedance is chosen linear and of second order, as in a mass-spring-damper system [7].

The user chooses the desired end-effector impedance which may be expressed by Equation 4.9, in order to fulfill the task requirements [7].

$$M_d(\ddot{x} - \ddot{x}_d) + B_d(\dot{x} - \dot{x}_d) + K_d(x - x_d) = f_e \qquad \textbf{(4.9)}$$

In this equation M_d, B_d and K_d are constant, diagonal, and positive definite matrices, representing the desired inertia, damping, and stiffness system matrices. The symbols x and x_d represents the actual and desired end-effector positions. And lastly, f_e is the generalized force the environment exerts upon the end-effector.

The block diagram of an example impedance controlled system is given with Figure 4.9 [7].

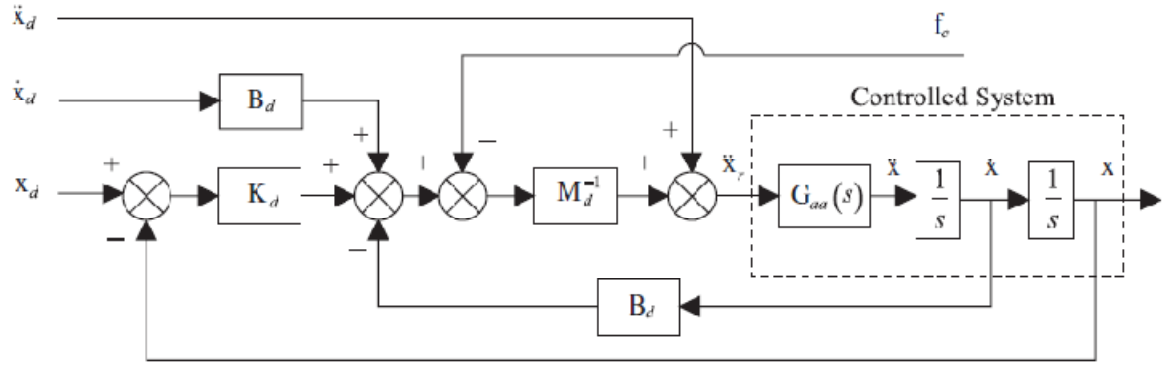

Figure 4.9 : Block diagram of impedance control.

4.2.5 Force control algorithm and discretization

In this section, the control architecture which is used in order to control our robotic arm RX-160 will be explained. Because of the reason that the robotic arm's technical data, which includes the dynamical parameters, does not exist, constructing a full model of the system is not possible. As it is not possible to measure parameters like robot part's inertias, the only way to obtain dynamical parameters would be to use an identification method. But still, even if the robot's parameters are obtained, the parameters that will occur during the contact between robot and the environment would be hard to predict.

Because of the reasons listed above, computing the parameters of robot and the environment in a detailed way is redundant, actually almost impossible. Instead of finding these parameters, the system is represented with simpler models which are more familiar, easier to implement and control.

While controlling the system, the model of the contact between the environment and the robotic probe will be represented with the model of contact between an object and a spring. Therefore, Equation 8.6 can be used in order to obtain relationship between position and the force. In other words, a small deviation of force will occur at the end-effector when end-effector position is changed with a small deviation. And the relationship between these two deviations will be a constant multiplier. Note that this constant multiplier can be changed depending on the environment in which the arm will be used. The relationship is also represented in Figure 4.10.

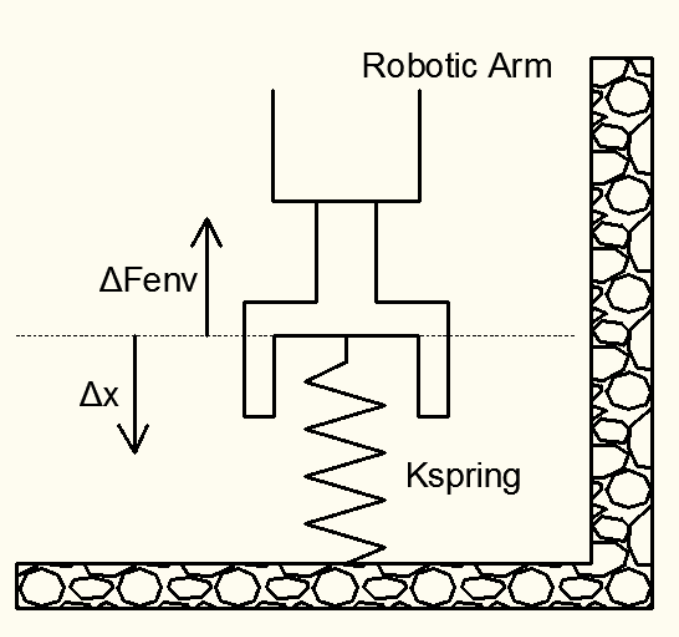

Figure 4.10 : Relationship between force and displacement.

In the Figure 4.10, the equation of the relation between force and displacement can simply be written as given below in Equation 4.10.

$$\Delta F_{env} = \Delta x \, . \, K_{spring} \qquad \textbf{(4.10)}$$

At this point, as the model which is used in order to achieve conversion between force and displacement is explained, the controller algorithm which will be used can be explained. Because of the reason that it is wanted to design and control our system in the simplest way which can fulfill desired force control application, PID controller will be used.

4.2.5.1 PID controller

PID is one of the most common controllers, widely used in industry. A PID controller simply calculates an error value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process control inputs. The PID controller algorithm (calculation) involves three separate parameters, and is accordingly sometimes called three term control which are the proportional, the integral and derivative values, denoted with the symbols P, I, and D. Briefly, it can be said that proportional term depends on the present error, integral term depends on the past errors, and the derivative term

depends on prediction of future errors [22]. Although PID controller doesn't guarantee optimal control of the system or the system stability for every configuration, it is the best option to use in the absence of exact knowledge of the underlying process at first shot [23].

The generalized control algorithm that will be used is represented by the block diagram given in Figure 4.11.

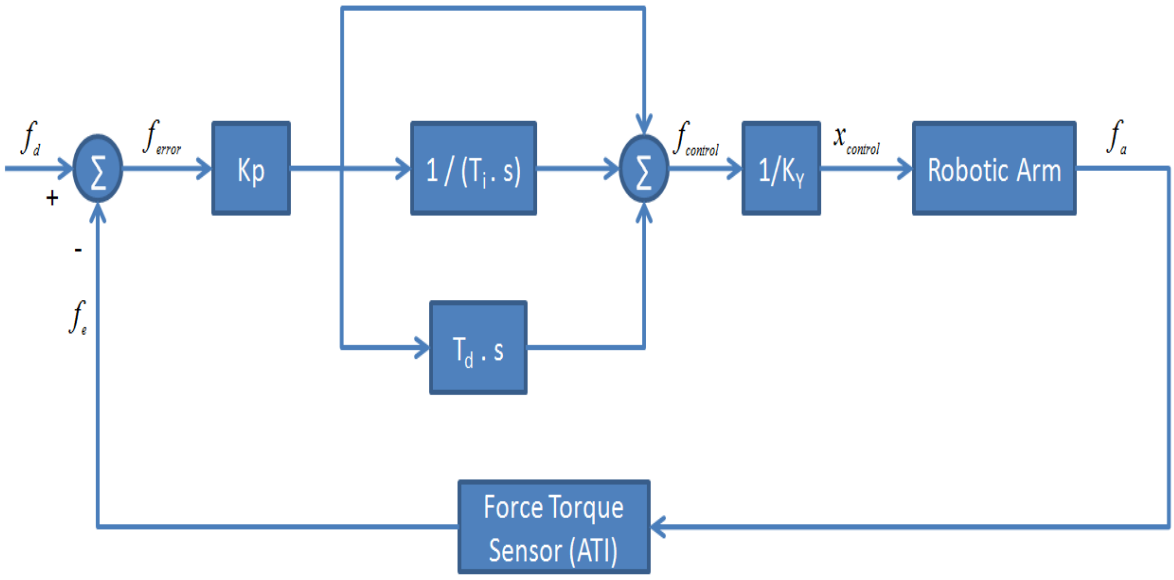

Figure 4.11 : Block diagram of PID force control algorithm.

In Figure 4.11, the elements K_p, T_i and T_d represent the parameters of the PID controller. The spring constant which will be used for conversion between force and position deviations is given by the parameter K_Y. The actual force, which occurs on the end-effector, is shown with f_a. The actual force is measured by the Force Torque sensor, converted to the values that can be used by controller, and fed back to our loop with the parameter name f_e.

As controller will be working on discrete time, the control algorithm will be obtained on s-domain first. Then, it will be transferred to z-domain. And lastly, it will be translated to difference equations.

The transfer function of the PID controller used is given on Equation 4.11.

$$G_{pid}(s) = K_p\ (1 + \frac{1}{T_i s} + T_d s) \quad \textbf{(4.11)}$$

The z-Transform of the PID controller will be found with the help of a zero order holder which is described on Equation 4.12. The parameter T given on Equation 4.14 will be used as the sampling time of our system.

$$G_{pid}(z) = \mathcal{Z}\left\{\frac{1-e^{-Ts}}{s} K_p\ (1 + \frac{1}{T_i s} + T_d s)\right\} \quad \textbf{(4.12)}$$

$$G_{pid}(z) = (1 - z^{-1})\ \mathcal{Z}\left\{\frac{1}{s} K_p\ (1 + \frac{1}{T_i s} + T_d s)\right\} \quad \textbf{(4.13)}$$

$$G_{pid}(z) = K_p + \frac{K_i}{1-z^{-1}} + K_d\ (1 - z^{-1}) \quad \textbf{(4.14)}$$

The Equation 4.14 shows the transfer function in z-domain. The parameters in that equation can be expanded as $K_i = K_p \frac{T}{T_i}$ and $K_d = K_p \frac{T_d}{T}$.

This equation can be written in difference equation with the following calculations.

$$\frac{f_{control}(z)}{f_{error}(z)} = \frac{U(z)}{E(z)} = K_p + \frac{K_i}{1-z^{-1}} + K_d\ (1 - z^{-1}) \quad \textbf{(4.15)}$$

$$U(z)(1 - z^{-1}) = E(z)\big(\ K_p\ (1 - z^{-1}) + K_i + K_d\ (1 - 2z^{-1} + z^{-2})\ \big) \quad \textbf{(4.16)}$$

$$u[k] = u[k-1] + K_p(e[k] - e[k-1]) + K_i e[k] + K_d(e[k] - 2e[k-1] + e[k-2]) \quad \textbf{(4.17)}$$

The Equation 4.17 can be written in a more compact form which is given on Equation 4.18.

$$u[k] = u[k-1] + e[k]\big(K_p + K_i + K_d\big) - e[k-1]\big(K_p + 2K_d\big) + e[k-2](K_d) \quad \textbf{(4.18)}$$

The Equation 5.18 will be used as the control algorithm of our applications in VAL3. The arrays shown with [k] represent the values in current cycle time, while the arrays shown with [k-1] and [k-2] represents the values in previous cycles.

Our controller will have the general characteristics of a PID which are:

- Increasing the proportional effect will make the system give fast responses, however this effect may induce oscillations.

- The integral term will eliminate the steady state error but will introduce low frequency, which causes oscillations.
- The derivative term will help to reduce oscillations and increase the stability, however high values of it may cause the system to go saturation.

The PID program which is written in VAL3 language for PID control can be found in Appendix A.1 saved under directory "PID Control" with the filename "PID.txt". The program created includes three input parameters, and can be used with the instruction $void\ PID(num\ nRef, num\ nY, num\ axis)$.

The first parameter $nRef$ is used to define the desired value f_d in our case. The second parameter nY is used to define the actual value f_a in our case. The last parameter is used in order to define the axis being worked on. It can take values from 0 to 5, which in order defines X, Y, Z, RX, RY, RZ. For example if it is wanted to control arm with force control on X axis, 0 must be chosen for this parameter.

The program simply calculates the difference equation it is found on Equation 5.18 in the line 30. In order to avoid robot to move in dangerous speeds a maximum and a minimum output value is defined which limits the arm's motion with $\frac{0.05\ mm}{0.004\ sec} = 12{,}5\ mm/sec$. This value is experienced to be the most appropriate value for our force control applications.

In order to calculate PID controller coefficients ultimate oscillations method is used. This method is a closed loop method and can be applied to our system as output oscillations are possible for our system.

At the first step the PID controller is reduced to a proportional controller configuration by setting following constants.

$\tau_i = \tau_{imax}$ Therefore $\frac{1}{\tau_i} = 0$

$\tau_d = 0$

The block diagram of the system setup used for Ziegler Nichols Ultimate Oscillations Method is shown on Figure 4.12.

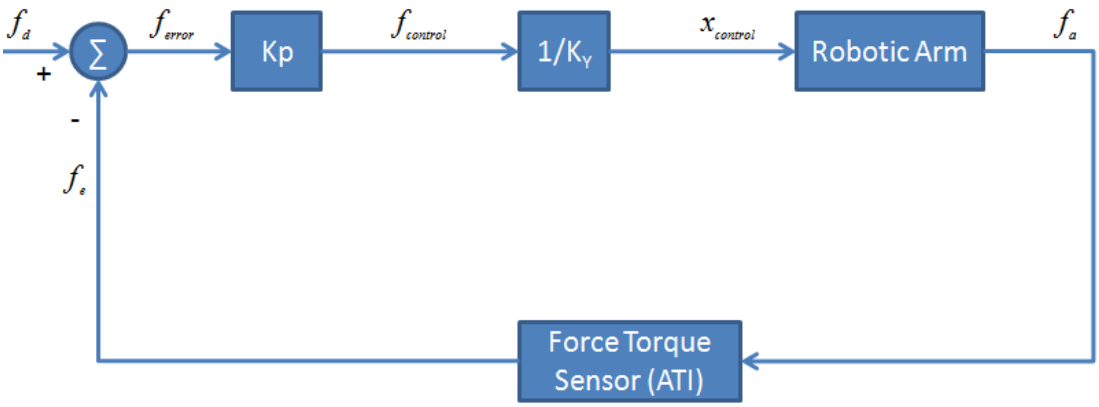

Figure 4.12 : Ziegler Nichols method implementation diagram

On next step, the proportional constant is increased until constant amplitude oscillations occur at the step response (for a reference of 20N step input) output.

For $K_{pu} = 0.0002$ response of the system is shown on Figure 4.13.

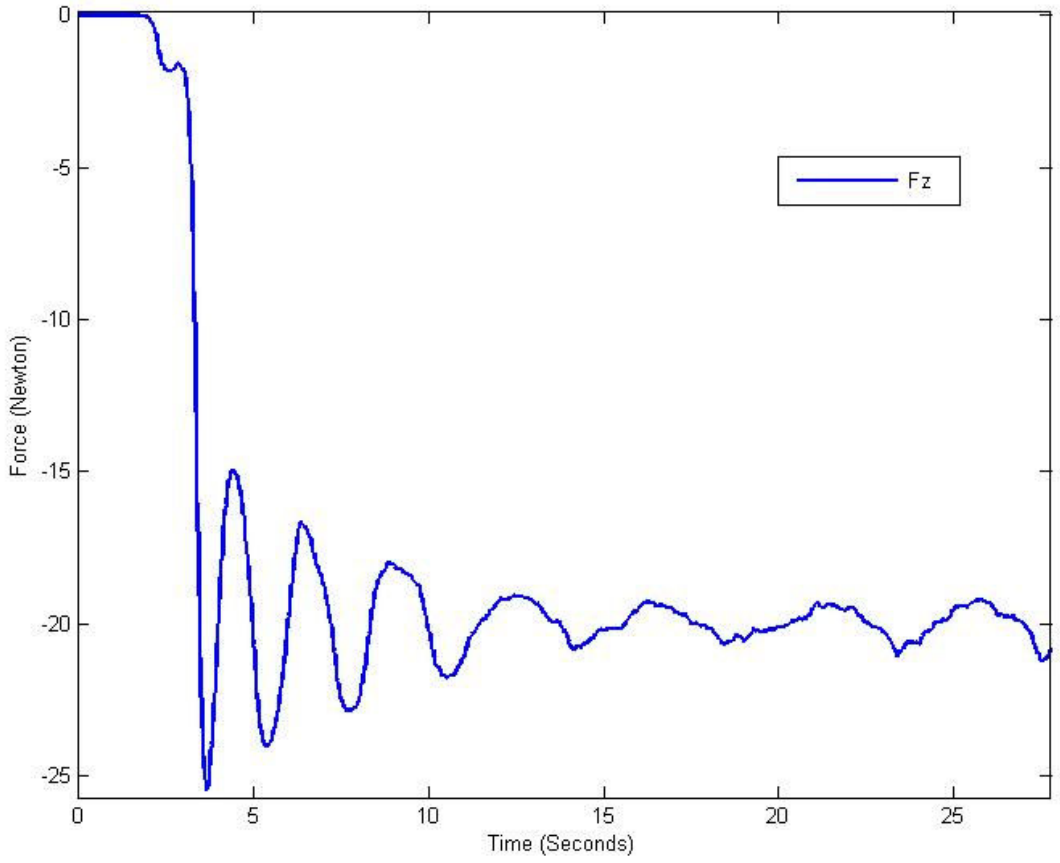

Figure 4.13 : Ziegler Nichols experiment 1

For $K_{pu} = 0.00025$ response of the system is shown on Figure 4.14.

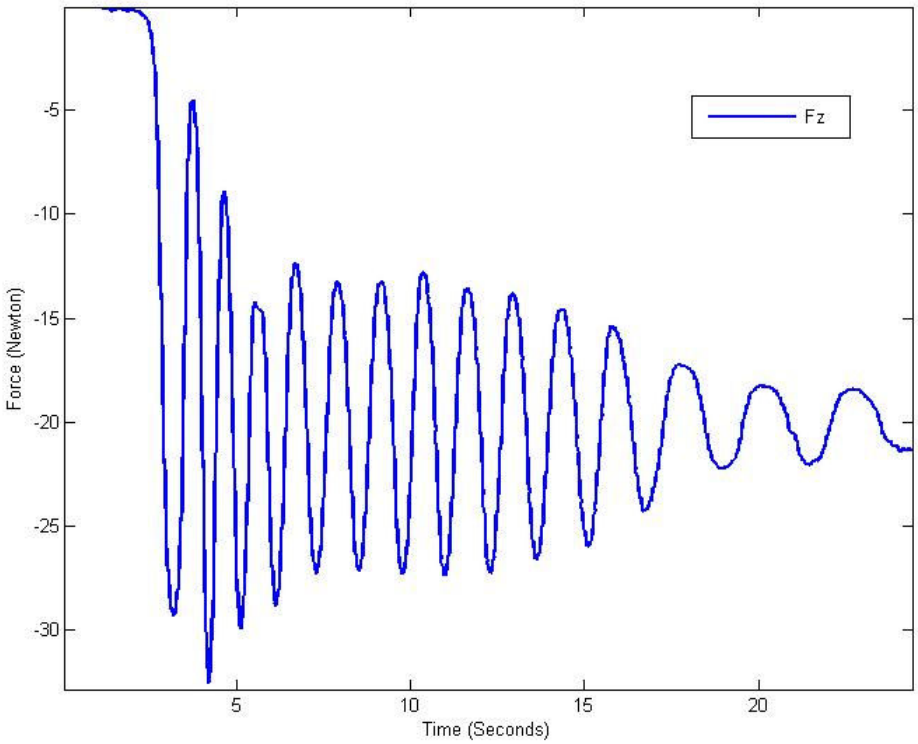

Figure 4.14 : Ziegler Nichols experiment 2

For $K_{pu} = 0.00028$ response of the system is shown on Figure 4.15.

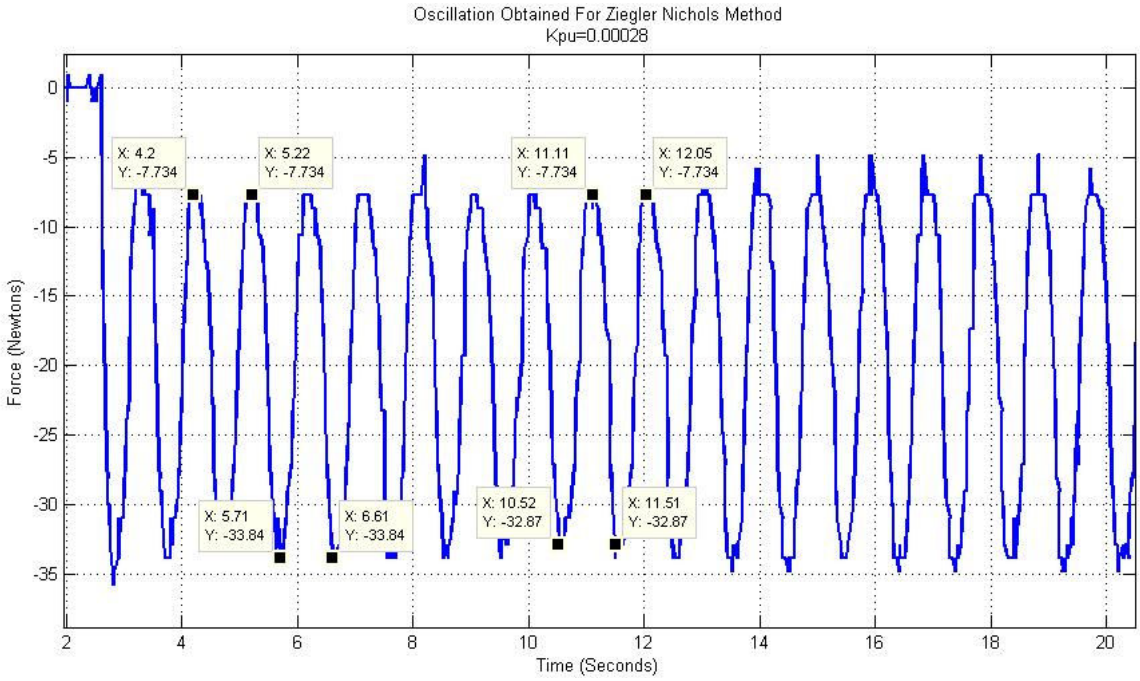

Figure 4.15 : Ziegler Nichols experiment 3

The period of the oscillations, $P_u = 1$, is measured and the gain used for this reason is obtained which is $K_{pu} = 0.00028$.

The coefficients of the controller according to Ziegler-Nichols are as indicated below.

$K_p = 0{,}6\ K_{pu} = \ 1{,}6 \times 10^{-4}$

$\tau_i = \frac{P_u}{2} = \frac{1}{2}$

$\tau_d = \frac{P_u}{8} = \frac{1}{2}$

Using these coefficients the PID controller performance of system output is given below on Figure 4.16 for a step reference input of 20N.

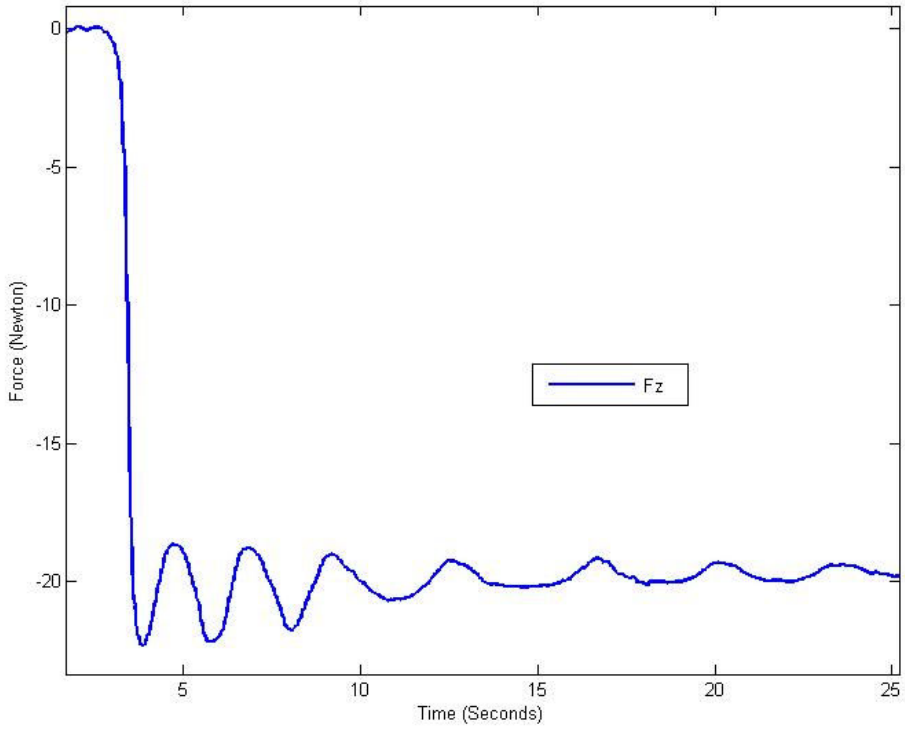

Figure 4.16 : PID response for Ziegler Nichols coefficients

By fine tuning coefficients (increasing Ti for decreasing oscillations and increasing Td for decreasing overshoot) the following response is obtained which is shown on Figure 4.17.

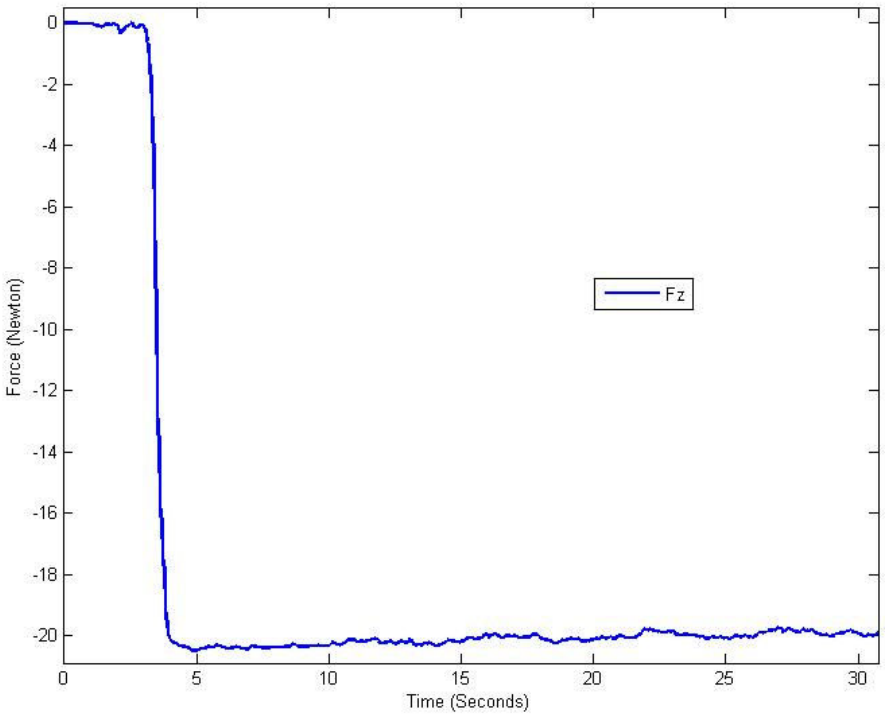

Figure 4.17 : Response for fine tuned PID parameters.

4.2.5.2 Sliding mode controller

In this section sliding mode controller which is used for force control will be explained. The algorithm which is used in order to produce control signal is given on Equation 4.19.

$$f_{control} = -|f_{error}|\, sgn\left(f_{error} + k_{smc}\, \dot{f}_{error}\right) \quad \textbf{(4.19)}$$

The block diagram with the use of sliding mode controller is given on Figure 4.18.

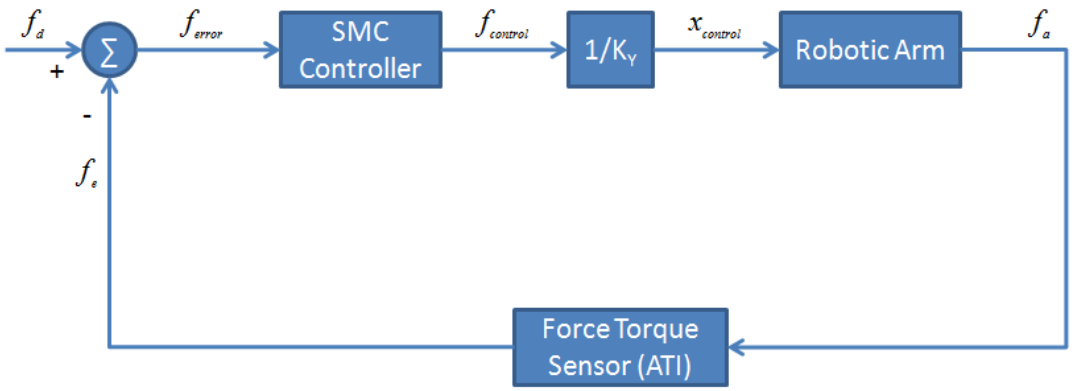

Figure 4.18 : Block diagram of SMC force control algorithm.

The difference equation of the used for providing control signal is given on Equation 4.20.

$$\left(f_{error}(k) + k_{smc}\frac{f_{error}(k)-f_{error}(k-1)}{T_{sampling}}\right) > 0 \Rightarrow f_{control}(k) = -f_{error}(k) \quad \textbf{(4.20)}$$

$$\left(f_{error}(k) + k_{smc}\frac{f_{error}(k)-f_{error}(k-1)}{T_{sampling}}\right) < 0 \Rightarrow f_{control}(k) = f_{error}(k)$$

The sliding mode coefficients are selected experimentally. Increasing the coefficient k_{smc} provided a better system respond. On the other hand, increasing this coefficient too much requires system to switch between $-f_{error}(k)$ and $f_{error}(k)$ too fast which can cause redundant motions.

The Figures from 4.19 to 4.23 shows the system's respond for a step reference of 30N. The value of k_{smc} is increased until desired response is obtained. The velocity which causes friction is 1.25 mm/sec.

Experimental results for $k_{smc} = 0{,}01$ are given on Figure 4.19

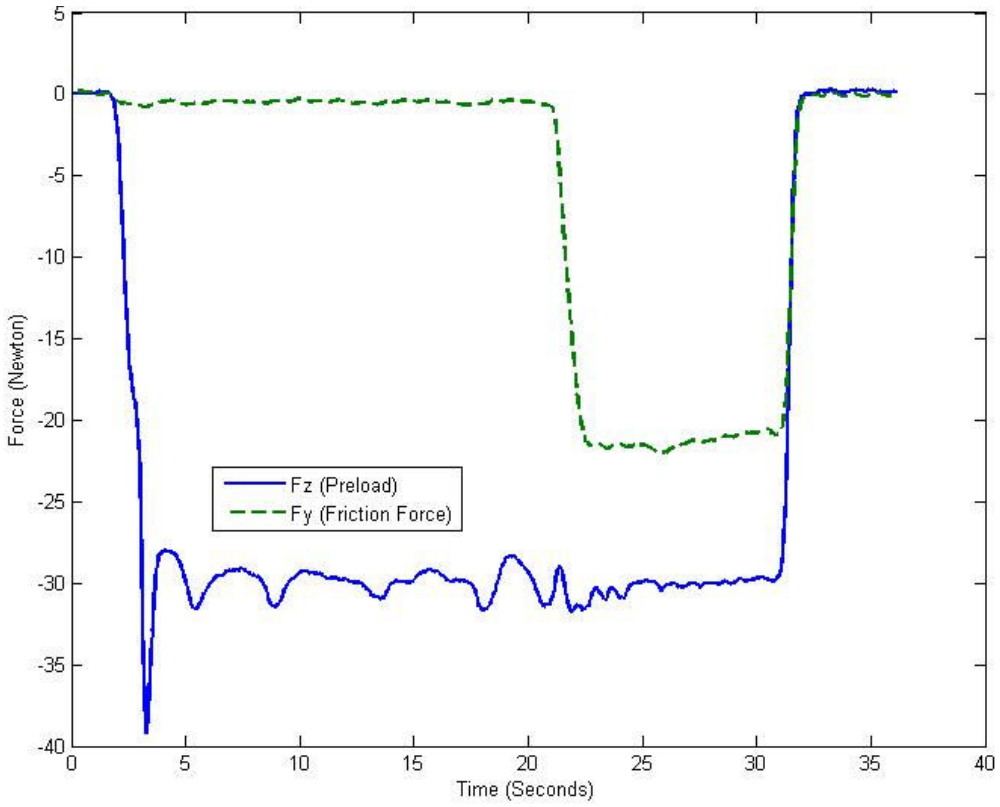

Figure 4.19 : Tune results for $k_{smc} = 0{,}01$.

Experimental results for $k_{smc} = 0{,}02$ are given on Figure 4.20.

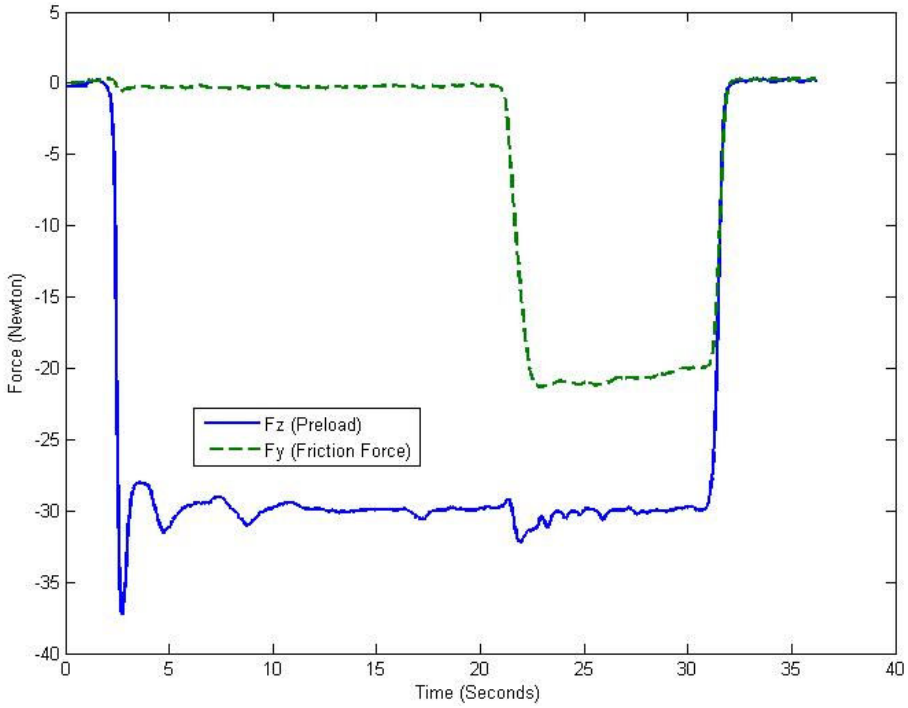

Figure 4.20 : Tune results for $k_{smc} = 0{,}02$.

Experimental results for $k_{smc} = 0{,}05$ are given on Figure 4.21.

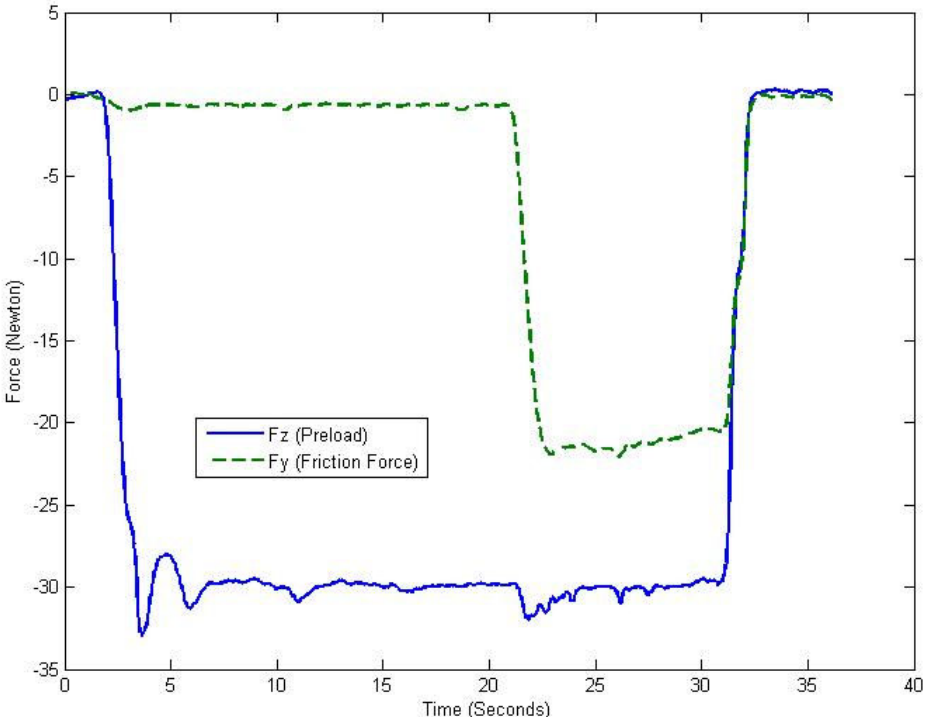

Figure 4.21 : Tune results for $k_{smc} = 0{,}05$.

Experimental results for $k_{smc} = 0{,}1$ are given on Figure 4.22.

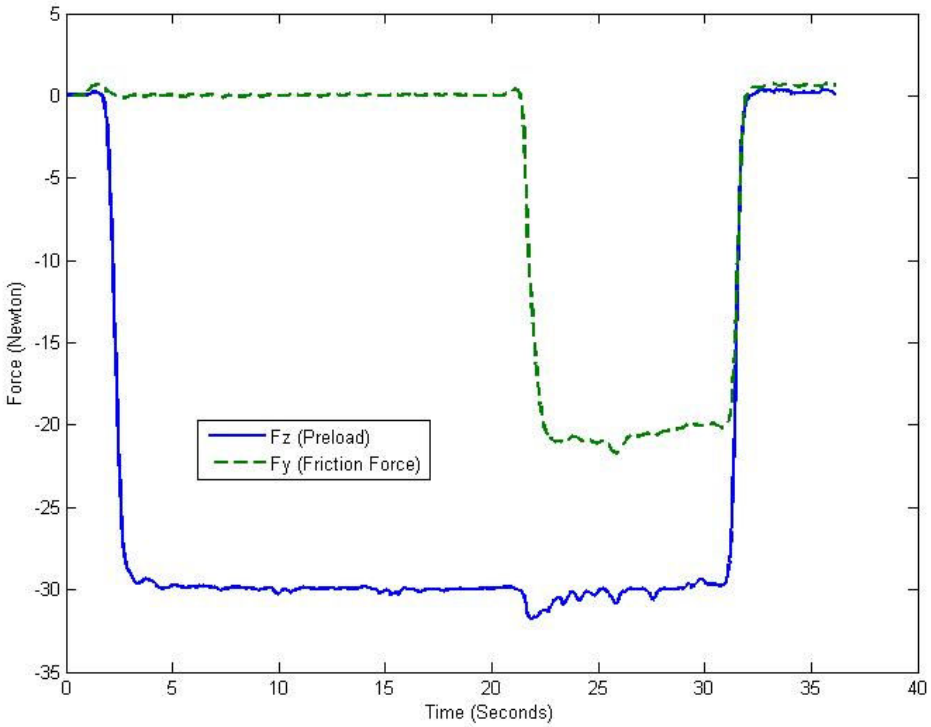

Figure 4.22 : Tune results for $k_{smc} = 0{,}1$.

Experimental results for $k_{smc} = 0{,}2$ are given on Figure 4.23

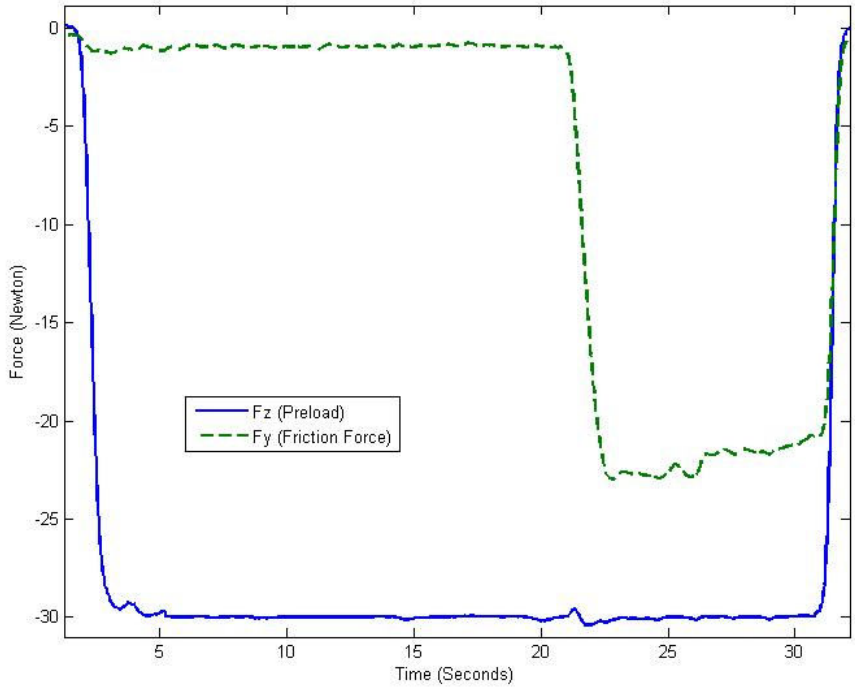

Figure 4.23 : Tune results for $k_{smc} = 0{,}2$.

It is obvious that the results for $k_{smc} = 0{,}2$ is the most desired response for Sliding Mode Controller.

4.2.5.3 PID and Sliding mode controller comparison

In this section the experimental results for PID and Sliding Mode controllers are compared on same experimental friction settings. The preload force is increased step by step and same measurements are taken for PID and SMC. The velocity which causes friction is 1.25 mm/sec for all experiments.

Measurements taken for 10N preload using SMC and PID are given on Figure 4.24.

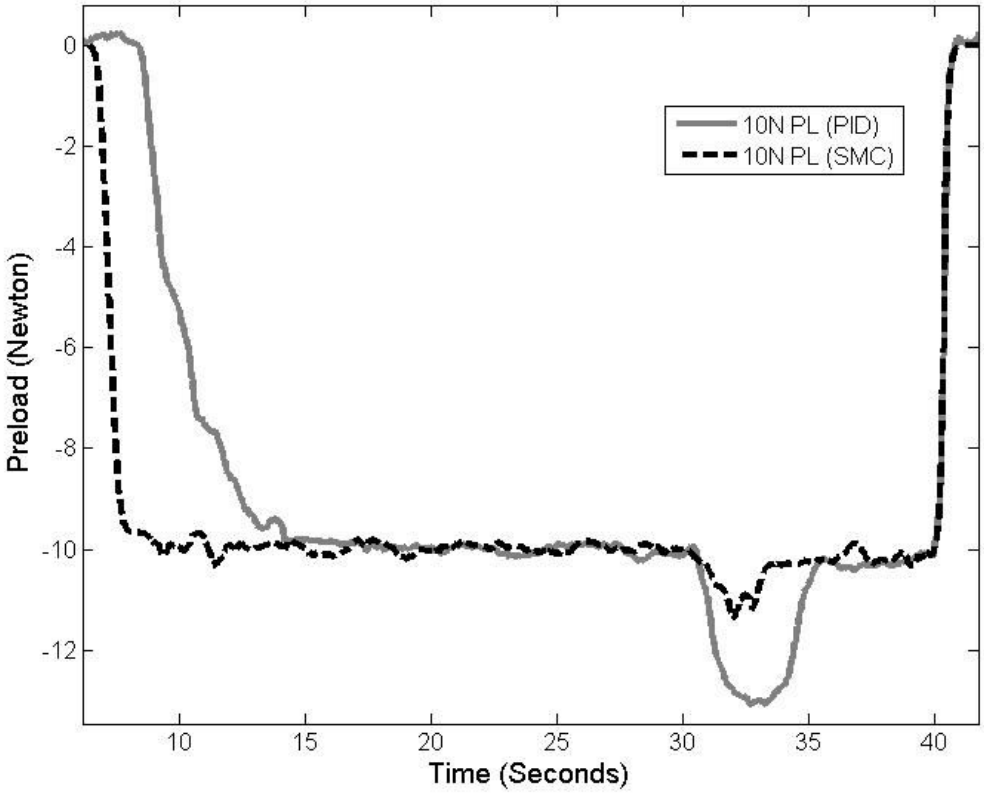

Figure 4.24 : Comparison results for 10N preload.

Besides, friction forces observed are shown in Figure 4.25.

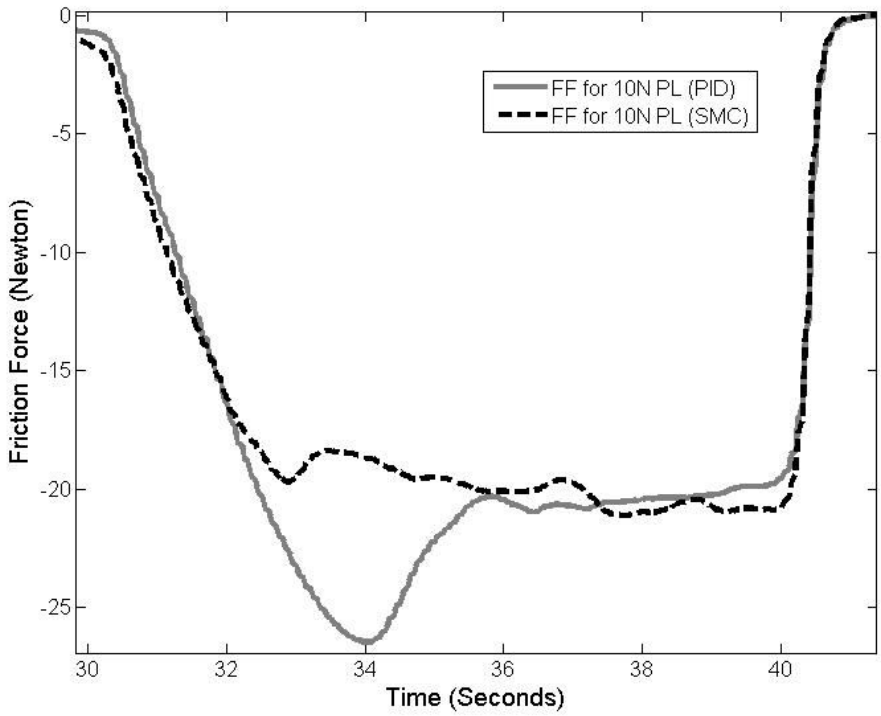

Figure 4.25 : Friction forces for 10N preload.

Measurements taken for 12N preload using SMC and PID are given on Figure 4.26.

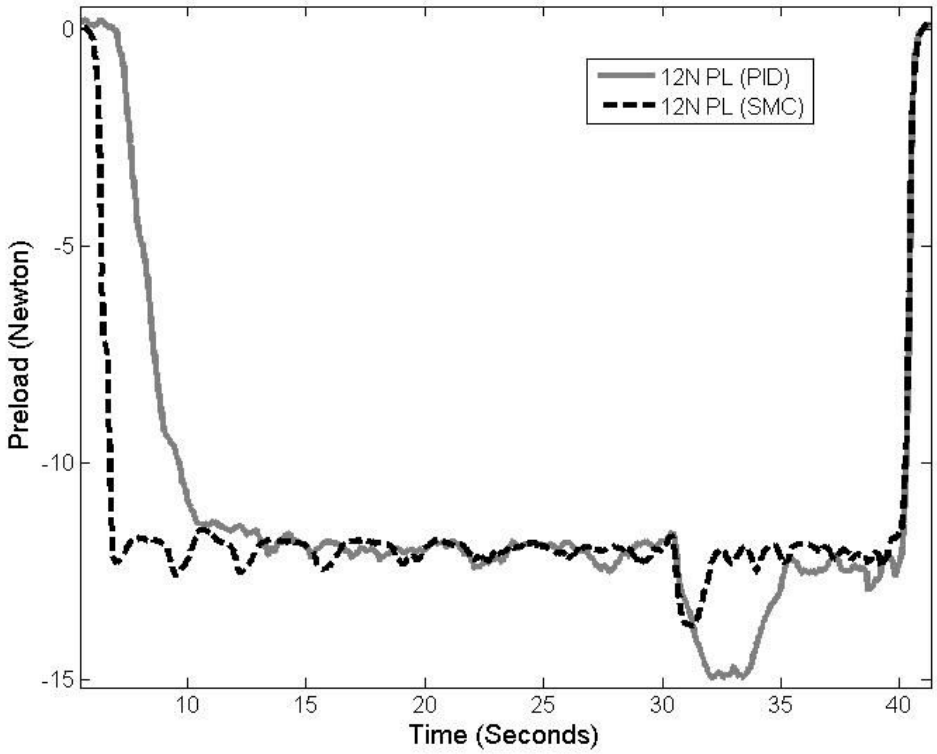

Figure 4.26 : Comparison results for 12N preload.

Besides, friction forces observed are shown in Figure 4.27.

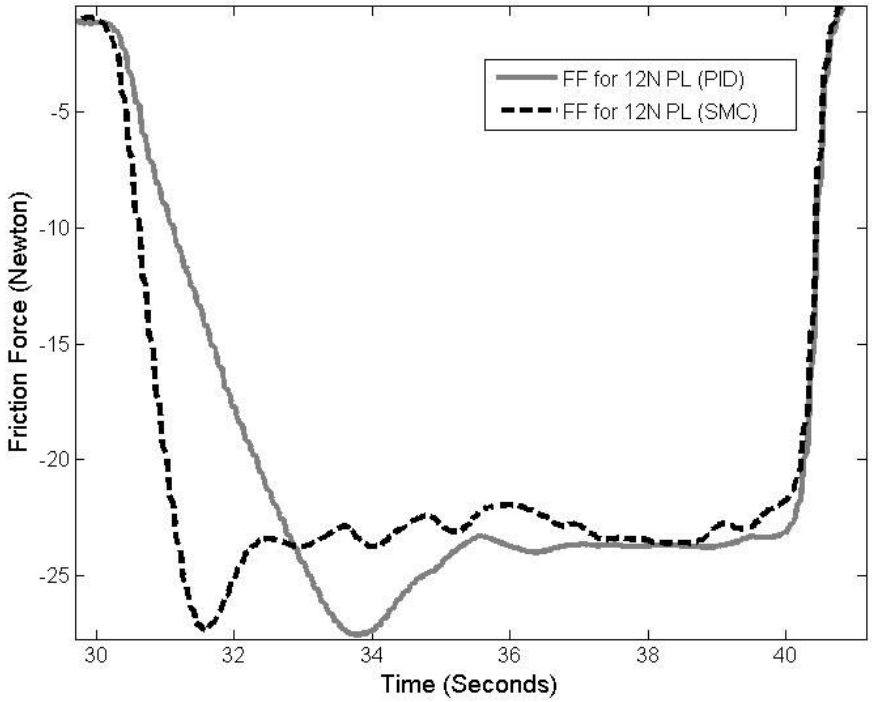

Figure 4.27 : Friction forces for 12N preload.

Measurements taken for 14N preload using SMC and PID are given on Figure 4.28.

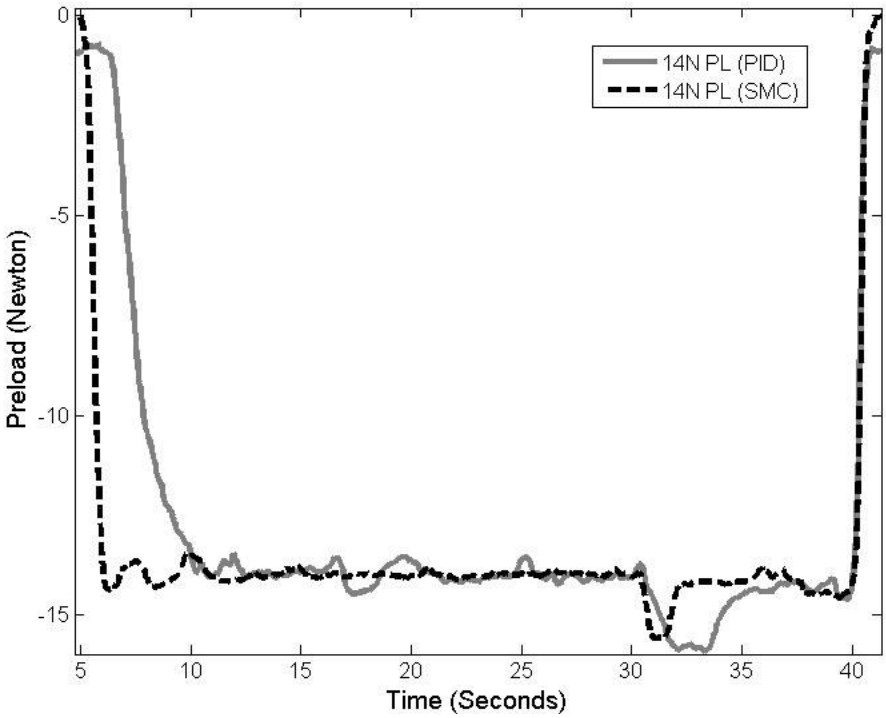

Figure 4.28 : Comparison results for 14N preload.

Besides, friction forces observed are shown in Figure 4.29.

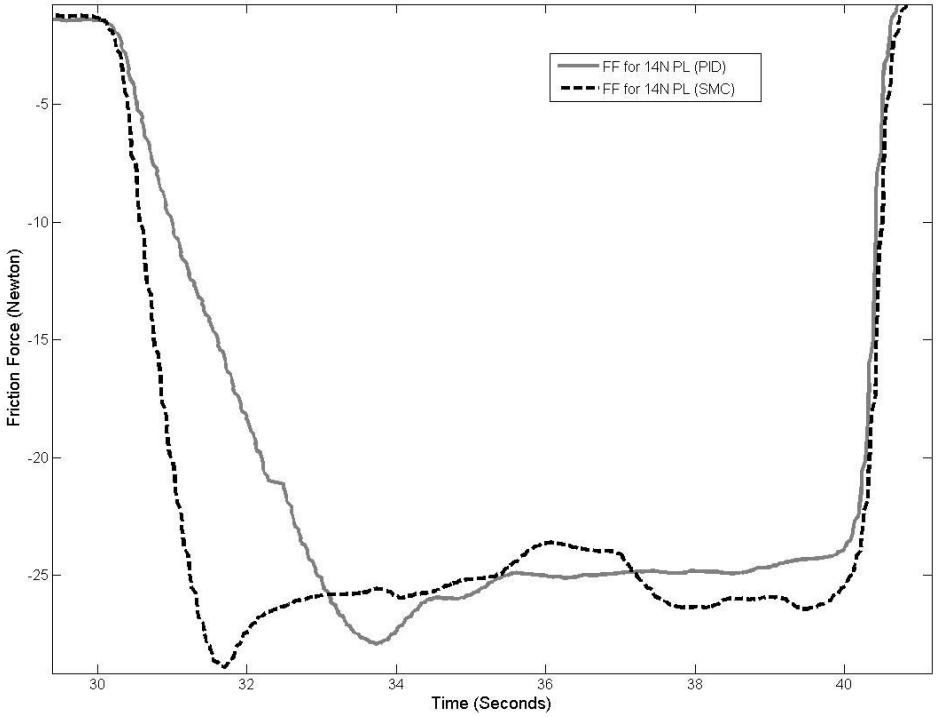

Figure 4.29 : Friction forces for 14N preload.

Measurements taken for 16N preload using SMC and PID are given on Figure 4.30.

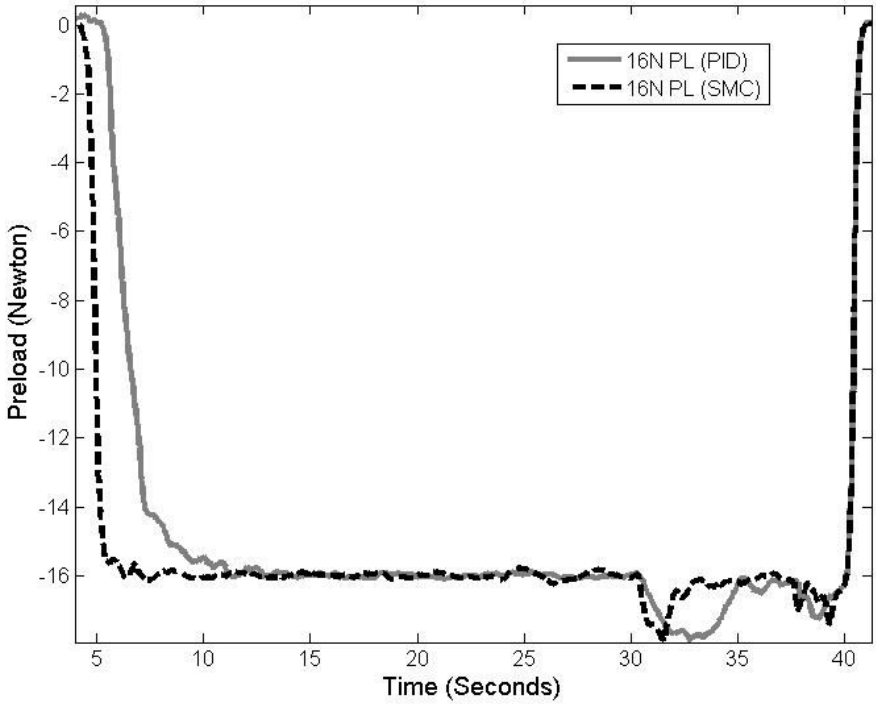

Figure 4.30 : Comparison results for 16N preload.

Besides, friction forces observed are shown in Figure 4.31.

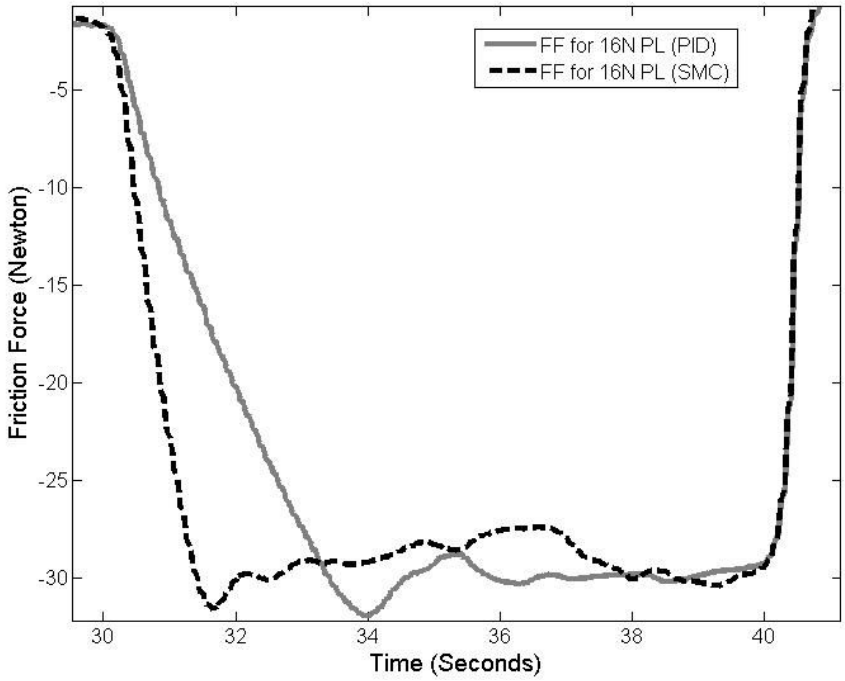

Figure 4.31 : Friction forces for 16N preload.

Measurements taken for 18N preload using SMC and PID are given on Figure 4.32.

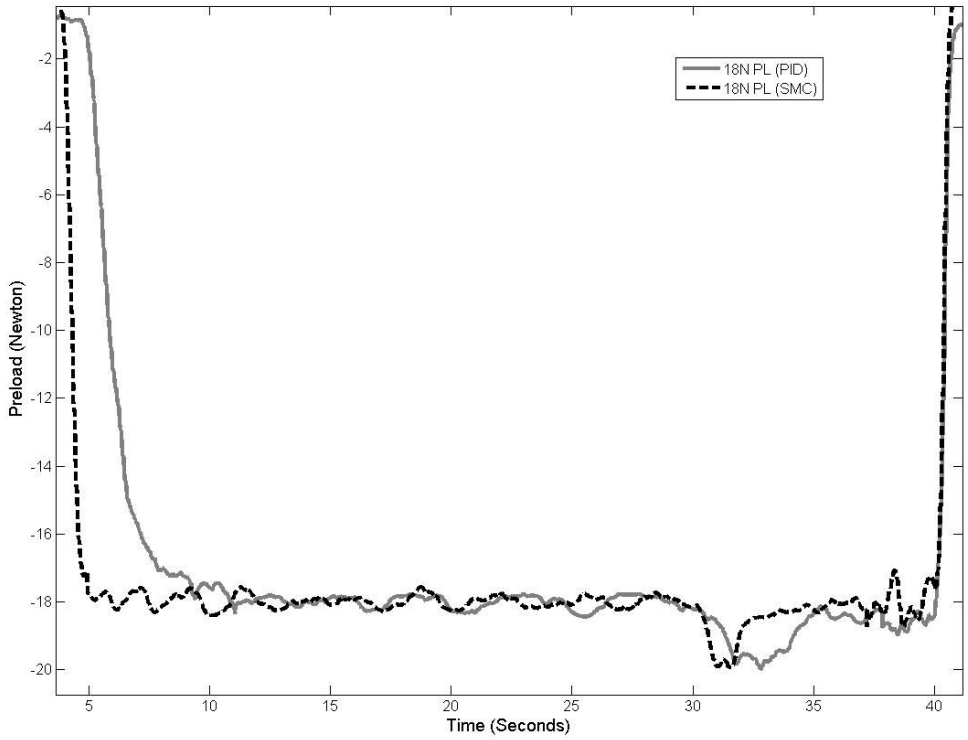

Figure 4.32 : Comparison results for 18N preload.

Besides, friction forces observed are shown in Figure 4.33.

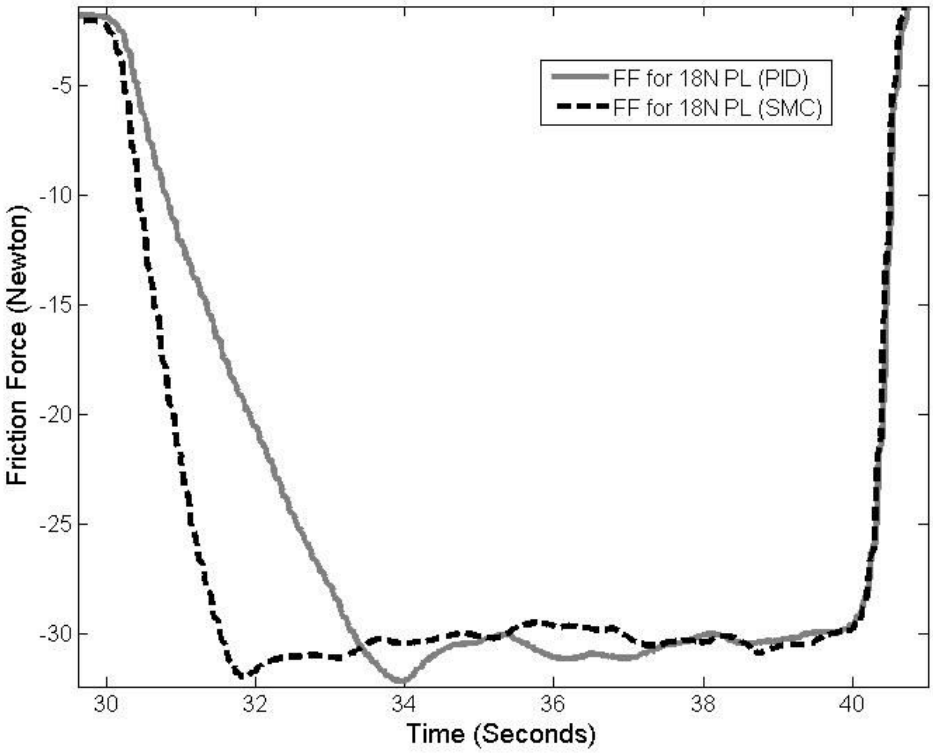

Figure 4.33 : Friction forces for 18N preload.

Observing the results obtained it can be easily seen that the Sliding Mode Controller is more robust and response of the system is better with Sliding Mode Controller.

4.3 Following CAD Data with Force Control

In this section the CAD data tracking which is explained on section 4.1 will be combined with force control which is explained on section 4.2. The combined use of these two control elements is fundamental as there are many application areas in industry which requires robotic arms to follow desired trajectory with desired forces on desired axes. Some examples to these applications are; polishing, deburring, cutting and quality control.

In order to combine these control methods, briefly the hybrid (force/position) control scheme which is explained on Section 5.2.3 will be used. In Figure 4.7 the generalized block diagram of the hybrid controller is given.

Let us investigate the generalized block diagram that is shown on Figure 4.34, which summarizes the control method which will be used.

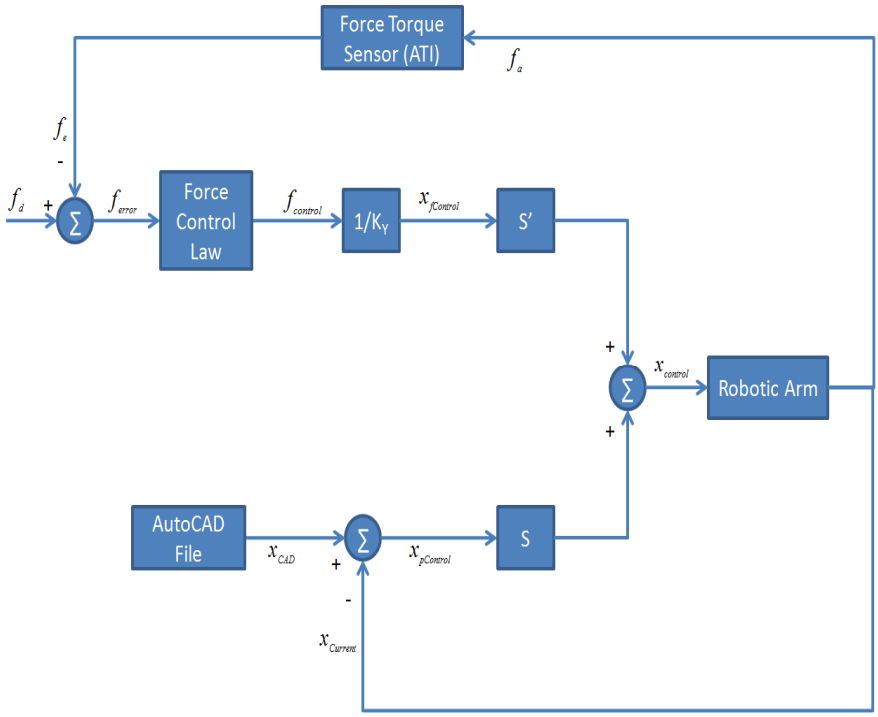

Figure 4.34 : Block diagram of force control and position tracking algorithm.

Force control and CAD position control is combined with the help of the matrix which is mentioned on Section 5.2.3. Generally the matrices in the form given on Equation 4.21 will be used as the force along the Z axis will be controlled and follow desired trajectory with the rest of axes.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad and \quad S' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(4.21)}$$

5. APPLICATIONS AND EXPERIMENTAL RESULTS

5.1 Force Control Applications and Experimental Results

5.1.1 Force data gathering library

In this section, the library which is written in order to achieve force data gathering will be introduced. This library is used in order to obtain the raw data which the controller of the arm gets from the controller of the ATI Force-Torque sensor, process this data in such a way that it can be used in force based applications easily.

The raw force values obtained from the force sensor's controller is 16-bit. As mentioned before, it is calibrated to read values between 660N and -660N on X and Y axes, and values between 1980N and -1980N on Z axis. In order to indicate the sign of the force values, 16-bit is divided into two parts from center; 2^8 of the raw value. The values greater than 2^8 represent negative force values. On the other hand, values smaller than 2^8 represents positive force values.

This library can not only be used as force conversion but also for saving the desired data in files. The controller is upgraded with expansion pack of VAL3 in order to achieve these kinds of tasks.

Note that this data collection program must be called and executed as a synchronous task. The cycle time of the task must be defined such a way that the sampling time matches the command times of the arm. The cycle time for data gathering is chosen 4ms in almost all of our applications as it is the smallest time division VAL3 accepts for sending commands to arm.

The simplified Petri Net graph of the library is given on Figure 5.2. The programs of the library are given in Appendix A.1 in the folder named "Force Gathering". This library will be used in all of force based applications in this thesis.

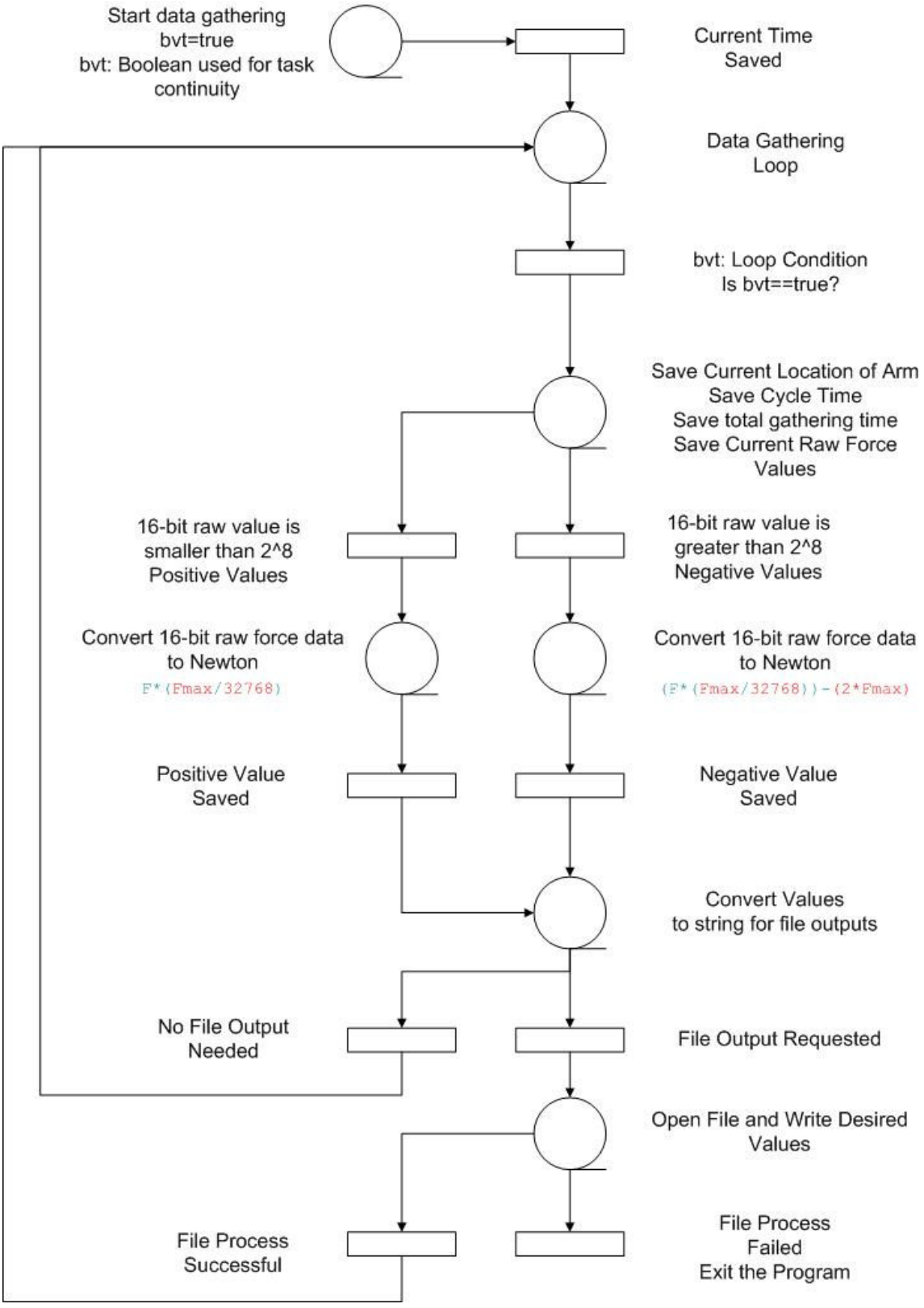

Figure 5.1 : PN graph of force data gathering library.

5.1.2 Coupling effect measuring program and experimental results

This section consists of applications which are used in order to measure coupling effects of force measurement system. Coupling effect mentioned here is the undesired interference between the force measurement axes. For example, imagine that arm is being pushed towards a flat surface along Z axis of end effector (tool), and assume that the force applied has effects only on Z axis. On this occasion, the measurements taken from the force measurement system should also indicate that there is force only on Z axis. But if coupling effects exist the measurements observed will consist of forces on axis other than Z. Note that the measurement system term will be used for not only force/torque transducer and controller but also the end effector and the elements connected together between contact space and the force/torque transducer.

The measurements and results taken from this application will not only help us observe the coupling effects but also give us information about the stiffness effect between contact object and the arm. This effect is mentioned on Section 4.2 with the Equation 4.10.

The force data will be gathered using the data gathering program mentioned on Section 5.2.1.

In order to achieve contact with the environment several probes are used as tool on end effector. The probes which are used are shown on following figures.

Figure 5.2 : Polyamide probe.

On Figure 5.2, the probe which is designed in order to be held with the tool called finger, which is also shown on Figure 3.6. This tool is produced from polyamide, therefore will be called as polyamide probe.

Figure 5.3 : Metal dome probe.

On Figure 5.3, the probe which is designed in order to be directly connected to the force/torque sensor is shown. This tool will be called as metal dome probe as it is made of metal and has a shape of dome.

Figure 5.4 : Pointer probe.

On Figure 5.4, the probe which is designed in order to be directly connected to the force/torque sensor is shown. This tool is also used for defining points using the pendant of the arm; therefore it will be called as pointer.

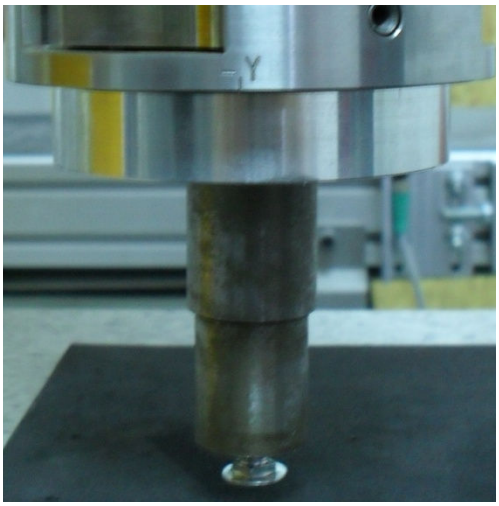

Figure 5.5 : Flexible probe.

On Figure 5.5, the tool which is designed in order to be directly connected to the force/torque sensor is shown. The tool itself consists of two parts connected each other with a screw socket. The part, which makes the contact with environment, is changeable. This tool will be called as flexible probe.

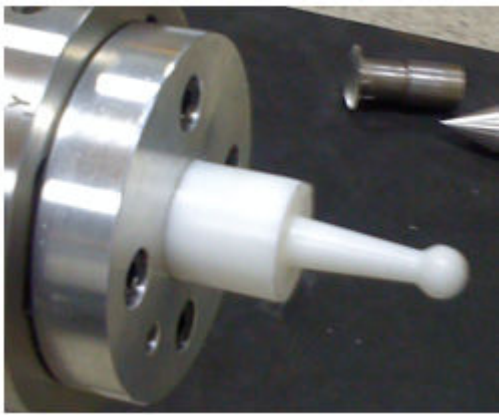

Figure 5.6 : Teflon Probe

On Figure 5.6, the tool which is designed in order to directly connect to the force/torque sensor is shown. The tool is produced using Teflon material therefore it will be called as Teflon probe.

The simplified Petri Net graph of the VAL3 application written is given on Figure 5.7. The application can be found on Appendix A.1 saved under the folder “Coupling Effect”.

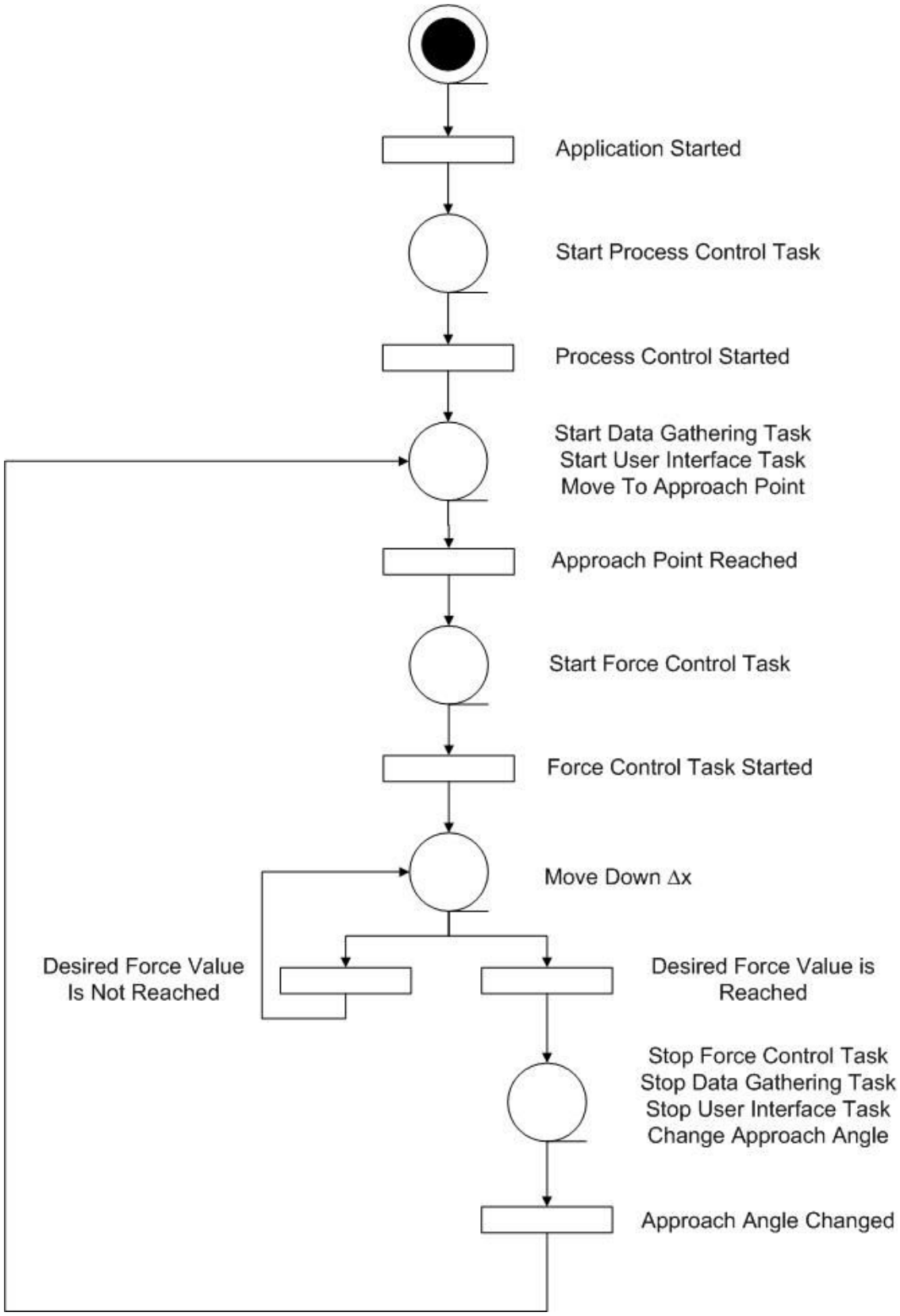

Figure 5.7 : PN graph of coupling effect measurement application.

How application works can be also observed by watching the video file saved with filename “Coupling Effect.mp4” in Appendix A.1. Additionally, Figure 5.8 can be observed in detail with the file “Coupling Effect.jpg” given in Appendix A.1.

Let us investigate the results obtained using the coupling effect measurement application. On the first step, the results of polyamide are given on Figure 5.8, for end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$.

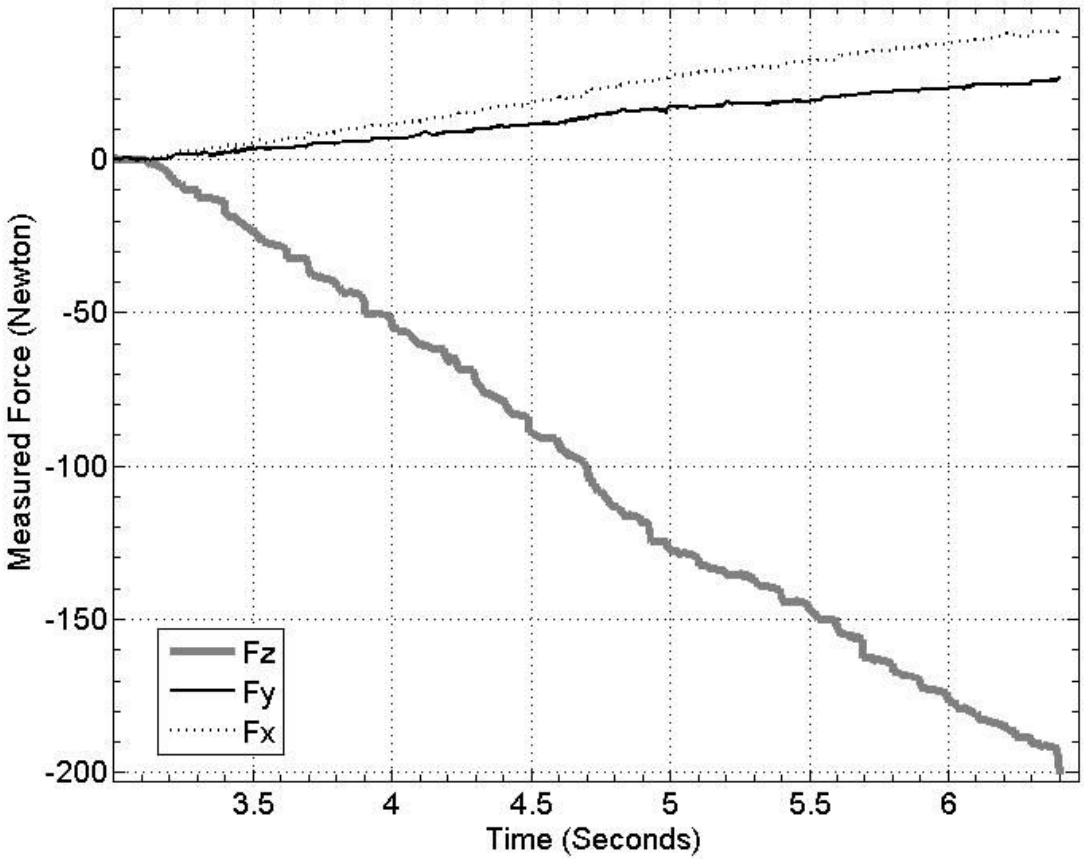

Figure 5.8 : Coupling measurement for polyamide probe.

The Figure 5.8 is the most obvious clue of coupling effect, on this figure if there are no coupling effects, Fx and Fy would be zero.

Next, the results of metal dome probe are given on Figure 5.9, for end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$.

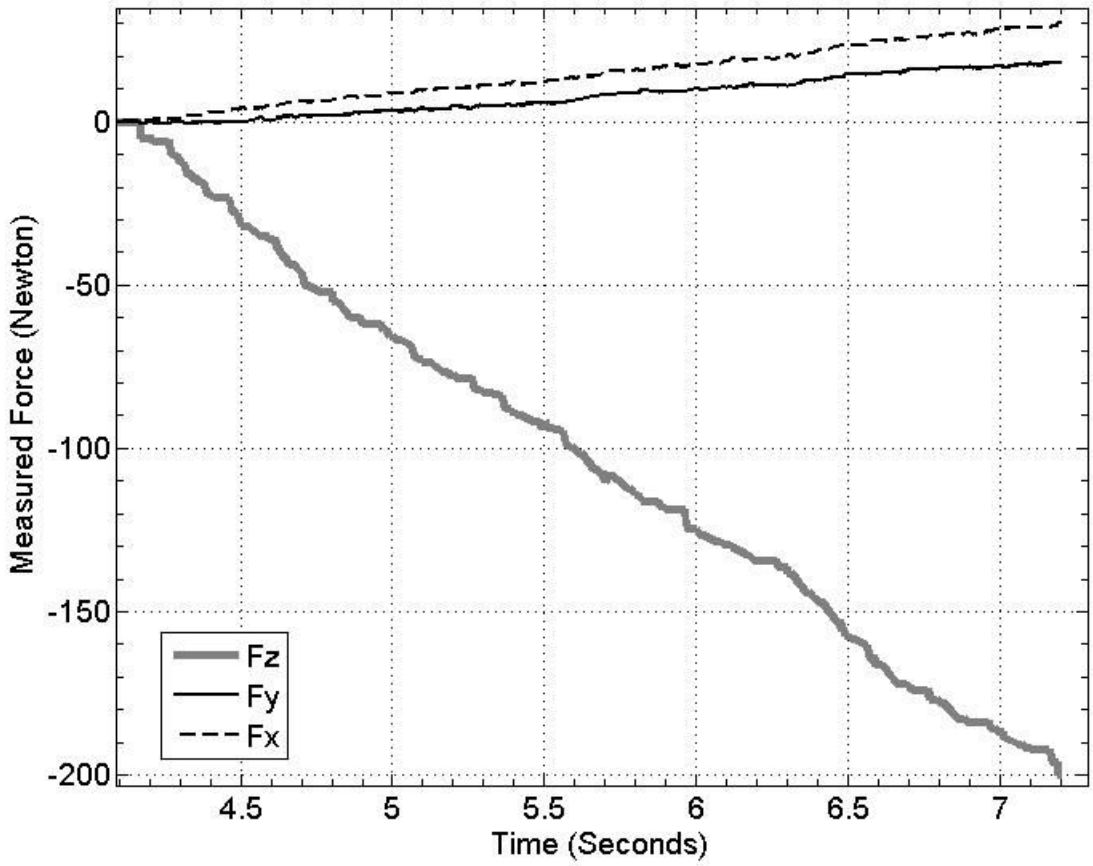

Figure 5.9 : Coupling measurement for metal dome probe.

Investigating the figure given above it can be easily said that the coupling effect which occurs with the use of metal dome probe is at unavoidable amounts but smaller than the polyamide probe. This is because of the fact that the probe is directly connected to the force/toque sensor. The Figure 5.9 is the most obvious clue of coupling effect, on this figure if there were no coupling effects, Fx and Fy would be zero.

Next, the results of pointer probe are given on Figure 5.10, for end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$.

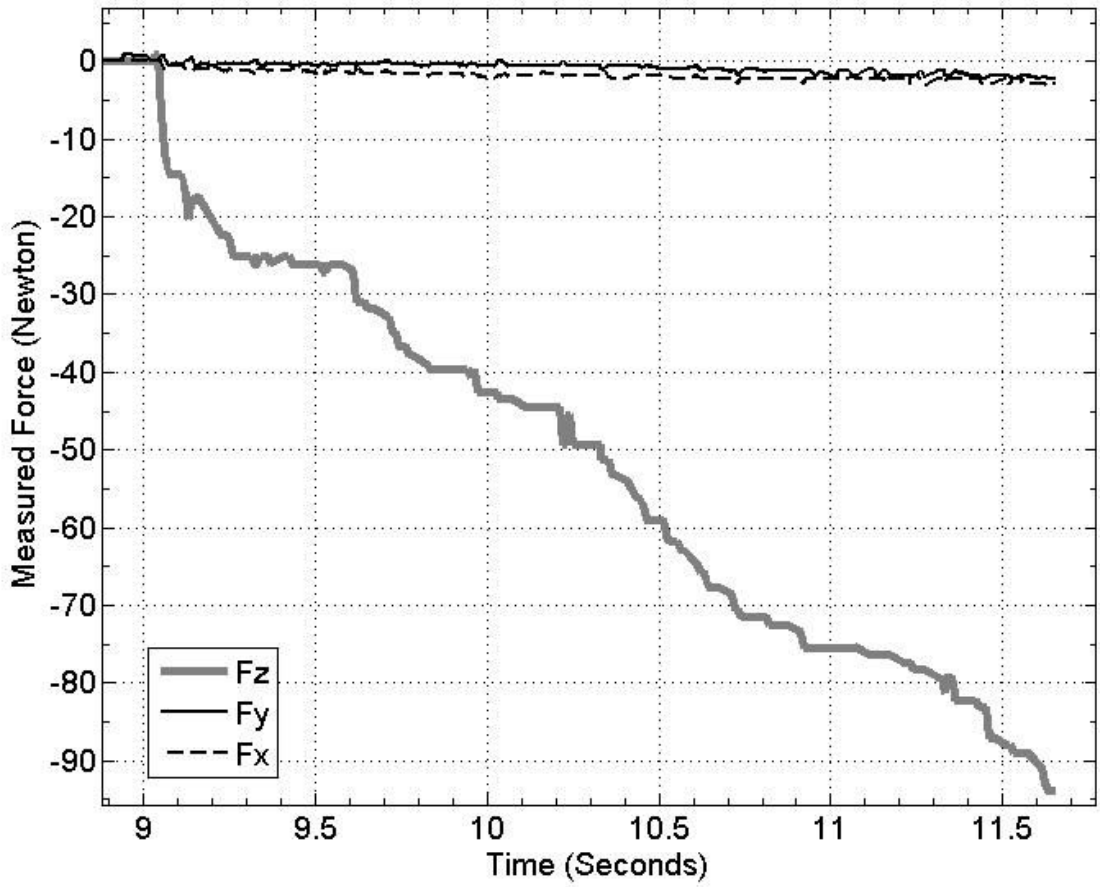

Figure 5.10 : Coupling measurement for pointer probe.

Investigating the figure given above it can be easily said that the coupling effect which occurs with the use of pointer probe is avoidable especially when compared with polyamide and metal dome probes. This is because of the fact that the probe is directly connected to the force/toque sensor, and the force distribution is correctly spread through pointer as the contact point of the probe is smaller. Figure 5.10 shows that the force applied only on Z axis is affecting other axes by avoidable small amounts.

On next step, the results of flexible probe are given on Figure 5.11, for end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$.

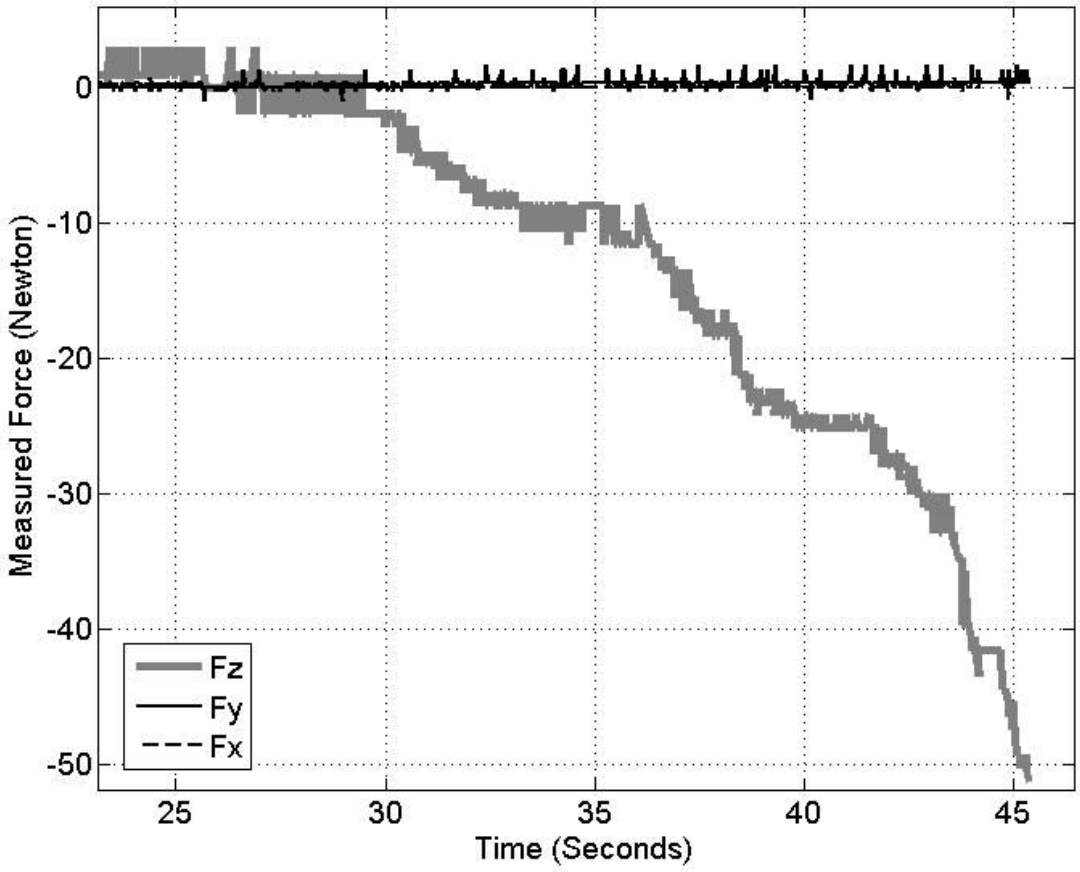

Figure 5.11 : Coupling measurement for flexible probe.

Investigating the figures given above it can be easily said that the coupling effect which occurs with the use of flexible probe is avoidable like pointer probe especially when compared with polyamide and metal dome probes. Note that the maximum force applied on this probe is 50N. This is because of the fact that the screw connection between two elements of the probe would break up or damage in case of greater forces.

The experimental results for Teflon probe coupling effects are given on Figure 5.12, for end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$.

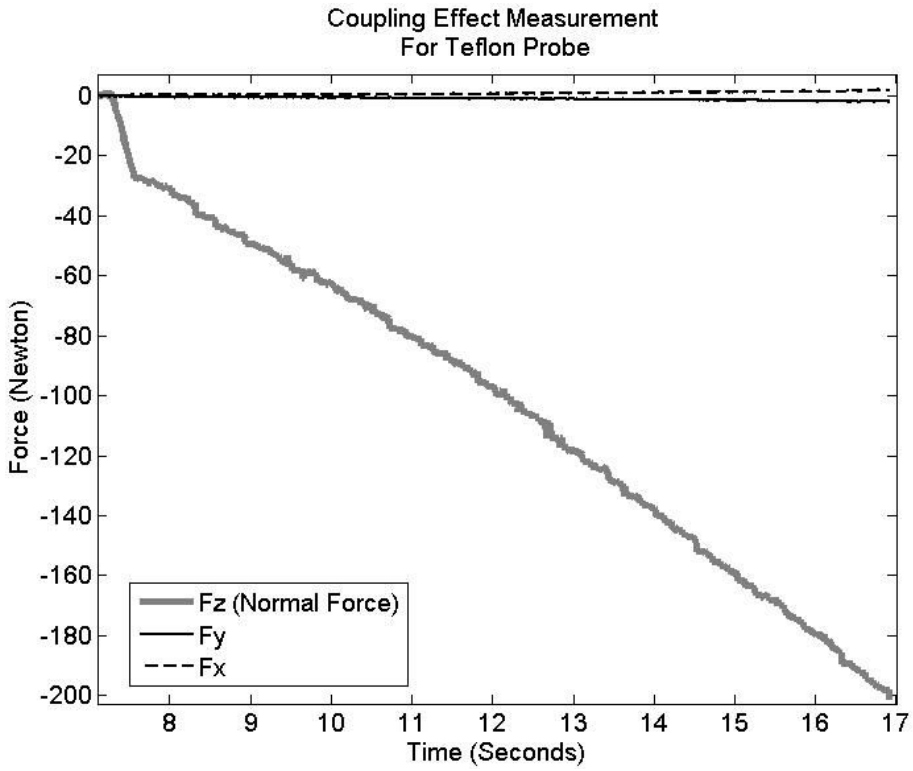

Figure 5.12 : Coupling measurement for Teflon probe.

Investigating the figure given above it can be easily said that the coupling effect which occurs with the use of Teflon probe is avoidable and smaller than other probes. This is because of the fact that the probe is directly connected to the force/toque sensor, and the tip of the probe is produced in a way that contact area is kept as small as possible while enough contact is provided for trustable measurements.

Although not shown on previous figures the position of the arm is also collected on coupling effect measurements. On section 4.2, it is mentioned that the stiffness control will be used on applications. Using the force and position measurements gathered the stiffness relationship between force and position can be observed.

Equation 4.10 gives the relationship between stiffness, force and position. Let us investigate the forces which occurred on flexible probe with given positions and see if this relationship fits our situation. On Figure 5.13 the flexible probe is pressing on the application surface with the end-effector angles $Rx = 180$, $Ry = 0$ and $Rz = 0$. Therefore, the force should only occur on Z axis.

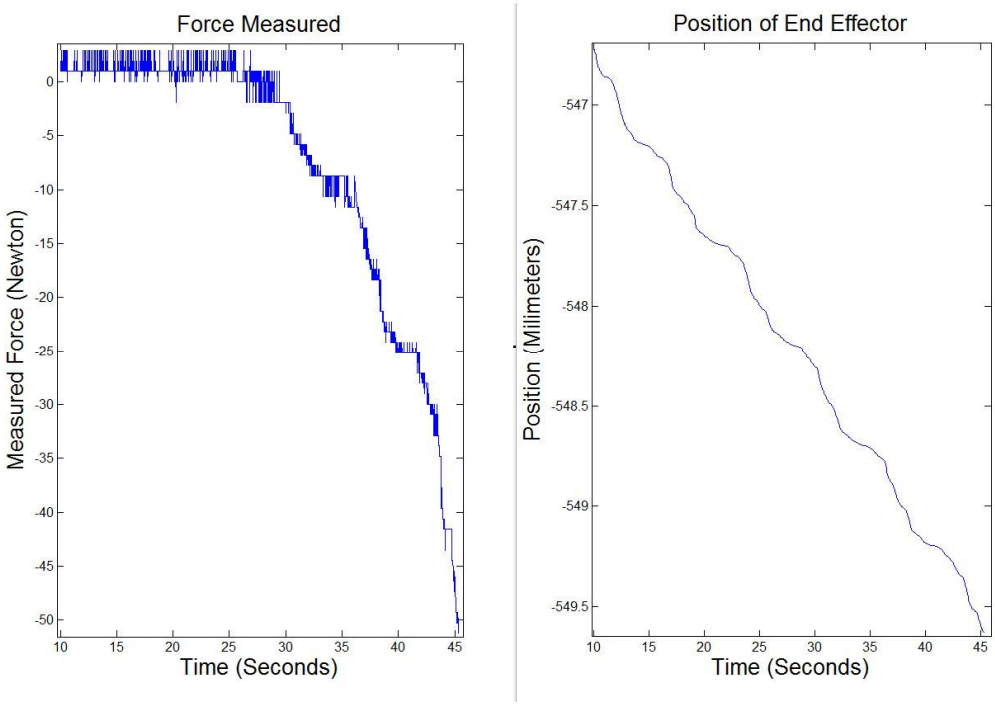

Figure 5.13 : Force and position of end effector on Z axis for flexible probe.

In Figure 5.14 the change in force with respect to position can be observed. It is obvious that the stiffness effect is not completely linear, besides it can be assumed linear while working in specific force ranges.

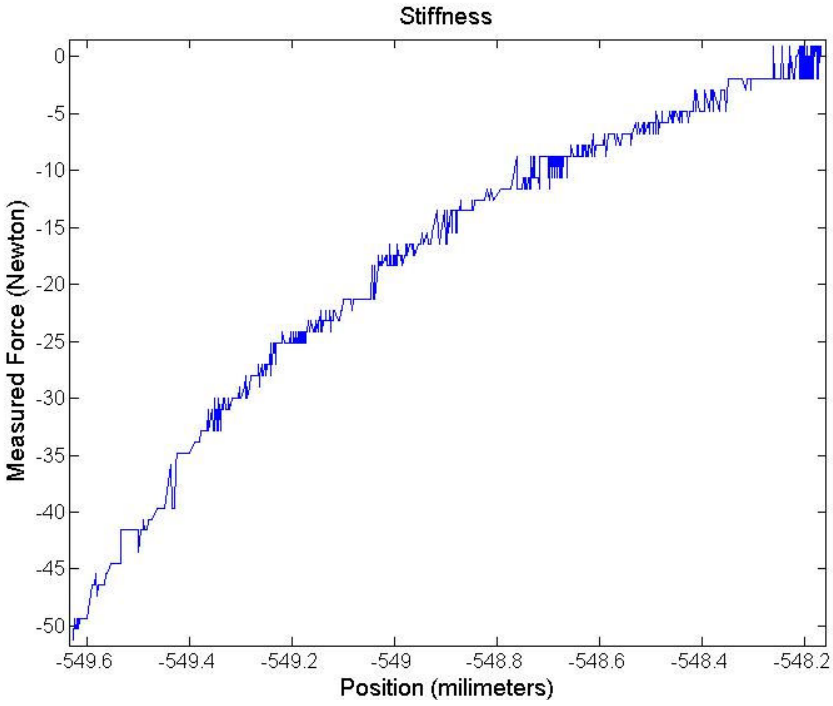

Figure 5.14 : Change in force with respect to position for flexible probe.

On Figure 5.15 the Teflon probe is pressing on the application surface with the end-effector angles $Rx = 180, Ry = 0$ and $Rz = 0$. Therefore, the force should only occur on Z axis. Using this figure the change in force with respect to position can be observed. It is obvious that the stiffness effect is not completely linear, besides it can be assumed linear while working in specific force ranges.

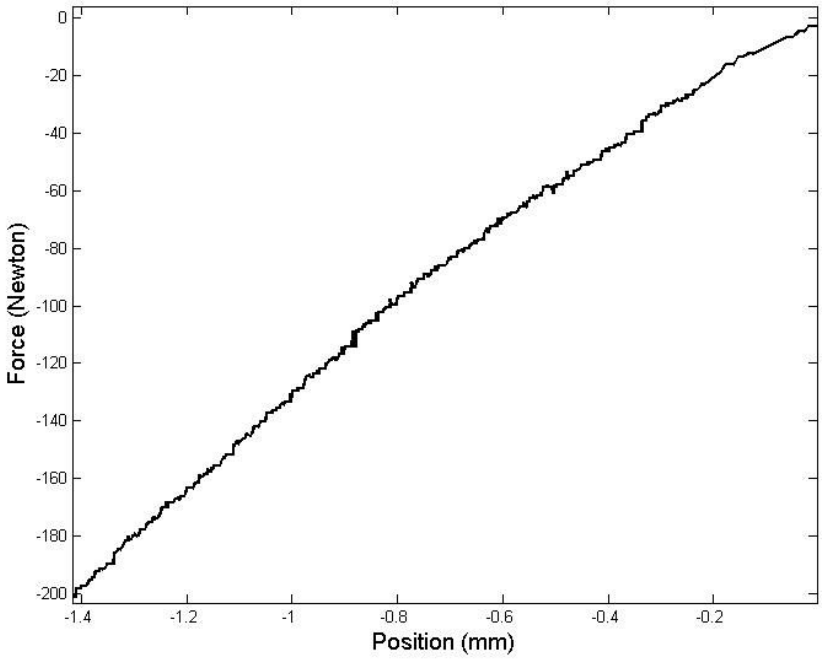

Figure 5.15 : Change in force with respect to position for teflon probe.

5.1.3 Program for measuring friction forces on surfaces and experiments

This section consists of application which is written for measuring friction forces on surfaces. In order to achieve friction measurements, arm presses on the object to be measured with the desired tool and with the desired constant force on the first step. The constant force is achieved using the control algorithm which is mentioned on Section 4.2. On the next step after the desired force is obtained, the arm moves towards the object to be measured causing a friction force between the tool and the object. This motion is achieved with the control algorithm mentioned on Section 4.2. After the measurement is complete the arm rises to a safe point and the measurement data is saved in a file.

How application works can be observed by watching the video file “Friction Measurement.mp4” which is given in Appendix A.1.

The simplified Petri Net Graph of the application for friction measurements is given on Figure 5.16.

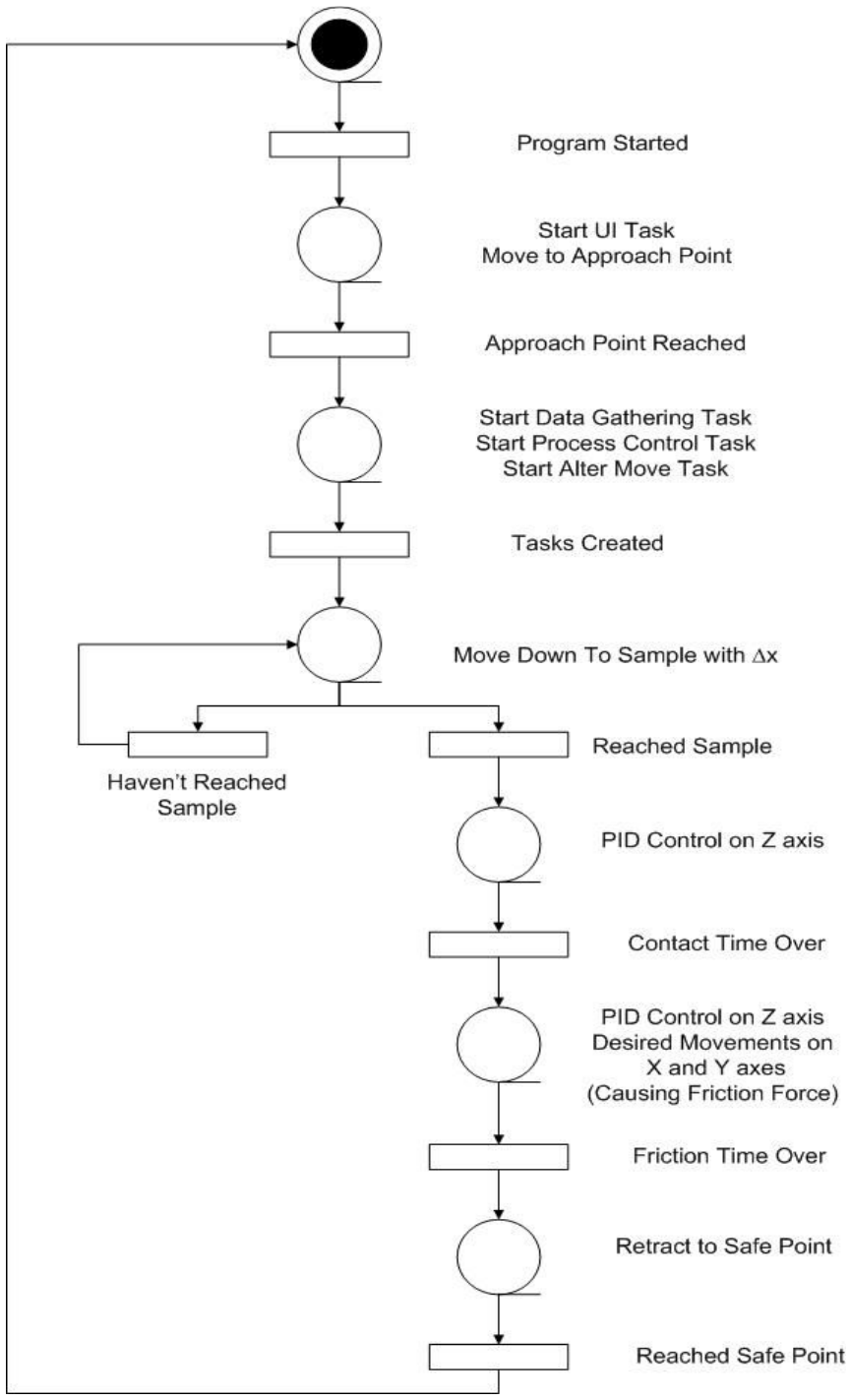

Figure 5.16 : PN graph of friction measurement application.

The graph can be examined in a more detailed way by observing picture file "Friction Measurement. jpg" given on Appendix A.1.

5.1.3.1 Friction measurements on textile fabrics

The preloads which are applied for friction measurements using the satin fabric are given on Figure 5.17. The value of velocity which causes friction is 1.25 mm/sec for all measurements.

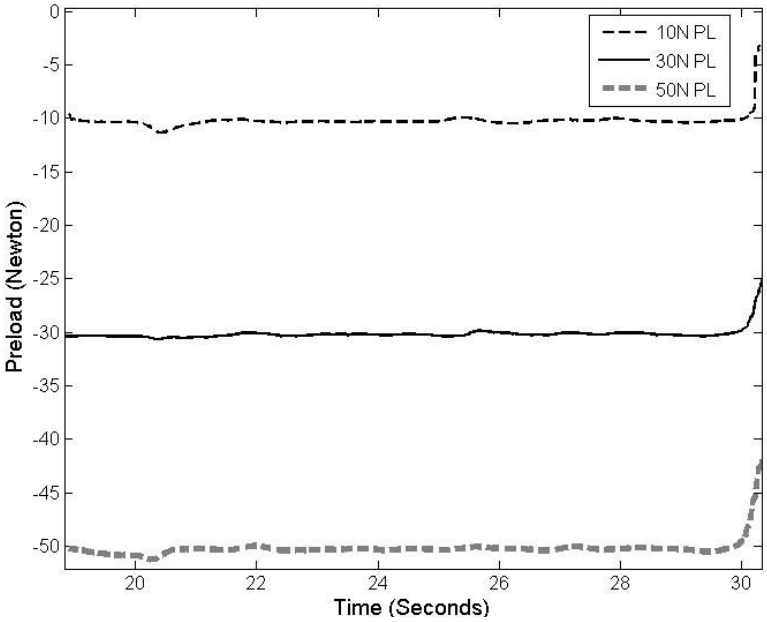

Figure 5.17 : Preloads of friction measurements on satin fabric.

The friction values obtained with preloads above are given on Figure 5.18.

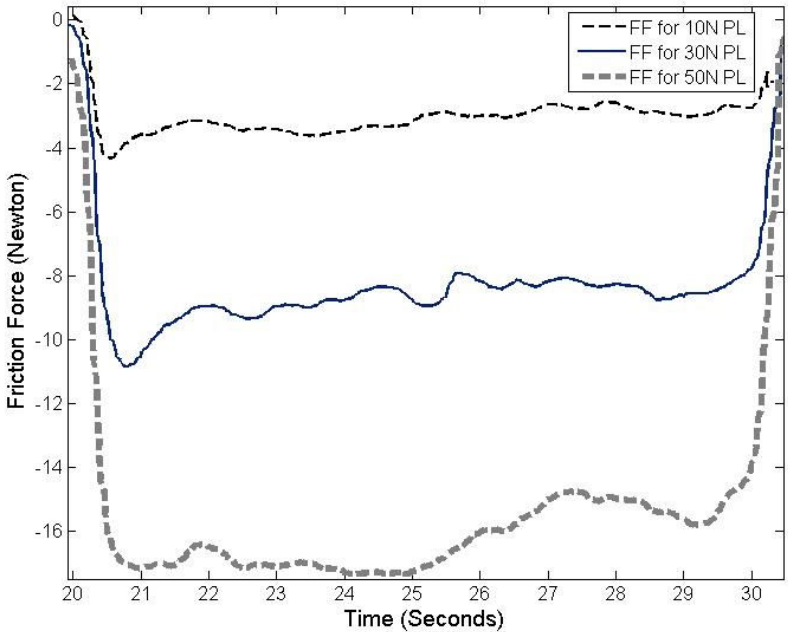

Figure 5.18 : Friction forces for satin fabric.

The preloads which are applied for friction measurements using the plain fabric are given on Figure 5.19. The value of velocity which causes friction is 1.25 mm/sec for all measurements.

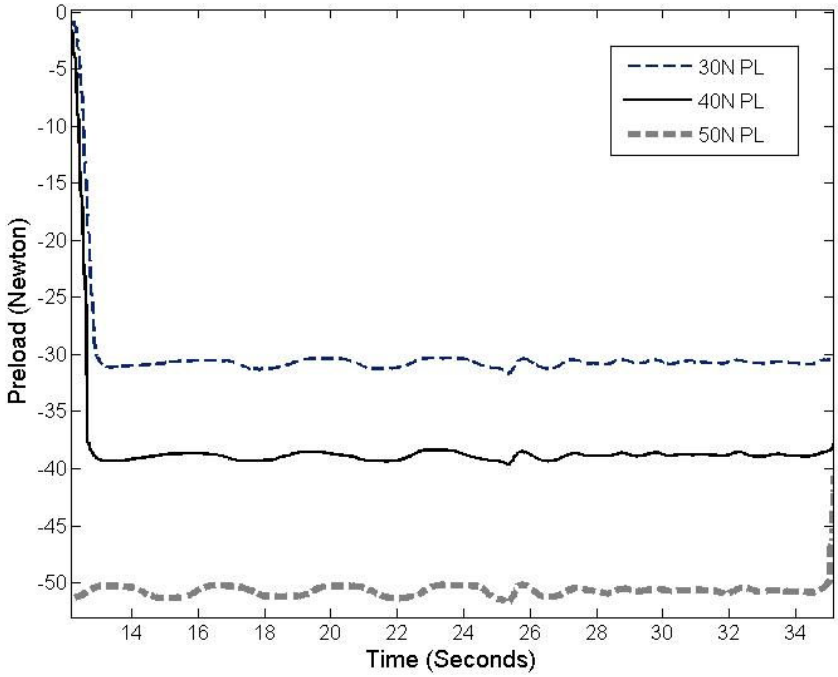

Figure 5.19 : Preloads of friction measurements on plain fabric.

The friction values obtained with preloads above are given on Figure 5.18.

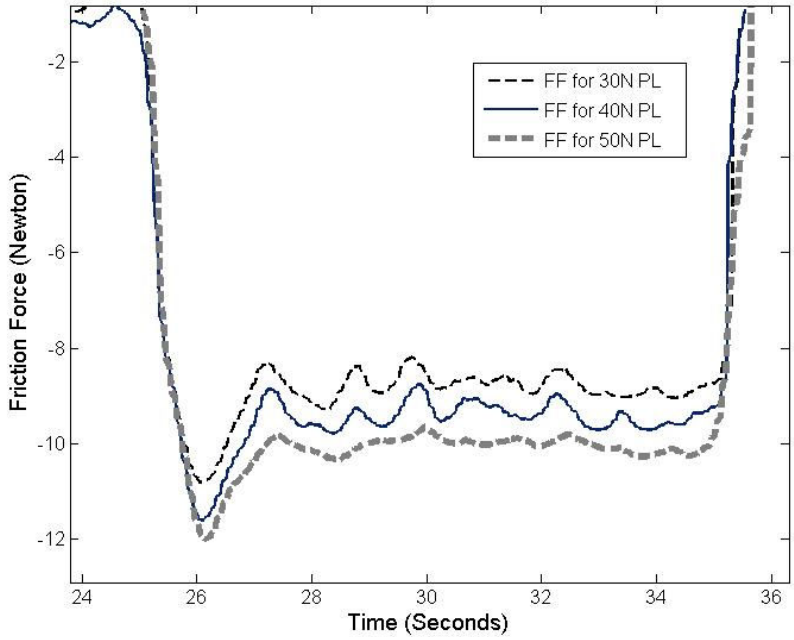

Figure 5.20 : Friction forces for plain fabric.

The preloads which are applied for friction measurements using the twill fabric are given on Figure 5.21. The value of velocity which causes friction is 1.25 mm/sec for all measurements.

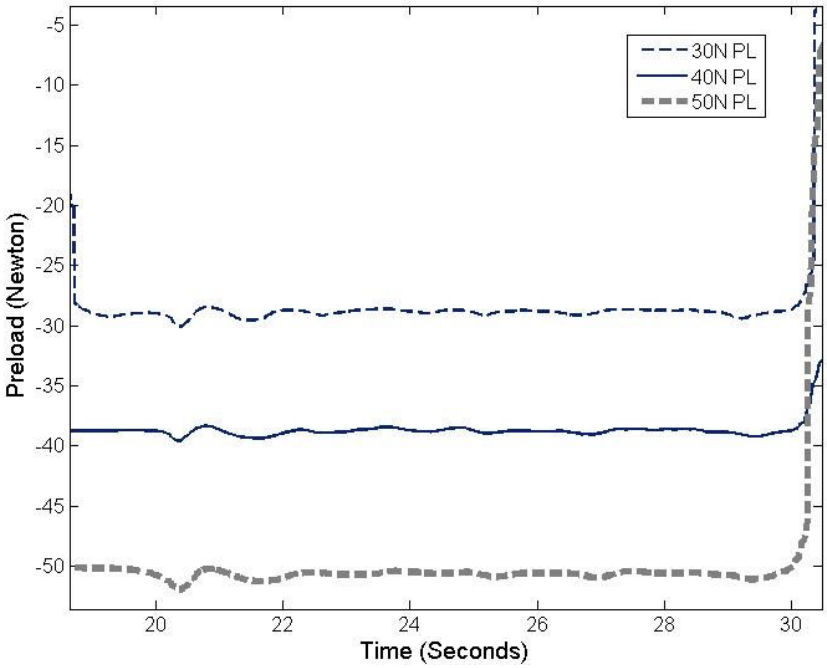

Figure 5.21 : Preloads of friction measurements on twill fabric.

The friction values obtained with preloads above are given on Figure 5.22.

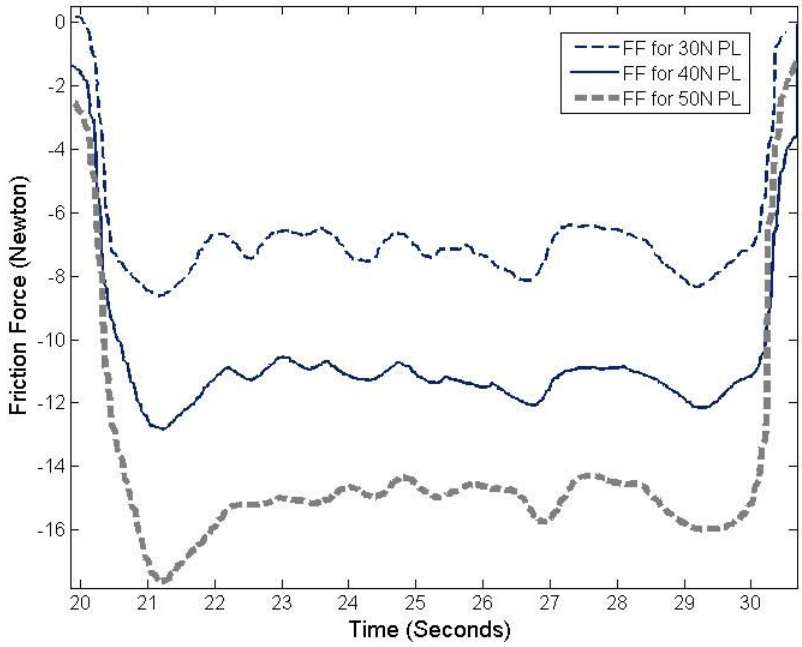

Figure 5.22 : Friction forces for twill fabric.

5.1.3.2 Friction measurements with teflon probe on elastomer surface

In order to obtain friction force, a preload is applied to Elastomer surface with Teflon probe.

The following results are obtained with constant velocities. The motion which creates friction is 1 mm/sec. The preloads are given on Figure 5.23.

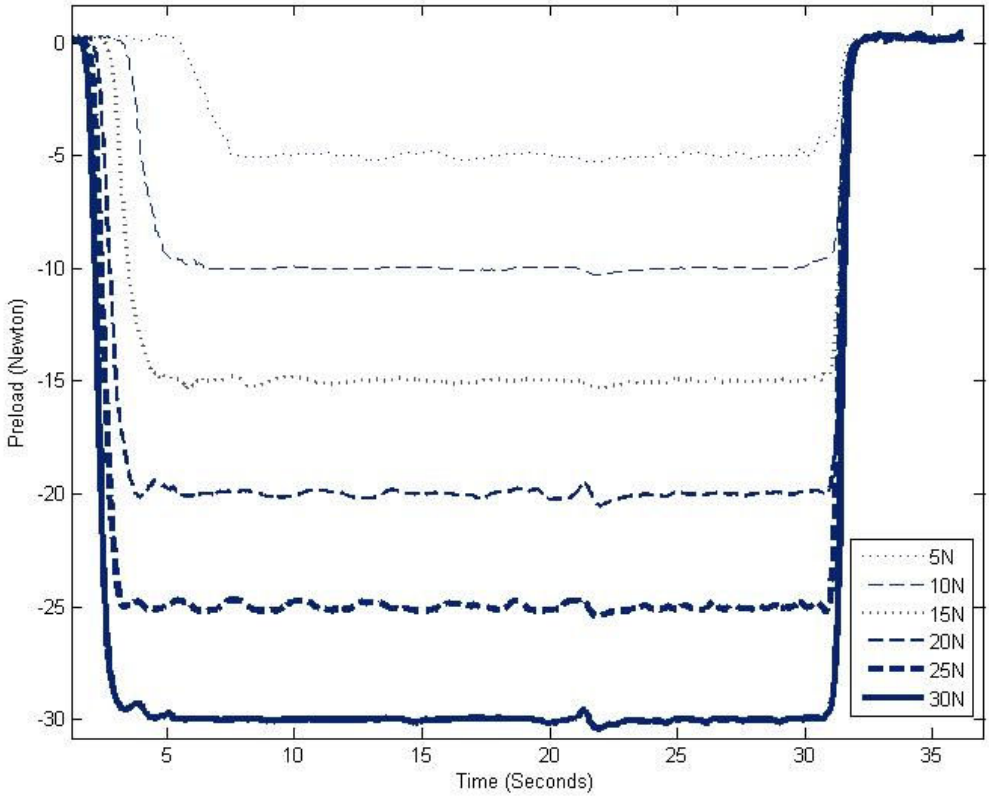

Figure 5.23 : Preloads for elastomer and teflon.

The friction forces are shown on Figure 5.24. The motion that creates friction starts at 21th second. The legend in this figure shows the preload values for given friction forces.

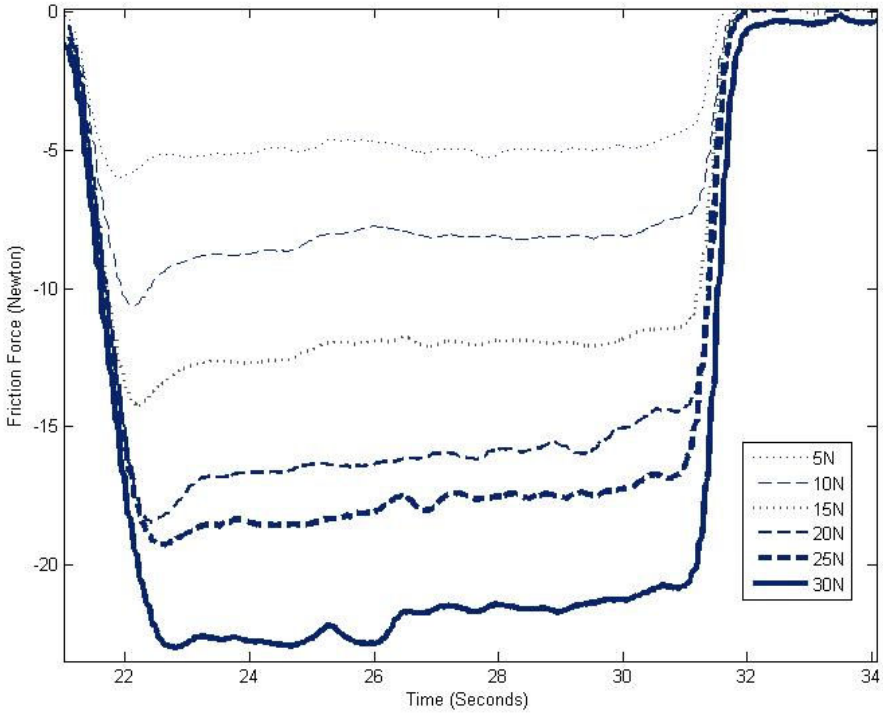

Figure 5.24 : Friction Forces for elastomer and teflon.

Using these results changes in friction coefficient with respect to preload is shown in Figure 5.25.

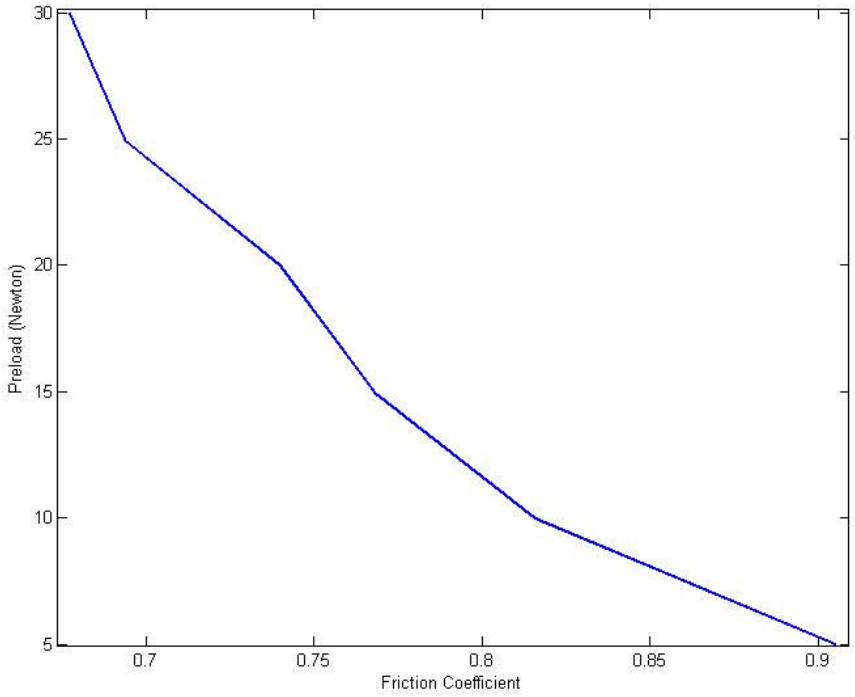

Figure 5.25 : Friction coefficients with respect to preloads.

Besides, experimental results for constant force (30N) and variable speeds are given on Figure 5.26.

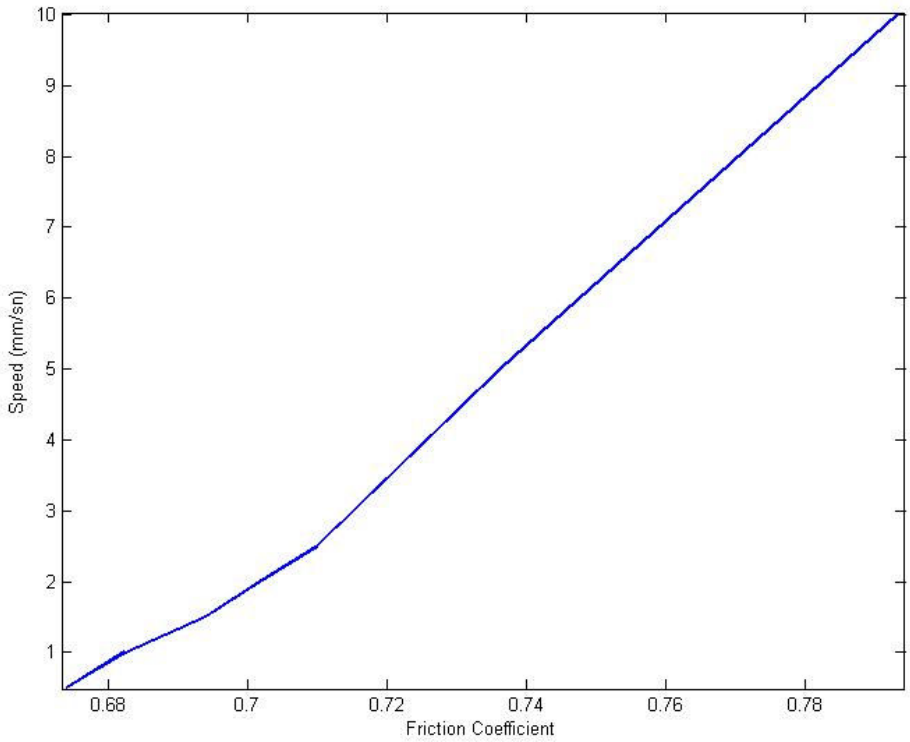

Figure 5.26 : Friction coefficients with respect to speed values.

It can be easily observed that as preload force increases the friction coefficient decreases. On the other hand, as speed of the motion which creates friction increases the friction coefficient increases.

5.1.3.3 Friction measurements with glass probe on elastomer surface

In this section the experimental measurements obtained from friction glass probe and Elastomer surface is explained. The preload forces which are applied on surface are given on Figure 5.27.

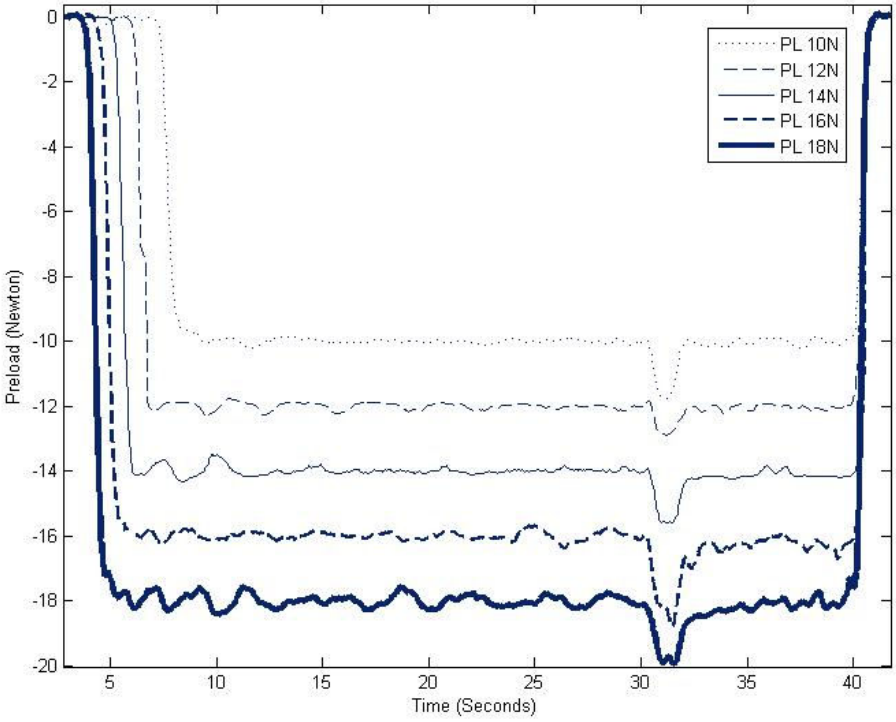

Figure 5.27 : Preloads for elastomer and glass friction.

The friction forces observed for these preloads are given on Figure 5.28.

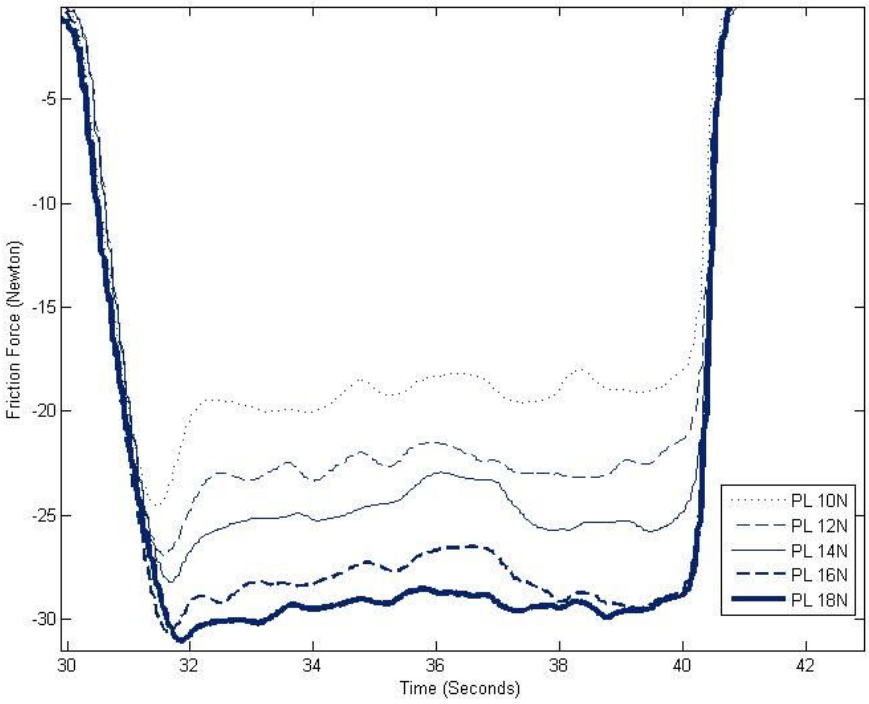

Figure 5.28 : Friction forces for elastomer and glass.

Using these results changes in friction coefficient with respect to preload is shown in Figure 5.29.

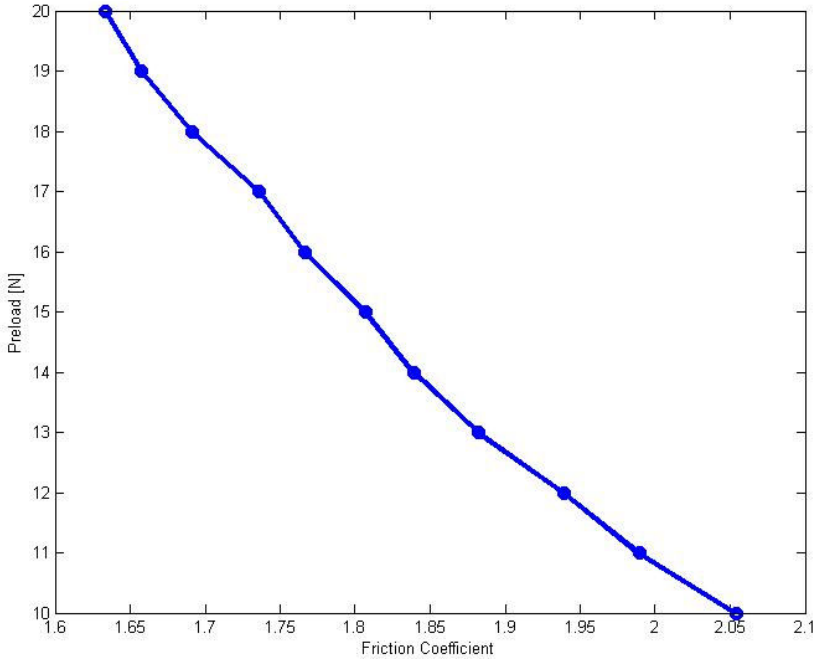

Figure 5.29 : Friction coefficient with respect to preload.

5.1.3.4 Theoretical friction calculation for comparison

In order to calculate the theoretical friction values, calculation of the probe's contact area is required. In order to achieve this information Equation 5.3 will be used. This equation is obtained from Johnson-Kendall-Roberts model for elastic contact.

$$a = \left[\frac{R}{K}\left(P + 3\pi WR + \sqrt{6\pi WRP + (3\pi WR)^2}\right)\right]^{1/3} \quad \textbf{(5.3)}$$

In this equation;

- a: Contact Area Radius (m);
- R: Radius of Curvature (m) ;
- K: Effective Modulus (N/m^2) ;
- P: Normal Force (N) ;
- W: Work of Adhesion (J/m^2).

Effective modulus value will be calculated using Equation 5.4.

$$\frac{1}{K} = \frac{3}{4}\left[\frac{1-v_1^2}{E_1} + \frac{1-v_2^2}{E_2}\right] \quad \textbf{(5.4)}$$

In this equation;

- v_1: Poisson ratio of glass probe
- v_2: Poisson ration of elastomer on surface
- E_1: Young Modulus value of glass probe (N/m^2)
- E_2: Young Modulus value of Elastomer (N/m^2)

Using the contact area radius which will be found using Equation 5.3 the contact area of the probe can be easily found using Equation 5.5.

$$A = \pi a^2 \quad \textbf{(5.5)}$$

And on last step the friction force can be calculated using Equation 5.6.

$$F = \tau * A \quad \textbf{(5.6)}$$

In this equation τ is the shear value between glass probe and elastomer surface.

Using MATLAB the theoretical values are calculated with respect to equations given above. The calculated results are compared with experimental data and the results are shown on Figure 5.30.

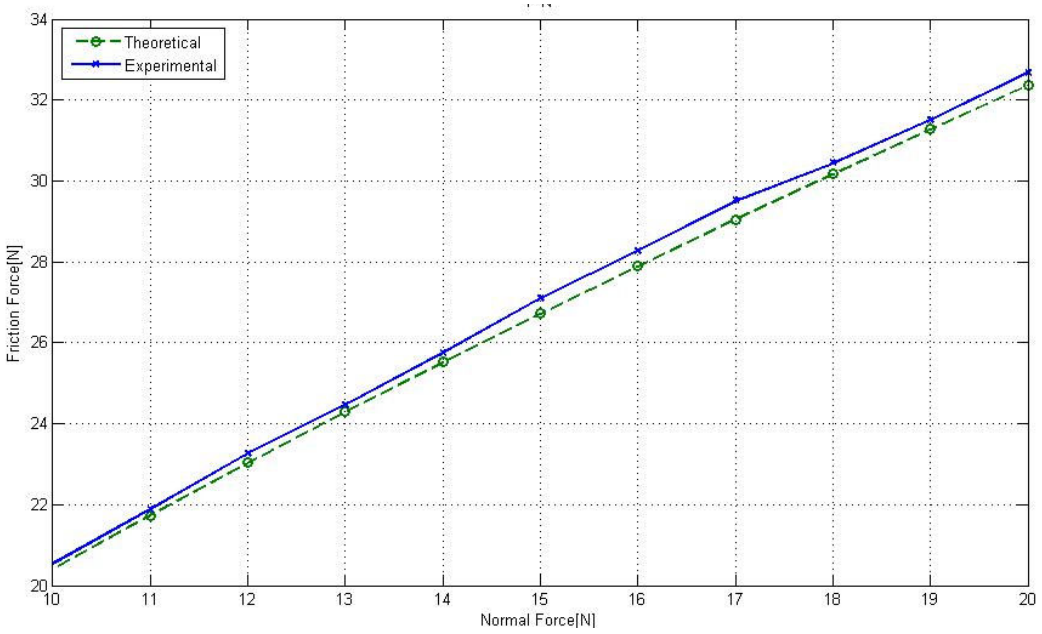

Figure 5.30 : Comparison of experimental and theoretical values.

5.1.4 Program for human robot collaboration

In this section an application based on human robot collaboration is investigated. Due to industrial demands in assembly processes this concept is a part of highly efficient production providing flexibility and adaptability compared to full automation [9].

In order to achieve the task, the controller algorithm which is introduced on Section 8.5 is used. The arm is controlled with force on X, Y, and Z axes. The PID reference is set to 0N force on these axes; therefore the arm will move on opposite direction on axes force is applied, trying to satisfy reference signal zero.

The execution of application is illustrated on video “Human Machine 01.mp4” saved in Appendix A.1. If desired sliding motions can also be achieved with tuning PID constants. This is also illustrated on video “Human Machine 02.mp4” saved in Appendix A.1.

The VAL3 application is given on Appendix A.1 saved under the folder “Human Robot Collaboration”.

The simplified Petri Net Graph of the application is given on Figure 5.31.

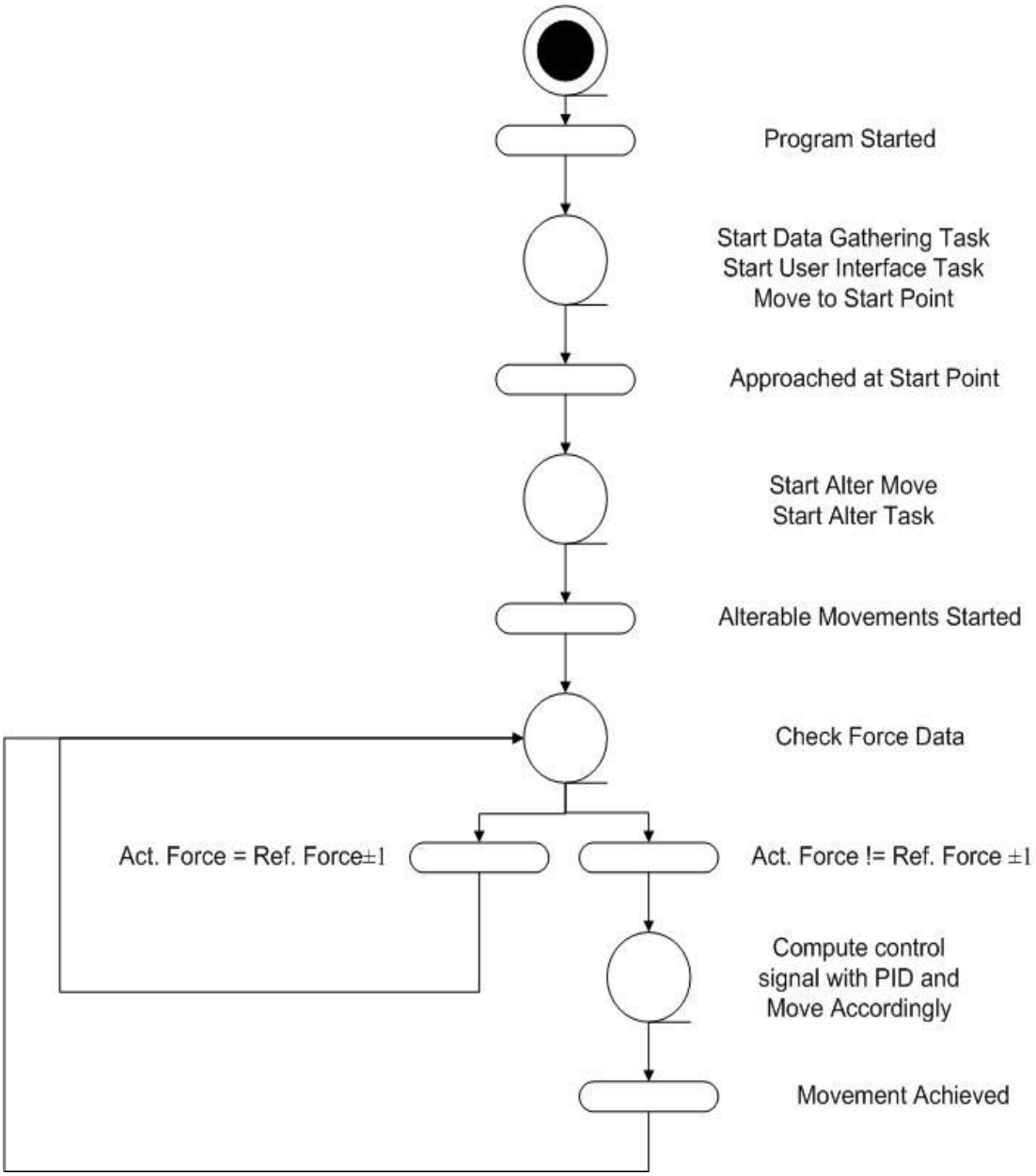

Figure 5.31 : PN Graph of human robot collaboration application.

5.2 Program for 3D CAD Data Tracking and Experimental Results

This section consists of application which is prepared in order to achieve 3D CAD data tracking. As the working algorithm of this subject is explained on a detailed way in Section 4.1, only applications will be explained here.

The VAL3 application written in order to achieve 3D CAD Data tracking is given on Appendix A.1 under the folder named “CAD Tracking”. The execution of the

application can be observed using the video file “CAD Helix 01.mp4”. In this file arm is programmed so that it will follow the helix path which is given on Figure 4.3. If the video is watched carefully it can be observed that the tail section of the helix is followed roughly as the distance between points in CAD file is chosen too long. By shortening distances a better tracking can be obtained as shown on video file “CAD Helix 02.mp4”.

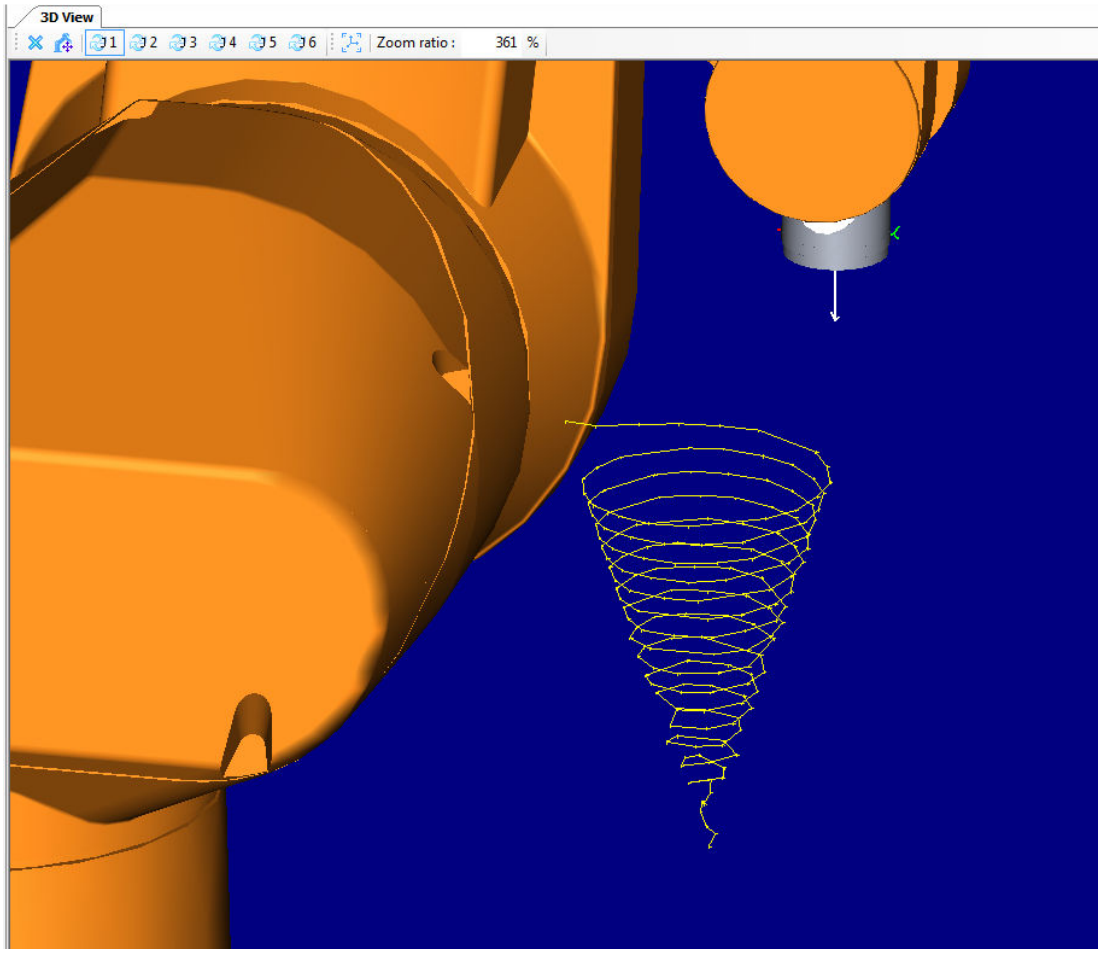

Figure 5.32 : Simulation of 3D helix path tracking 1.

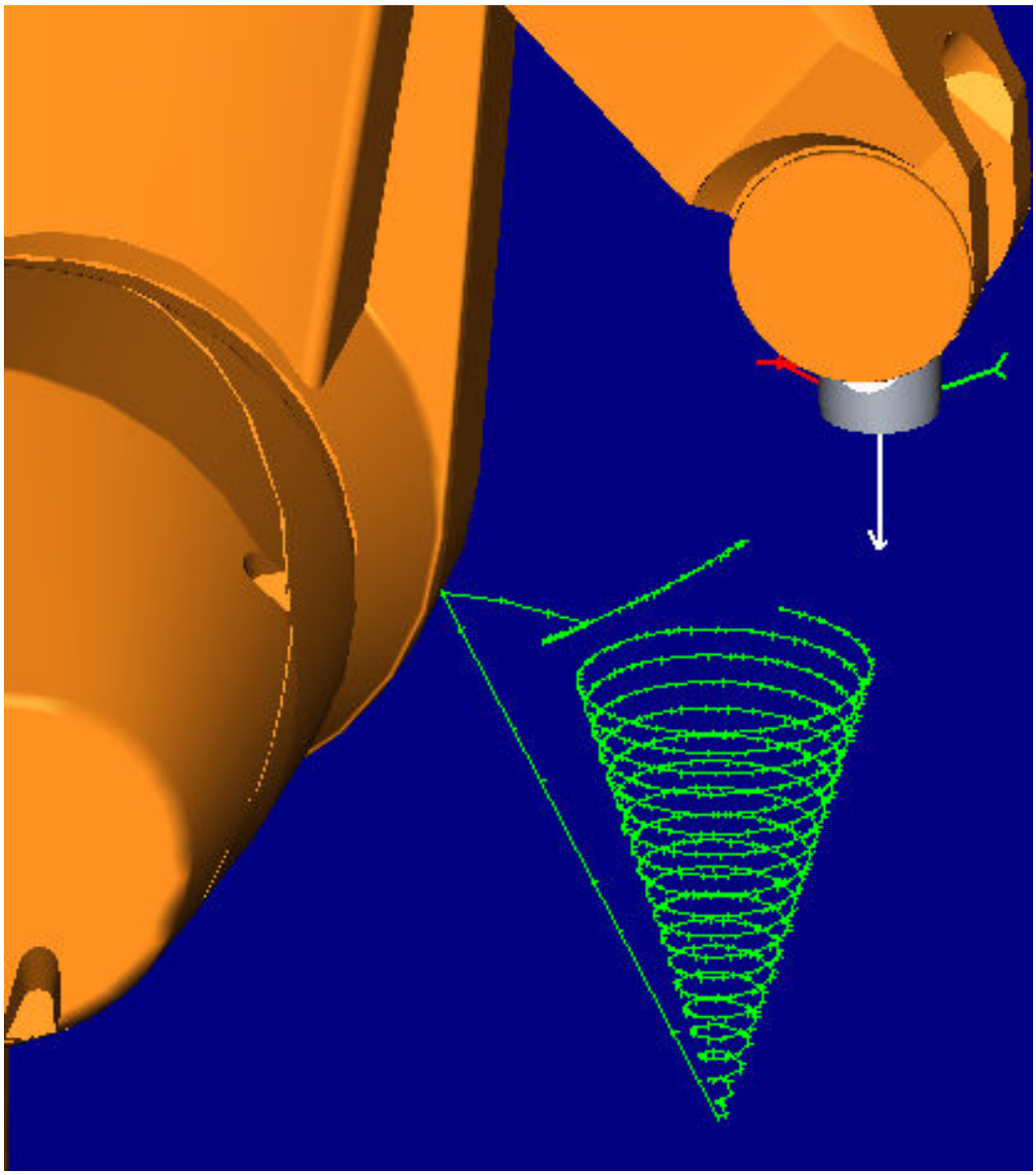

Figure 5.33 : Simulation of 3D helix path tracking 2.

Figure 5.32 shows the simulator results of helix with points defined 20mm distance between each other. On the other hand the Figure 5.33 shows the simulator results of helix with points defined 5mm distance between each other.

Note that; although choosing a small distance between points gives better results, the number of point increases. This choice results with the increase of array dimension in which the transformation and point data is hold. A great number of point and transformation data can cause system overruns and errors, therefore the number of points to be used should be chosen carefully.

5.3 Program for Tracking Surfaces Using CAD Data with Constant Forces

In this section, application written for tracking surfaces using CAD data with constant forces is explained. The control algorithm which is explained on Section 5.3 is used in order to achieve this process. Briefly, force control algorithm is used for Z axis while CAD data is being followed on X and Y axes.

Because of the reason alterable motions have to be used for force control and in the meantime a path must be followed on X and Y axes, all axes must be controlled in real time with alter instructions unlike 3D CAD data path tracking process where general motion instructions are adequate.

The VAL3 application written for this process is given in Appendix A.1 under the folder “CAD and Force”.

The application has been tested for many different CAD files. The drawing saved with the name “circle.dxf” which can also be found in Appendix A.1 is illustrated on Figure 5.34. The execution of application for this drawing can be observed by watching video “CAD Circle 01.mp4” saved in Appendix A.1. Besides, the simulation result is illustrated in Figure 5.35.

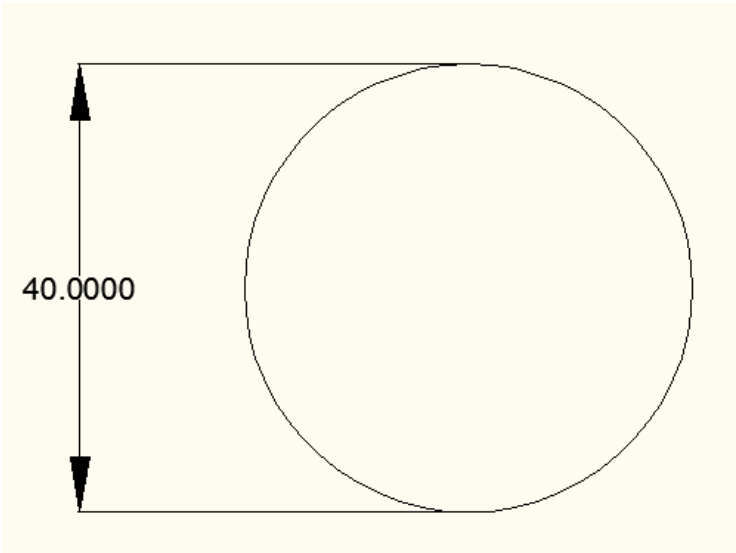

Figure 5.34 : 2D circle drawing.

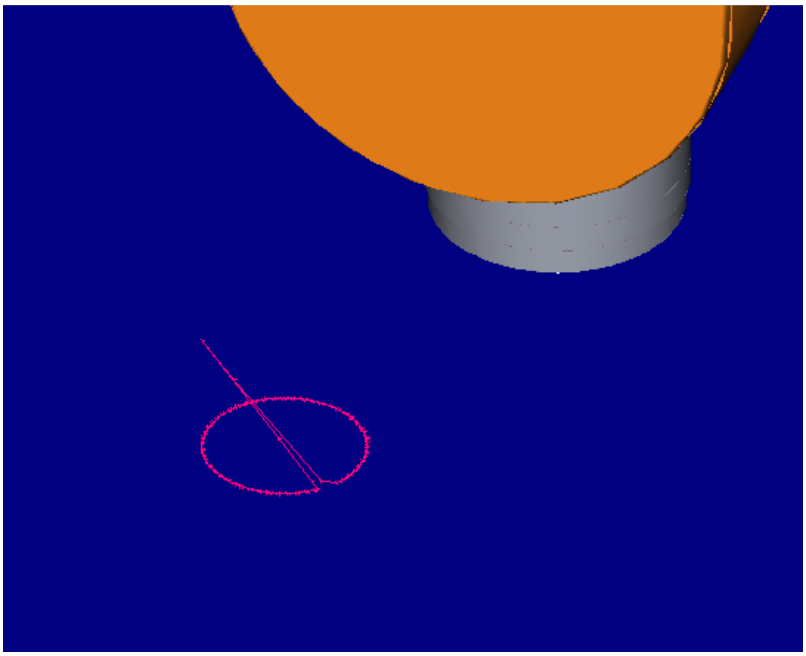

Figure 5.35 : Simulation results for circle.

The force measurements obtained from system for circle is shown on Figure 5.36. Note that reference force is 40N.

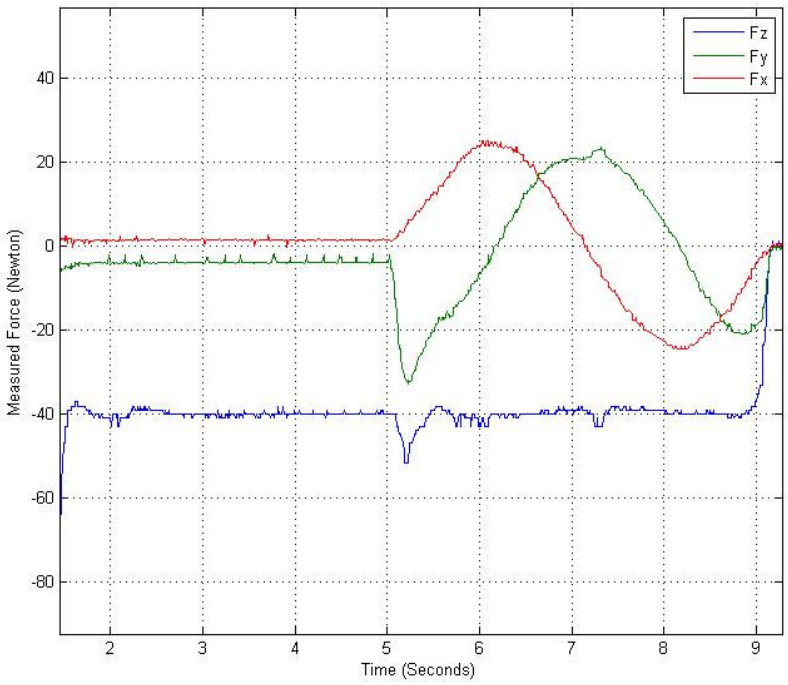

Figure 5.36 : Force measurements for circle.

The drawing saved with the name “triangle.dxf” which can also be found in Appendix A.1 is illustrated on Figure 5.37. The execution of application for this drawing can be observed by watching video “CAD Triangle 01.mp4” saved in Appendix A.1. Besides, the simulation result is illustrated in Figure 5.38.

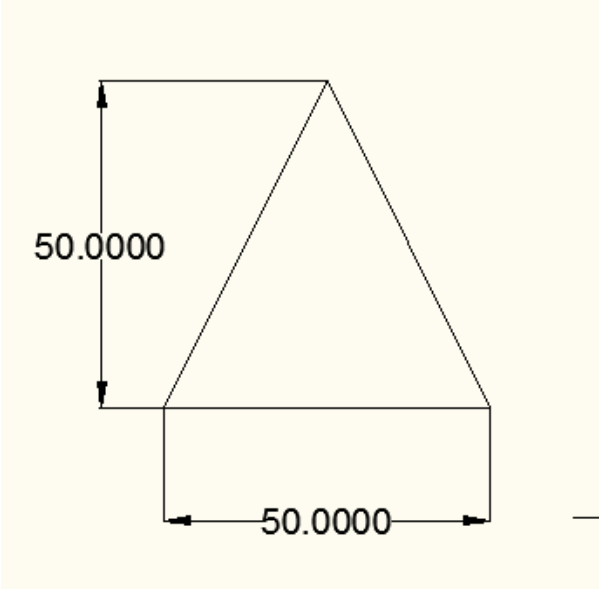

Figure 5.37 : 2D triangle drawing.

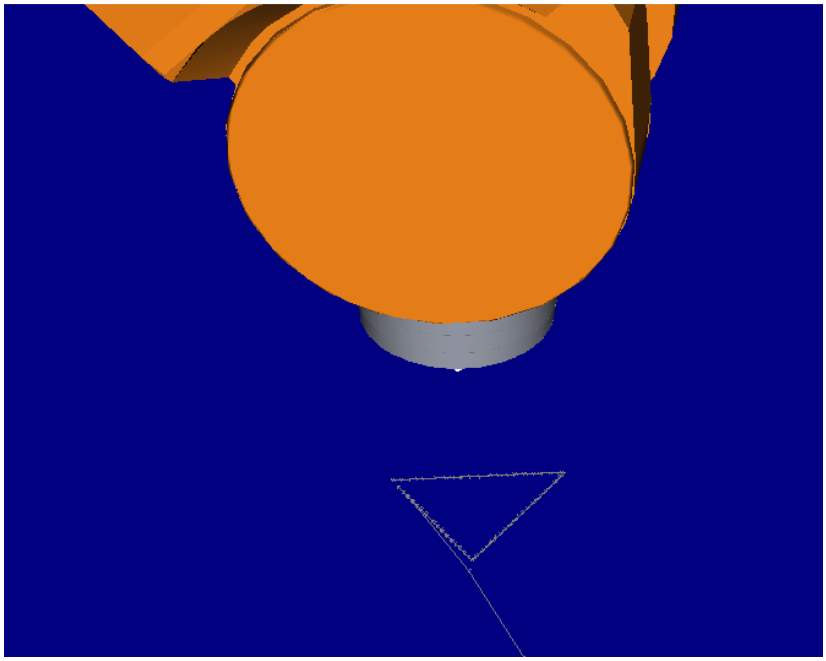

Figure 5.38 : Simulation results for triangle.

The force measurements obtained from system for circle is shown on Figure 5.39. Note that reference force is 40N.

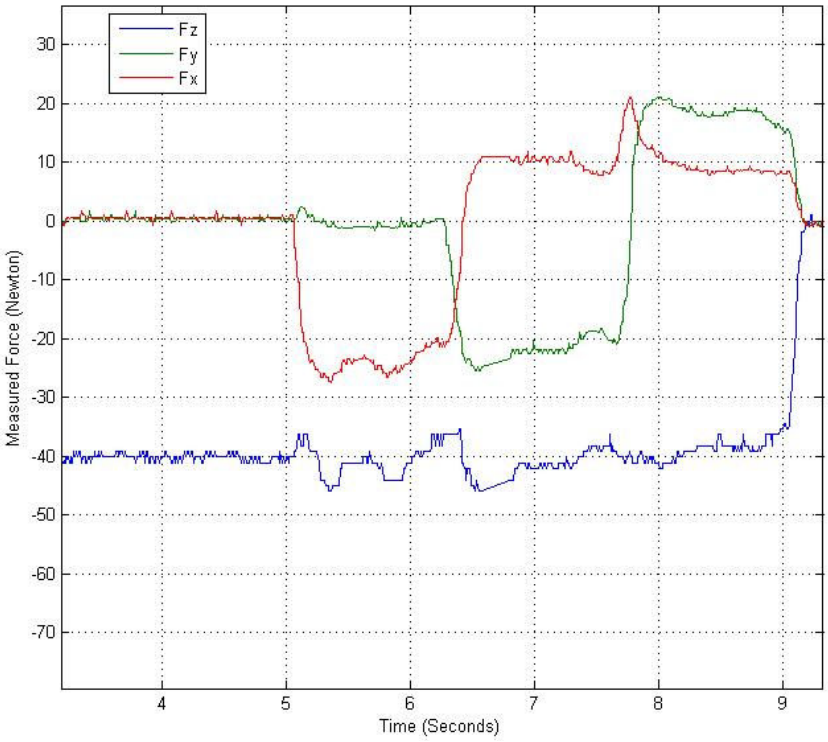

Figure 5.39 : Force measurements for triangle.

The drawing saved with the name "curl.dxf" which can also be found in Appendix A.1 is illustrated on Figure 5.40. The execution of application for this drawing can be observed by watching video "CAD Curl 01.mp4" saved in Appendix A.1. Besides, the simulation result is illustrated in Figure 5.41.

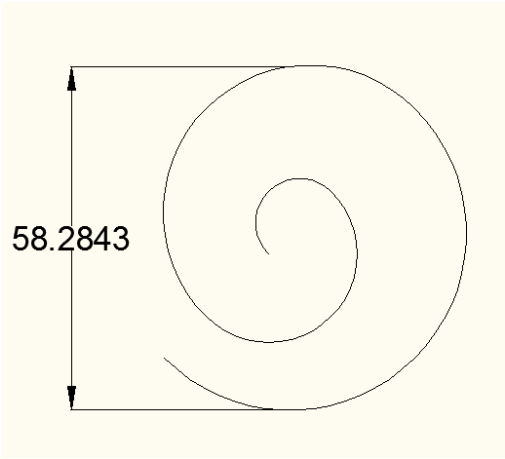

Figure 5.40 : 2D drawing of curl.

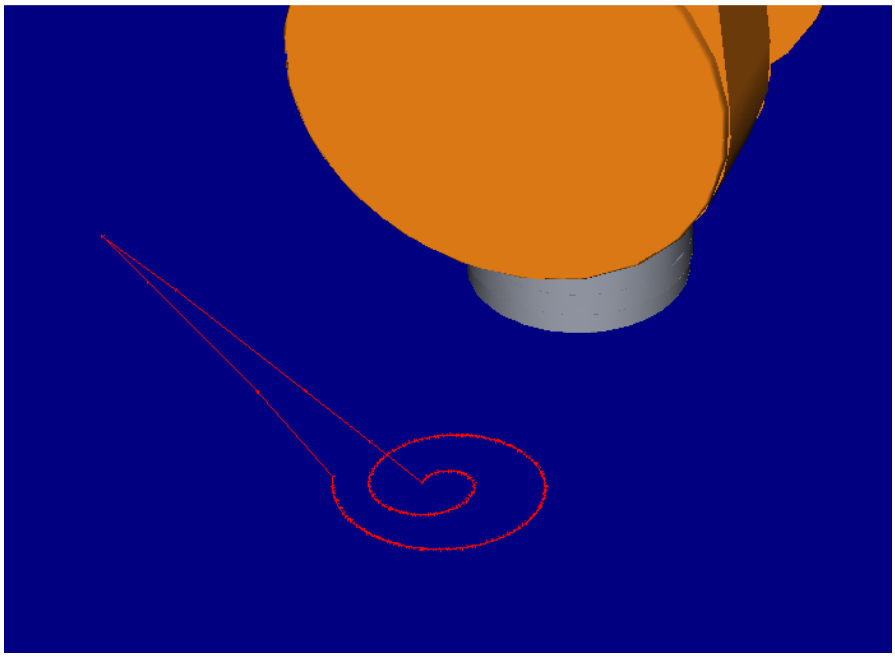

Figure 5.41 : Simulation results for curl.

The force measurements obtained from system for curl is shown on Figure 5.42. Note that reference force is 40N.

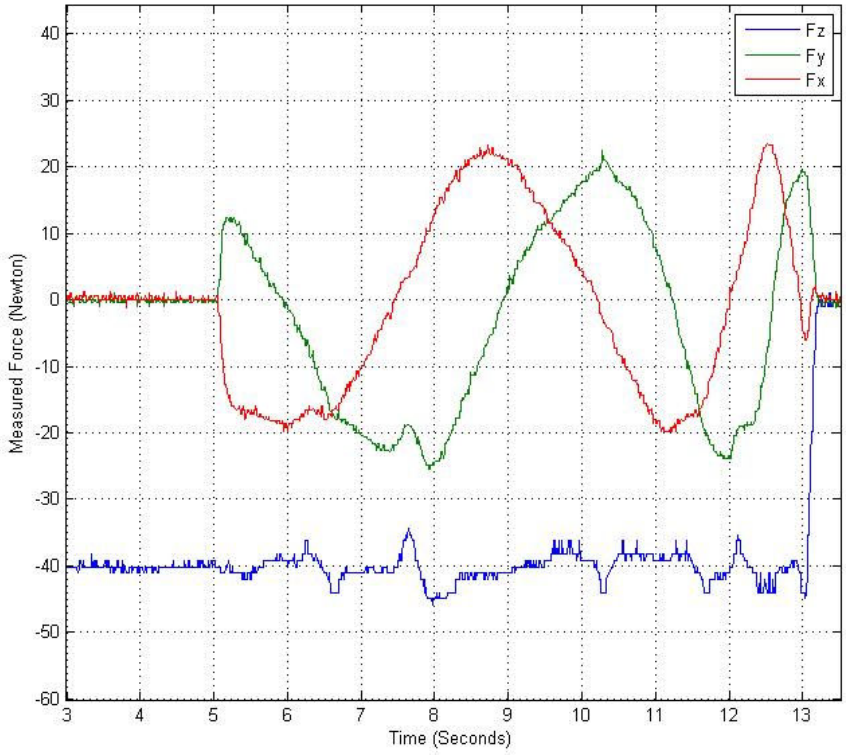

Figure 5.42 : Force measurements for curl.

5.4 Program for Coordinate Measurement and Experimental Results

In this section, application written for coordinate measurement is explained. This process is achieved by scanning a predefined path on X and Y axes while keeping a constant force on Z axis.

The surface to be scanned needs to be introduced by three points with the manual control pendant before application is run. These points are P_{start} , P_{top} and P_{side}. Defining these points and the path that will be followed after these points are defined is shown on Figure 5.43. P_{start} point is the starting point for scanning process. P_{top} point defines the highest point in X axis, and P_{side} point defines the highest point in Y axis to be scanned.

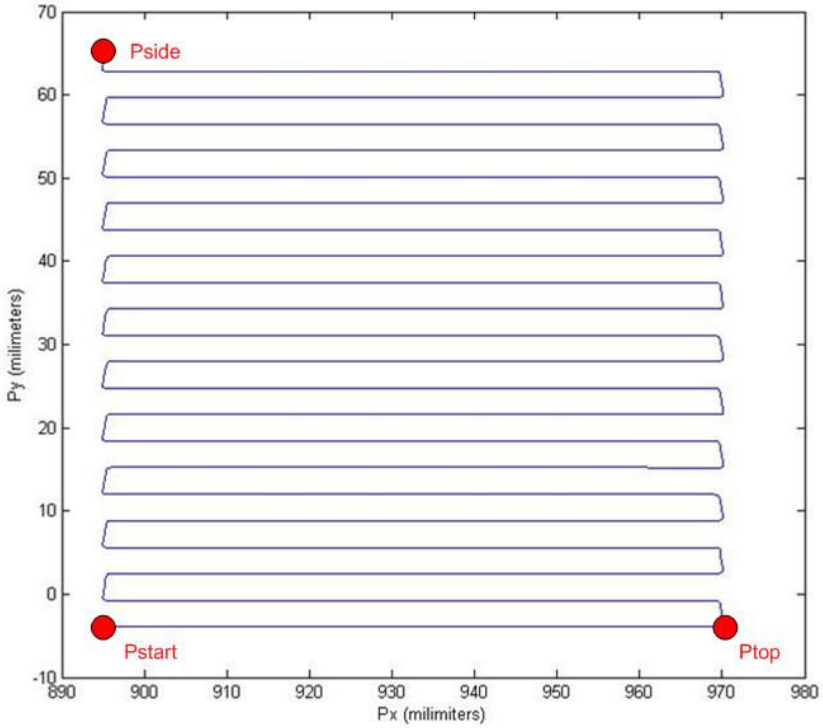

Figure 5.43 : Surface tracking pattern.

The application written for this process is saved under folder named "Coordinate Measurement" in Appendix A.1.

The application is tested on geometrical shapes whose patterns are pressed on a polystyrene block.

The circular shape to be measured is shown on Figure 5.44. On Figure 5.45 the 3D drawing of the shape which is obtained from the measurements taken is shown, this drawing is obtained by the help of MATLAB's surface fitting tool. The data gathered from arm's controller is saved in a file and imported to MATLAB workspace. The VAL3 application created automatically saves the files in a format that MATLAB can easily import data.

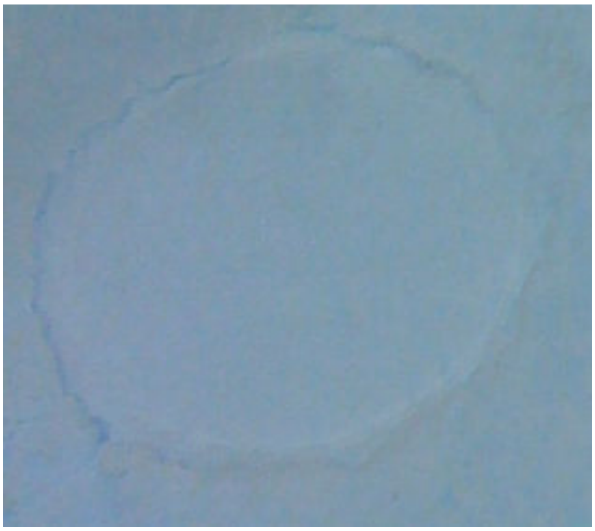

Figure 5.44 : Circular shape.

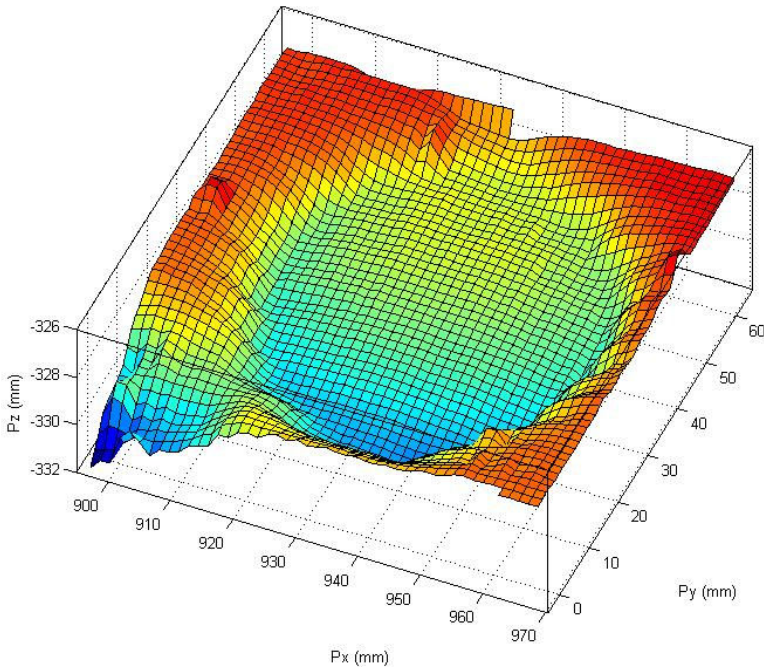

Figure 5.45 : Scan results for circular shape.

The triangular shape to be measured is shown on Figure 5.46. On Figure 5.47 the 3D drawing of the shape which is obtained from the measurements taken is shown.

Figure 5.46 : Triangular shape.

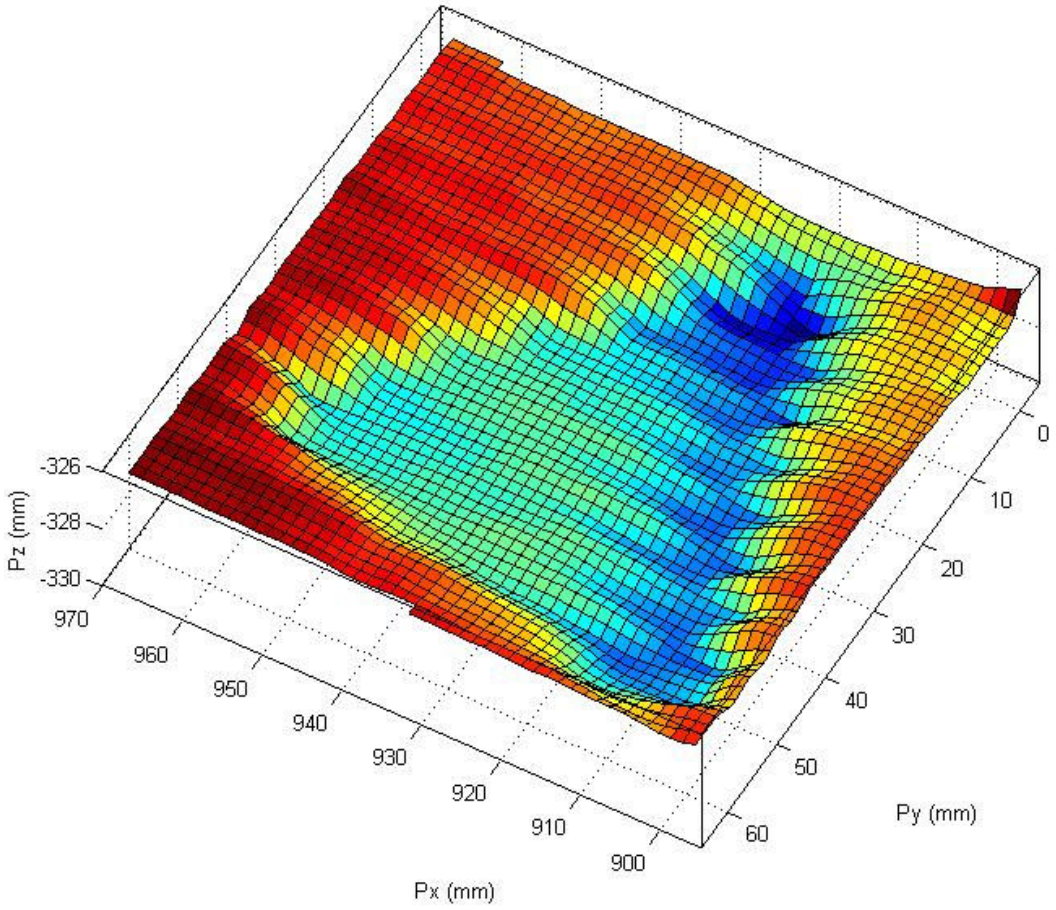

Figure 5.47 : Scan results for triangular shape.

The pentagonal shape to be measured is shown on Figure 5.48. On Figure 5.49 the 3D drawing of the shape which is obtained from the measurements taken is shown.

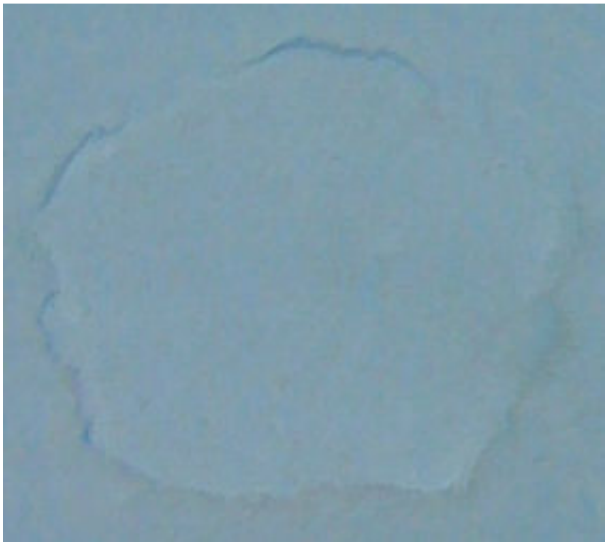

Figure 5.48 : Pentagonal shape.

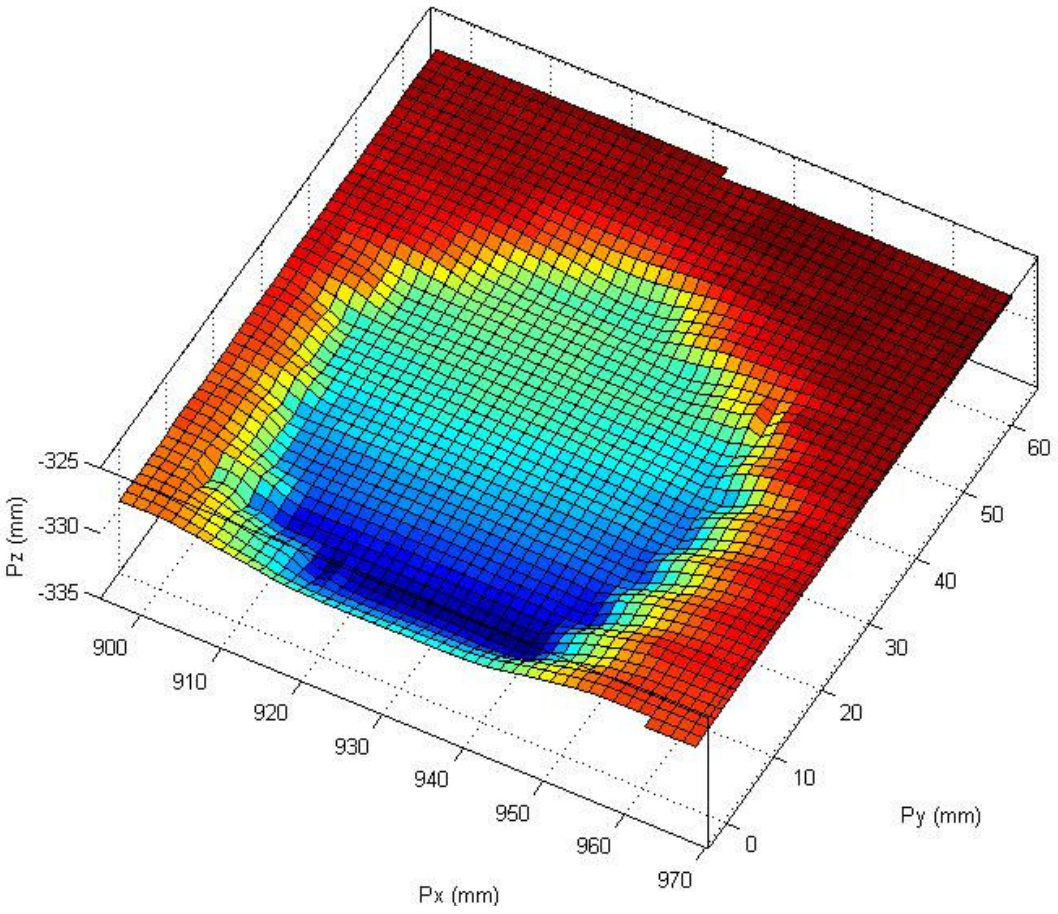

Figure 5.49 : Scanning results for pentagonal shape.

The square shape to be measured is shown on Figure 5.50. On Figure 5.51 the 3D drawing of the shape which is obtained from the measurements taken is shown.

Figure 5.50 : Square shape.

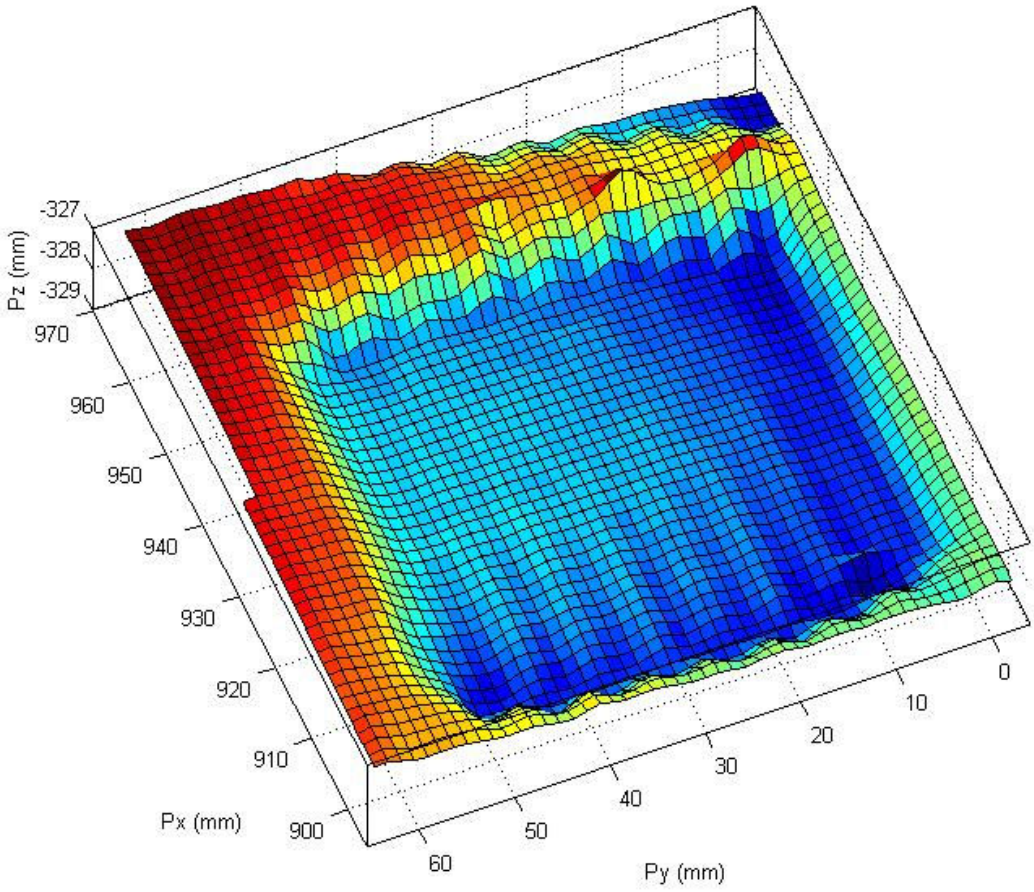

Figure 5.51 : Scanning results for square shape.

And lastly, the carvings of İTÜ and MEAM on polystyrene block are scanned using this program. The results are shown on Figure 5.52 and Figure 5.53.

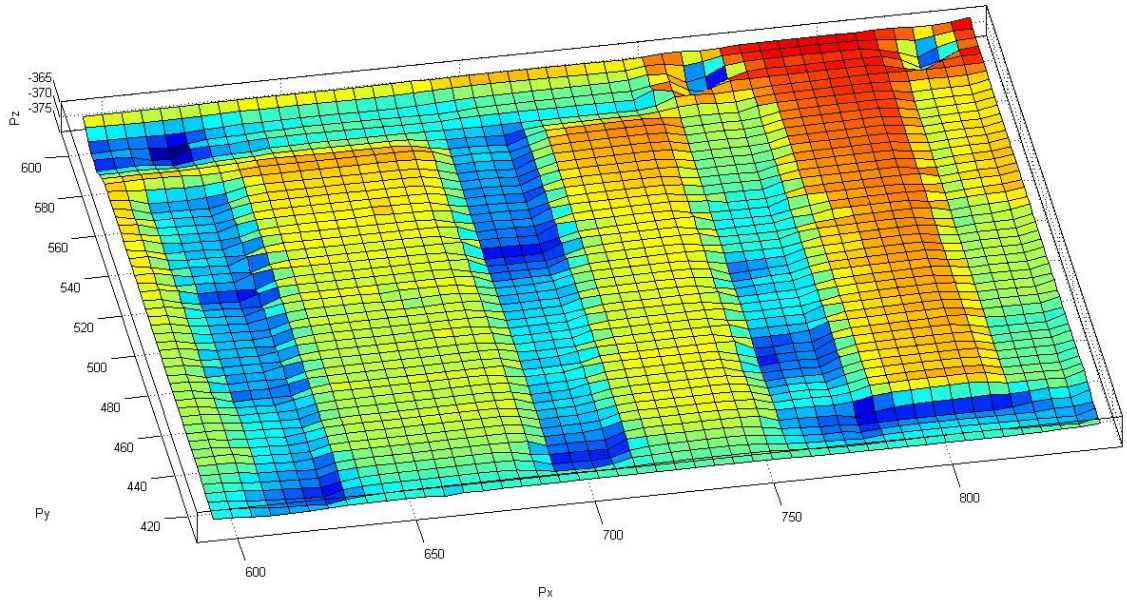

Figure 5.52 : Scanning results of İTÜ.

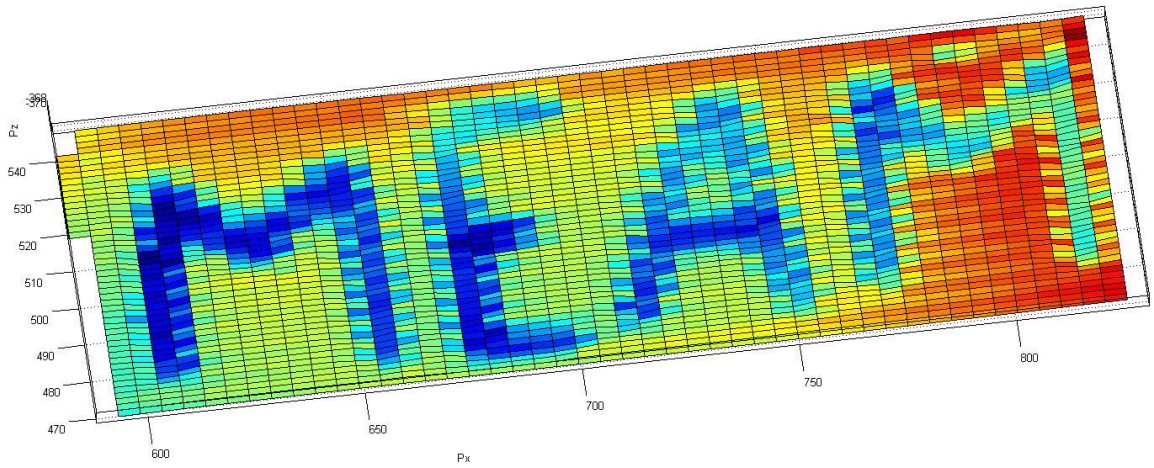

Figure 5.53 : Scanning results of MEAM.

6. CONCLUSION AND RECOMMENDATIONS

One of the major purposes of this research is to achieve force interaction of the industrial arm STAUBLI RX-160 with the environment. This aim is satisfied using the force/torque sensor mounted on wrist of the arm. In order to achieve force interaction, firstly the technical specifications of the force/torque sensor system are inspected. Then, a VAL3 program and library is written for obtaining force/torque data properly from this system. On next step, several sample probes are inspected and experimented in order to reveal which probe/tool should be used to get optimum results. Stiffness contact model is used for modeling the contact between end effector of robot and the environment. In order to achieve force control, PID and sliding mode controllers are used. These two algorithms are compared with each other using experimental results and it is observed that the use of sliding mode controller is more satisfying as it gives results with better settling times and less overshoots. After a satisfying environment is provided, many force applications like human-machine collaboration and friction measurements are experimented.

Additionally, in order to achieve complex surface tracking processes, CAD data tracking is investigated. For this purpose, creating CAD files to be used in arm's controller is explained on the first step. Next, how to import files created to arm's controller and how to read these files with VAL3 is explained. Lastly, moving the arm with respect to path defined on CAD file is inspected. After a satisfying environment is provided, sample CAD drawings are used in order to experiment the applications written.

Besides, tracking the path defined in CAD file and force control applications are combined together. After a satisfying background on both subjects is provided, force control and CAD tracking is achieved at the same time on different axes using the algorithm mentioned on Section 4.3.

The experiments and studies in this research revealed that tracking the path defined on CAD files and force interaction is possible with industrial arm RX-160. Besides, these subjects are very promising as there are limitless applications which can be produced on this base.

Finally, the VAL3 applications and control algorithms in this thesis can be developed many ways. The force control is achieved by PID and SMC; this can be developed by trying out other controllers like fuzzy logic or adaptive controllers. Besides, the probe/tool of the arm can be improved in order to provide better measurements.

REFERENCES

[1] **Siciliano, B., and Khatib, O.,** 2008. *Handbook of Robotics*, Springer-Verlag, Berlin.

[2] **Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G.,** 2009. *Robotics: Modelling, Planning and Control,* Springer, London.

[3] **Critchlow, A.,** 1985. *Introduction to Robotics,* MacMillan Press, London.

[4] **Niku , S.,** 2010. *Introduction to Robotics,* Prentice Hall, New Jersey.

[5] **Patel, R., and Talebi, H.,** 2009. A Robust Position and Force Control Strategy for 7-DOF Redundant Manipulators*, IEEE/ASME Transactions on Mechatronics,* Vol. **14**, No. 5, pp. 575-589.

[6] **Marrone, F., Raimondi, F., and Strobel, M.,** 2002. Compliant Interaction of a Domestic Service Robot with a Human and the Environment, *Proceedings of the 33rd* ISR, Stockholm, October 7-11.

[7] **Lopes, A., and Almeida, F.,** 2008. A force-impedance controlled industrial robot using an active robotic auxiliary device, *Robotics and Computer Integrated Manufacturing*, Vol. **24**, No. 3, pp. 299-309.

[8] **Babaci, S., Amirat, Y., Pontnau, J., and François, C.,** 1996. Fuzzy Adaptation Impedance of a 6 D.O.F Paralel Robot: Application to Peg in Hole Insertion, *Proceedings of the Fifth IEEE International Conference*, New Orleans, September 8-11.

[9] **Krüger, J., and Surdilovic, D.,** 2008. Robust Control of Force-Coupled Human-Robot-Interaction in Assembly Processes, *CIRP Annals – Manufacturing Technology,* Vol. **57**, No. 1, pp. 41-44.

[10] **Kumar, R., Berkelman, P., Gupta, P., and Barnes, A.,** 2000. Preliminary Experiments in Cooperative Human/Robot Force Control for Robot Assisted Microsurgical Manipulation, *Proceedings of the IEEE International Conference on Robotics and Automation,* San Francisco, April 24-28.

[11] **Wagner, A., Pott, P., Schwarz, M., and Scharf, H.,** 2009. Control of a Handheld Robot for Orthopedic Surgery, *World Congress on Medical Physics and Biomedical Engineering*, Vol. **25**, No. 6, pp. 99-102.

[12] **Jayender, J., Patel, R., and Nikumb, S.,** 2006. Robot-Assisted Catheter Insertion using Hybrid Impedance Control, *Proceedings of the IEEE International Conference on Robotics and Automation,* Orlando, May 15-19.

[13] **Nagata, F., and Watanabe, K.,** 2000. Teaching System for a Polishing Robot Using a Game Joystick*, Proceedings of the SICE Annual Conference,* Iizuka, July 26-28.

[14] **Park, J., Kim, H., and Kim, S.,** 2008. Active Compliant Motion Control for Grinding Robot, *Proceedings of the International Federation of Automatic Control,* Seoul, July 6-11.

[15] **Pulkkinen, T., Heikkila, T., Sallinen, M., Kivikunnas, S., and Salmi, T.,** 2008. 2D CAD based robot programming for processing metal profiles in short series manufacturing, *International Conference on Control, Automation and Systems,* Seoul, October 14-17.

[16] **Nagata, F., Hase, T., Haga, Z., Omoto, M., and Watanbe, K.,** 2007. CAD/CAM- based position/force controller for a mold polishing robot, *Mechatronics,* Vol. **17**, No. 5, pp. 207-216.

[17] **Neto, P., Pires, J., and Moreira, A.,** 2010. CAD-Based Off-Line Robot Programming, *Robotics Automation and Mechatronics IEEE Conference,* Singapore, June 28-30.

[18] **Url-1 <***http://www.staubli.com/en/robotics/***>**, accessed at 11.06.2010.

[19] **Url-2 <***http://www.ati-ia.com/products/ft/sensors/***>**, accessed at 11.06.2010.

[20] **Craig, J.,** 2005. *Introduction to Robotics*, Pearson Prentice Hall, New Jersey.

[21] **Lee, D., and ElMaraghy, W.,** 1990. ROBOSIM: a CAD-based off-line Programming and Analysis System for Robotic Manipulators, *Computer-Aided Engineering Journal*, Vol. **7**, No. 5, pp. 141-148.

[22] **Ogata, K.,** 2002. *Modern Control Engineering,* Prentice Hall, New Jersey.

[23] **Bennett, S.,** 1993. *A History of Control Engineering 1930-1955*, IET, London.

APPENDICES

APPENDIX A.1: Application Pictures, Videos, Drawings and Program Codes.

CURRICULUM VITAE

Candidate's full name:	Okan TÜRKMEN
Place and date of birth:	Şişli, 22.02.1984
Permanent Address:	Barbaros Mahallesi Gül Sokak Ağaoğlu My World Starland Sitesi E-1/46 Batı Ataşehir İstanbul
Universities Attended:	Istanbul Technical University, Control Engineering (Undergraduate 2002-2006)