Robot Contact Motion Control Using Environment Impedance Identification - Application of Neural Networks

by

Kazuo Kiguchi

A thesis submitted to the
University of Ottawa
in partial fulfilment of the requirement
for the degree of
Master of Applied Science
in
Mechanical Engineering

Department of Mechanical Engineering
University of Ottawa

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-315-89643-4

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Abstract

The robot manipulators must have capability of controlling mechanical interaction with objects which are involved in various tasks. In order to control the mechanical interaction, the environment impedance which is not always known has to be identified. In this thesis, environment impedance on-line identification methods based on Recursive Least Squares Estimation and Neural Networks approach are investigated in order to achieve an efficient force control. Experiments were done for both methods with the 3-DOF Direct-Drive planer robot manipulator, by applying a force on the environment with a robot manipulator and measuring the force at the same time with a force sensor. The hybrid controller is used for controlling the robot for environment impedance identification.

Currently, neural networks applications to Robotics have been studied in order to apply their ability of acquiring and storing experiential data and adjusting robot controllers to new environments. Although there are several types of neural networks robot controllers, the direct neural networks robot controller seems the most effective application of neural networks to robot controllers. It is known that the nonlinear neural networks are able to realize the nonlinear effects of the robot in the networks. This ability is very useful for controlling robot manipulators. Generally, the neural networks of the PDP model can only realize static mapping by itself and can not incorporate robot dynamics which is essential for an advanced controller. In order to realize dynamic mapping, past input/output information of the robot is used as input signals for the back-propagation type of the Neural Networks.

In this thesis, the nonlinear three-layer neural networks feedforward direct controller using a dynamic back-propagation method based on past input/output information of the plant is also proposed. This controller is used for a 3-DOF robot applying force to the environment. The neural networks controller controls the position (along the constraint surface) and the angle of the end-effector as well as the force applied to the environment. The generalized delta law is used as a learning law to adjust the weights of the neural networks at every sampling time. This proposed controller is experimentally tested with the 3-DOF Direct-Drive planar robot manipulator. The experimental results show the effectiveness and flexibility of the neural networks controller for robots.

Acknowledgements

I wish to express deep gratitude to my supervisor, Prof. Necsulescu, for his guidance and valuable suggestions. Also, I would like to thank all my colleagues for their encouragement and help. I am particularly grateful to Rahim Jassemi, Jean de Carufel and Bumsoo Kim for their cooperation and suggestions.

I am also very grateful to my parents, my wife Mika, my daughter Hanaka, and my son Sanetomo for their support and cooperation throughout my education.

Contents

Abstract ii

Acknowledgements iv

Nomenclature vii

1. Introduction 1

2. Literature Review 5

2.1. Force Control 5

2.2. Neural Networks for Control and Identification 7

2.3. Environment Impedance Identification 8

3. Experimental Set-Up 10

4. Control of the Robot 17

4.1. Relation between Joint Torque and Cartesian Force 18

4.2. Dynamics of the Robot 19

4.3. Position Control 20

4.4. Hybrid Control 21

4.5. Neural Networks Control 25

5. Environment Impedance Identification 31

5.1. Application of Least Square Estimation 32

5.2. Application of Neural Networks 34

6. Experimental Results 37

6.1. Identification using Least Square Estimation Method 38

6.2. Identification using Neural Networks 45

6.3. Force Control by the Neural Networks Controller 50

7. Conclusions 58

References 60

Appendix A: Manipulator Link Dimensions 65

Appendix B: Specification of Motors 70

Appendix C: Manipulator Kinematics 72

Appendix D: Manipulator Dynamics 77

Appendix E: Computer Code of Neural Networks Controller 90

Nomenclature

f - force vector measured by the force sensor

fd - desired force vector

q - angular position vector

x - Cartesian position vector

xd - desired Cartesian position vector

x - x position in Cartesian coordinates

y - y position in Cartesian coordinates

ϕ - ϕ position in Cartesian coordinates

xd - desired x position in Cartesian coordinates

yd - desired y position in Cartesian coordinates

ϕd - desired ϕ position in Cartesian coordinates

τ - command torque vector for motors

τ_h - command torque vector from hybrid control part of the controller

τ_n - command torque vector for motors from the neural network controller

I - identity matrix

S - diagonal selection matrix of ones and zeros for hybrid control

M - inertia matrix

h - vector of Coriolis and centrifugal terms

g - gravity term vector

F_v - viscous friction term vector

F_c - Coulomb friction term vector

J - Jacobian matrix

K_{pp} - position gain for position control

K_{pv} - velocity gain for position control

K_{fp} - position gain for force control

K_{fi} - integration gain for force control

u_x - command vector for position control

u_f - command vector for force control

w_{ij} - weight of the signal from i neuron to j neuron

η - learning rate of neural network

M_e - inertia coefficient of the environment

B_e - damping coefficient of the environment

K_e - spring coefficient of the environment

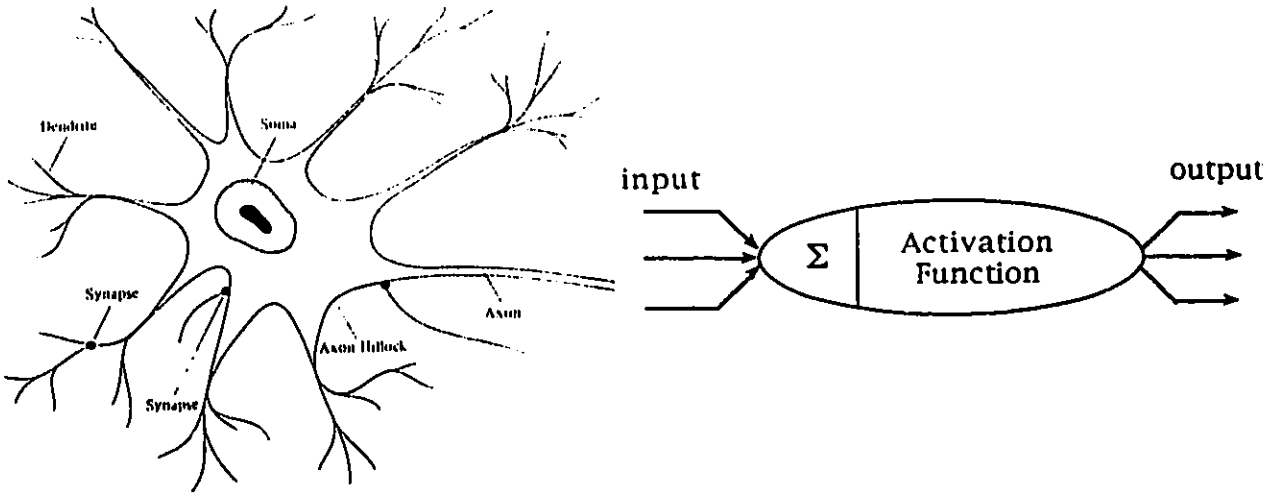
x_e - surface displacement of the environment

1. Introduction

Robots are expected to perform simple as well as complex applications. In achieving some sophisticated tasks such as grinding, sanding, assembling, opening or closing a door, window cleaning, etc., the robots are obliged to interact with the environment. In such cases, the environment impedance should be known for Cartesian contact force control with the environment. It is not, however, always possible to know the environment impedance before applying the force control. This is one of the reasons that the application of force control is far behind that of the position control. If the environment impedance is identified, we can use it to tune the Cartesian force control law or decide the parameters for impedance control for better control. Friction and backlash of transmission systems also cause difficulty to force control. Backlash of transmission systems and its friction can be removed by using Direct-Drive motors since transmission systems are not required in this case. The other friction of the system and nonlinearities such as centrifugal and Coriolis forces must be compensated in the control law. Stability analysis is one of important topics in robotics in order to understand the domain of applicability of the proposed controller. This topic is not included, however, in this thesis which focuses on the feasibility of some new methods for contact motion control. The stability will have, however, to be analysed in a future work in order to complete the theoretical analysis.

A neural network (this means an artificial neural network in this thesis) is motivated by the research of the human brain and its nervous system. Neural network consist of many synapses and neurons. The human brain contains roughly

10^{11} neurons and 10^{15} synapses. Neural Networks are able to store pattern or function information in themselves. There are many types of neural network architectures and learning laws. Although neural networks were studied for a long time since McCulloch and Pitts first proposed a neuron model in 1943 [1], they are still subject of research because the understanding of biological neural networks is not yet developed enough. Figure 1.1 shows a biological neuron model and an artificial neuron model. Currently, neural networks applications to Robotics have been studied in order to apply neural networks' ability of acquiring and storing experiential knowledge and adapting to new environment. In Robotics, neural networks can be used for realization of forward or inverse kinematics and dynamics, control and identification of the system. The author applies neural networks to environment impedance identification and robot control.



(a) Biological Neuron Model
(from B. Kosko 'Neural Networks and Fuzzy Systems')

(b) Artifitial Neuron Model

Fig.1.1 Neuron Models

There are mainly two contributions in this thesis. First, two kinds of environment impedance identification methods are proposed. If we assume the environment is not rigit, we can model it as a mass, a damper, and a spring. Environment impedance identification means to identify these coefficients respectively. One of these proposed methods is using the recursive least-square

estimation. The basic idea is minimising the error between the measured force and the force which is calculated from estimated coefficients. The other is using neural networks. The idea is reducing the error between the measured force and the force which is estimated by neural networks. The rate of error reducing, which is called a learning rate, must be chosen to be proper small value. If the learning rate is too big, we will overshoot the ideal position and the error will even be increased. If the learning rate is too small, it takes time to minimize the error. Both environment impedance identification methods are experimentally tested by applying a force using the hybrid position/force control on the environment with a 3-DOF direct-drive planar robot manipulator. The hybrid position/force control is applied to control the position (along the constraint surface) and the angle of the end-effector as well as the force to an unknown environment. The environment is pushed using the manipulator arm and the environment impedance is identified by the recursive least-squares estimation or neural networks using reaction force of the environment. Styrofoam and rubber are used for the environment. The experimental results show that both methods are feasible. However, since the hybrid position/force controller can not be applied to an unknown environment with several reasons, one of the adaptive controllers must be used for the unknown environment. Therefore, as a second contribution of this thesis, the neural networks controller for multi-DOF robots is proposed in order to perform force control to the unknown environment. The originality of this neural networks controller is that the neural networks is combined with the hybrid position/force controller. This neural networks controller is also experimentally tested with the 3-DOF direct-drive planar robot manipulator. Styrofoam and rubber are used for the unknown environment. The experimental results show that we can obtain desired force response with the unknown environment by using the proposed neural network controller.

Recursive Newton-Euler formulation is used for calculating the dynamics of the robot since Lagrangian formulation needs more calculation than Recursive Newton-Euler formulation [2]. Since the robot is a planar robot, the effect of gravity

is ignored. Nonlinearities of the robot dynamics are compensated in the controllers.

2. Literature Review

2.1. Force Control

There are several types of force control approach, namely, stiffness control, damping control, impedance control, resolved acceleration control and hybrid control. Force control can be mainly categorized into two groups, hybrid control (hybrid position/force control) and impedance control. The hybrid position/force control was proposed by Raibert and Craig in 1981 [3]. The basic idea is separating direction for force control that is normal to the constraint surface and for position control that along the constraint surface in Cartesian coordinate system. Then calculated Cartesian forces is transfered to joint torques. Their controller block-diagram is shown in Figure 2.1. Similar idea can be seen in Mason's work [4]. However, since their control algorithm has some problems [5][6] and does not include dynamics of the robot, it has been further developed by some other researchers [7][8]. In this thesis, this

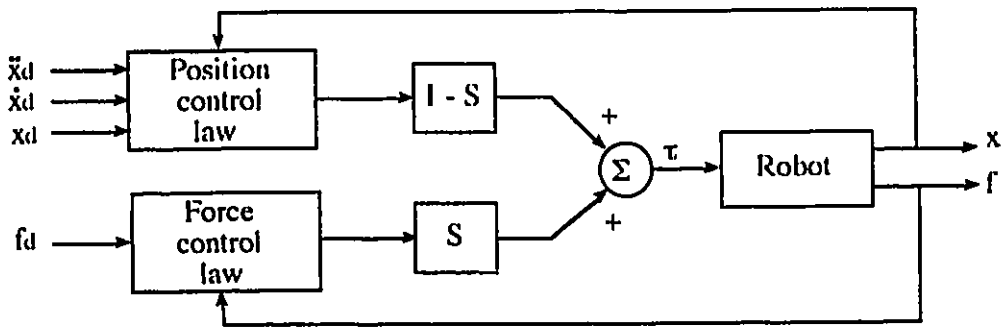


Fig.2.1 Hybrid position/force controller

modified hybrid position/force control was used for the robot control for environment impedance identification. The modified hybrid position/force control is also used for the part of the neural network controller.

The stability of the hybrid control has also been studied, using Liapunov method [9-11]. This topic, however, is not included in this thesis.

Impedance control is an expansion of the stiffness control algorithm [12] and of damping control [13] since the impedance is equivalent to spring-damper system. This control algorithm is proposed by Hogan in 1985 [14]. In this algorithm, the robot's task fundamentally is described not in terms of motions, nor in terms of forces, but in terms of the relation between them.

Suppose that the end-effector is moved from a desired trajectory by an external force f.

$$\Delta x(t) = x_d(t) - x(t) \tag{2.1}$$

In the impedance control, the relation between force and displacement, is controlled to satisfy following equation.

$$M\Delta\ddot{x} + B\Delta\dot{x} + K\Delta x = f \tag{2.2}$$

So that acceleration command becomes

$$\ddot{x} = \ddot{x}_d + M^{-1}B\Delta\dot{x} + M^{-1}K\Delta x - M^{-1}f \tag{2.3}$$

The model of the impedance control is shown in Figure 2.2.

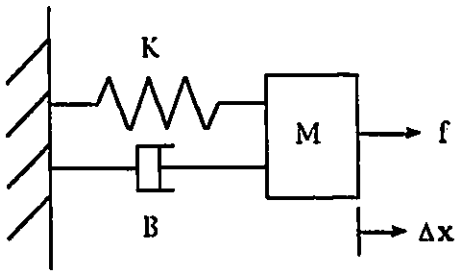


Fig.2.2 The model of the impedance control

2.2. Neural Networks for Control and Identification

Application of neural networks to control is first stated by Widrow [15] in 1960s. Afterwards, Kawato et al. first applied neural networks to robot control [16][17] in 1980s, and later many researchers have applied neural networks to robot control in various ways. Narendra and Parthasarathy proposed identification and control of dynamical systems using neural networks and showed simulation results [18]. They discussed static and dynamic back-propagation methods for the adjustment of parameters. In this thesis, one of the neural networks is used not for system idenfitication but for environment impedance identification.

The identification using neural networks is implicit method of identification. The neural networks, however, are used for explict identification method in this thesis. Yabuta and Yamada proposed a neural networks force controller for a 1DOF robot [19][20] and tested it experimentally. The architecture of their neural networks controller is shown in Figure 2.3 and the block diagram of their system is shown in Figure 2.4. The neural networks controller proposed in this thesis is extention of this controller. This controller is extended for Cartesian space control of 3-DOF robot manipulator and combined with the hybrid controller. They studied also the

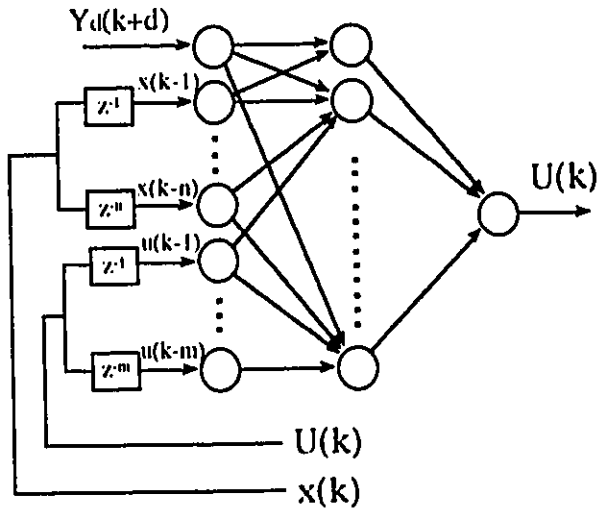


Fig.2.3 The architecture of the neural network controller

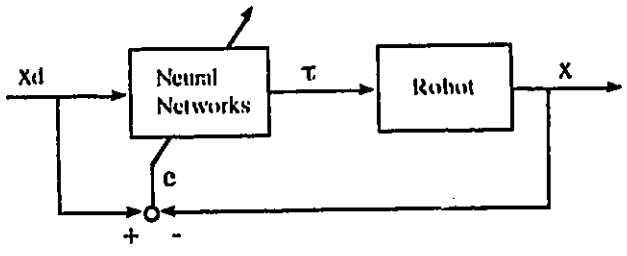


Fig.2.4 Direct Controller

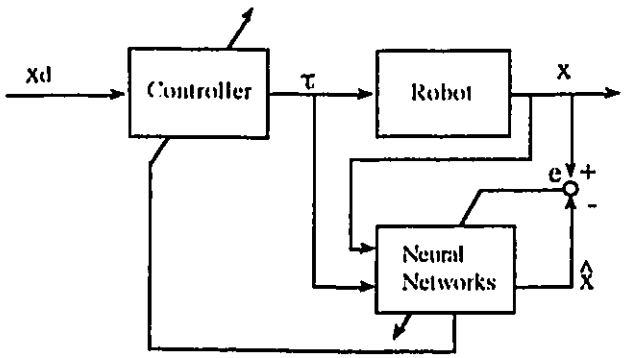


Fig.2.5 Indirect Controller

stability of the linear neural network controller using a Lyapunov function [21]. Fukuda et al. applied neural networks for hybrid position/force controller in the different way [22-24]. In this controller, neural networks self-tune PID parameters, as shown in Figure 2.5. Their simulation results and experimental results show that the controller is robust to change of the environment dynamics and the configuration of the manipulator. Furthermore, they used neural networks to control a manipulator in order to approach the environ-ment efficiently and to recognize the contact with the environment.

The neural networks controller proposed in this thesis is cobinationation of the hybrid controller and neural networks. This type of neural networks controller is not yet proposed in the literature.

2.3. Environment Impedance Identification

In order to apply efficiently force control to an unknown environment, various methods are used to identify environment impedance explicitly or implicitly. Xu et al. [25] proposed fuzzy compensation to identify an unknown environment. Gomi and Kawato [26] used neural networks for impedance

control to compensate for an unknown environment. Cohen and Flash [27] also used neural networks as a method of learning impedance control parameters. These methods mentioned above are implicit environment impedance idenfitication. For explicit environment impedance identification method, Kalaycioglu and Brown [32] used Continuous Least Square Algorithm to identify unkown environment impedance parameters and showed simulation results. Their method is used as one of the identification methods in this thesis and tested experimentally. The method of environment impedance identification using neural network, which is proposed in this thesis, is not yet used in the proposed form.

3. Experimental Set-Up

The experimental set-up consists of a 3-DOF Direct-Drive robot manipulator, motor drivers, a DSP based controller and a force sensor. Figure 3.1 shows configuration of 3-DOF Direct-Drive robot manipulator. The manipulator links are made of aluminum. The manipulator geometry is given in Appendix A. Figure 3.2 shows the experimental set-up. There are different type of motors in each manipulator joint. The motor1 is NSK 0810 (resolution 614,400 step/rev), the motor2 is NSK 0408 (resolution 409,600 step/rev) and the motor3 is Parker Hannifin System7 7521.250 (resolution 2,000 step/rev). The specification of motors is given in Appendix B. The dSPACE is a development and real time control system based on a DSP (TMS320C30). TMS320C30, which is located on DS1002 processor board component of the dSPACE (see Figure 3.3), is a CMOS 32-bit floating device in the TMS320 family of single-chip DSPs [35]. A

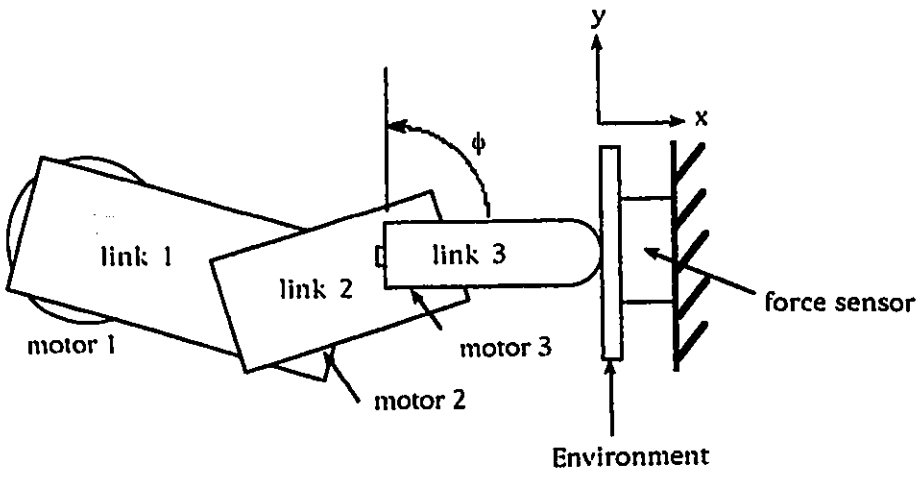


Fig.3.1 Configuration of the robot manipulator

DSP - dSPACE

Motor 1 - NSK Megatorque Motor System 0810 (Max. 88.2 Nm)

Motor 2 - NSK Megatorque Motor System 0408 (Max. 9.8 Nm)

Motor 3 - Parker Hannifin Co. System7 Type 7521.250 (Max. 2 Nm)

Force Sensor - Barry Wright Co. 6-axis force sensor FS6-120A

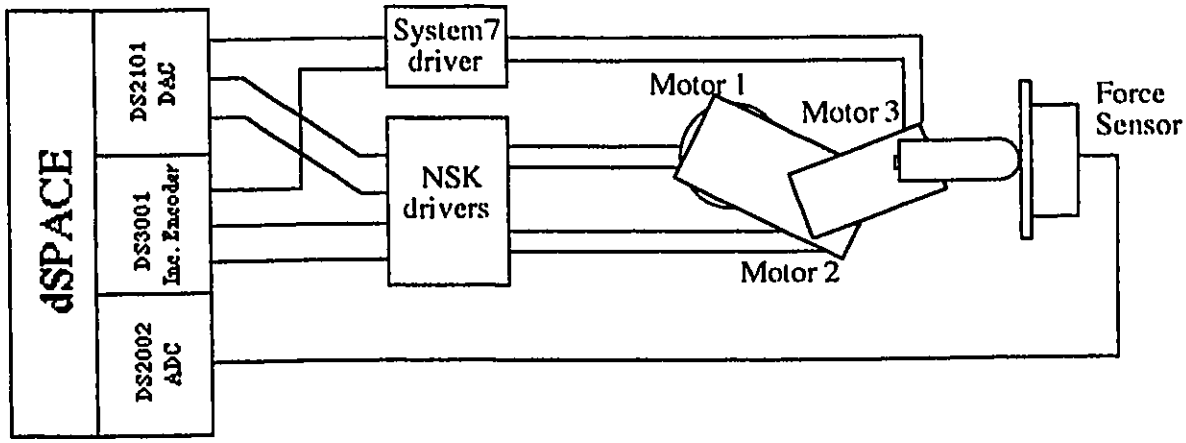


Fig.3.2 Experimental Set-Up

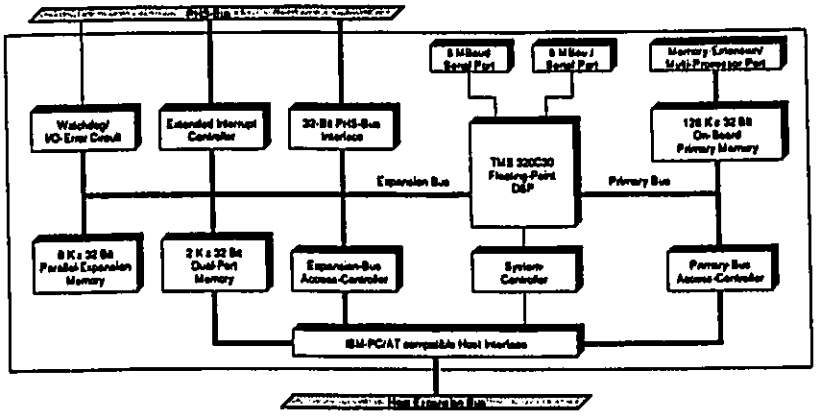


Fig. 3.3 Block diagram of the DS1002
(from dSPACE 'DS1002 User's Guide')

total memory space of this DSP is 16M 32-bit words including program, data and I/O space. The cycle time of the TMS320C30 is 60 nsec. This fast cycle time allows it to execute operations up to 33MFLOPS. The dSPACE contains ADC (DS2002), DAC (DS2101) and incremental encoder (DS3001) [36]. The conversion time of the ADC is 5μs for 16-bit resolution. The full scale settling time of DAC to 0.01% is 3μs. All peripheral boards like ADC or DAC are connected to the DS1002 by using the PHS-bus. The PHS-bus is a high-speed, bidirectional synchronous bus that allows direct I/O operations between DSP and peripheral boards without host intervention. The PHS-bus supports 32-bit parallel transfers and is designed for a maximum transfer rate of 26.7 Mbyte/s (150nsec/32-bit transfer). On the DS1002 this bandwidth is limited by the timing of the TMS320C30 to 16.6 Mbyte/s.

The torque commands of the controller installed in dSPACE are sent to NSK motor drivers and System7 driver. The drivers provide the power for motor motion supplied in accordance to the control command. The force sensor (FS6-120A) sends feedback signals to dSPACE.

FS6-120A is 6-axis force sensor uses 6 strain-gage transducers to measure the 6 force components (3 forces, 3 moments). The process in the force sensor is shown in Figure 3.4. Each strain-gage bridgesignal is amplified and passed through a 120Hz low-pass filter so that the bandwidth is appropriately limited for sampling purposes. These signals are then multiplexed under control of onboard processor for conversion by the 12-bit A/D converter. The processor then multiplies the 6 sensor readings by a calibration matrix to decouple the forces into the 6 linearly independent orthogonal components and outputs the data through either the RS-232 output port or the analog output port [37]. The analog output mode is chosen in the experiment.

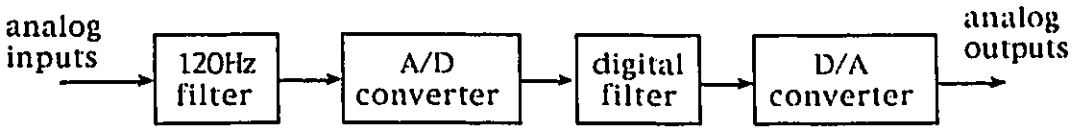


Fig.3.4 Process in the force sensor

Limitation of the bandwidth of the output data from the force sensor is important consideration. Undersampling the analog signal cause unwanted aliasing problem. In order to avoid this, the sampling rate must be greater than twice the signal bandwidth. This force sensor's maximum internal sampling rate is 480Hz (the process loop cycle time is 2.08msec). The 6 analog signals (only x-direction force signal in Figure 3.1 is used for this experiment) are filtered individually by a third order low-pass filter at 120Hz before sampling to the force sensor. Then the signals are converted from analog signals to digital signals by A/D converter at 480Hz even though analog signal output mode is chosen. The force sensor filters digital signals (bypass mode is chosen in the experiment) and converts to analog signals again through D/A converter at 480Hz. Therefore analog signals from the sensor includes noise caused by D/A converting around 480Hz. This noise might cause aliasing problem since the sampling rate used in this experiment is 1000Hz. Because the noise is very close to half of the sampling rate. In order to get rid of the noise, a low-pass filter is required. The zero-order holding, which has the characteristics of a low-pass filter, is used for sampling the signal to dSPACE. The noise, however, is not enough reduced with this. Therefore an analog low-pass filter should be used to get rid of this noise before sampling these signals in the dSPACE to avoid aliasing problem.

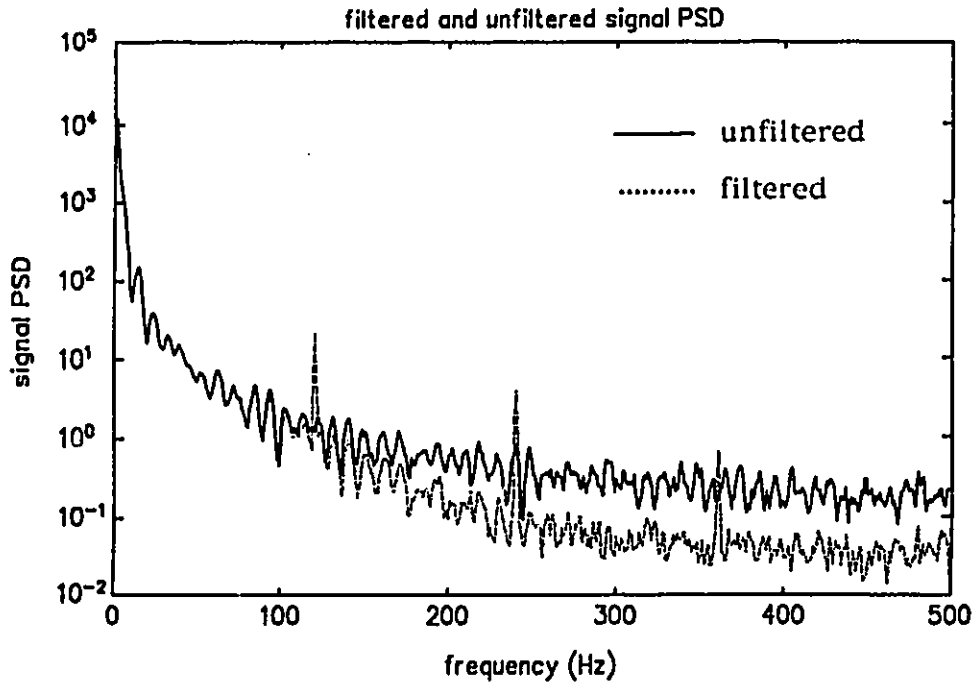


Fig.3.5 Filtered and Unfiltered signals PSD

Experiment done in this thesis does not use analog filter before sampling to the dSPACE. In order to prove the bandwidth (120Hz) of the force sensor is not affected aliasing, unfiltered signals and filtered signals to dSPACE are compared experimentally. Figure 3.5 shows signal Power Spectrum Density of unfiltered signals and filtered signals. We can see the signals are almost same under 120Hz. That means the bandwidth output signals is not affected by aliasing. Therefore there is no problem to use unfiltered signals for experiment. The spikes of filtered signals are caused by excited internal dynamics of the filter by 480Hz.

The 2nd-order low-pass filter used for this signal comparison is designed to cut-off frequency at 190Hz. Figure 3.6 shows the architecture of the filter [38].

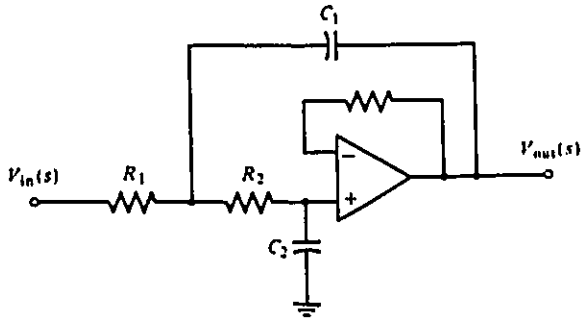


Fig.3.6 2nd-order Low-Pass Filter

The transfer function of the filter is written as

$$T_{LP}(s) = \frac{1/R_1R_2C_1C_2}{s^2 + (R_1+R_2)/R_1R_2C_1\ s + (1/R_1R_2C_1C_2)}$$

where

$R_1 = 100\Omega$, $R_2 = 1.5\Omega$, $C_1 = 470\mu F$, $C_2 = 10\mu F$

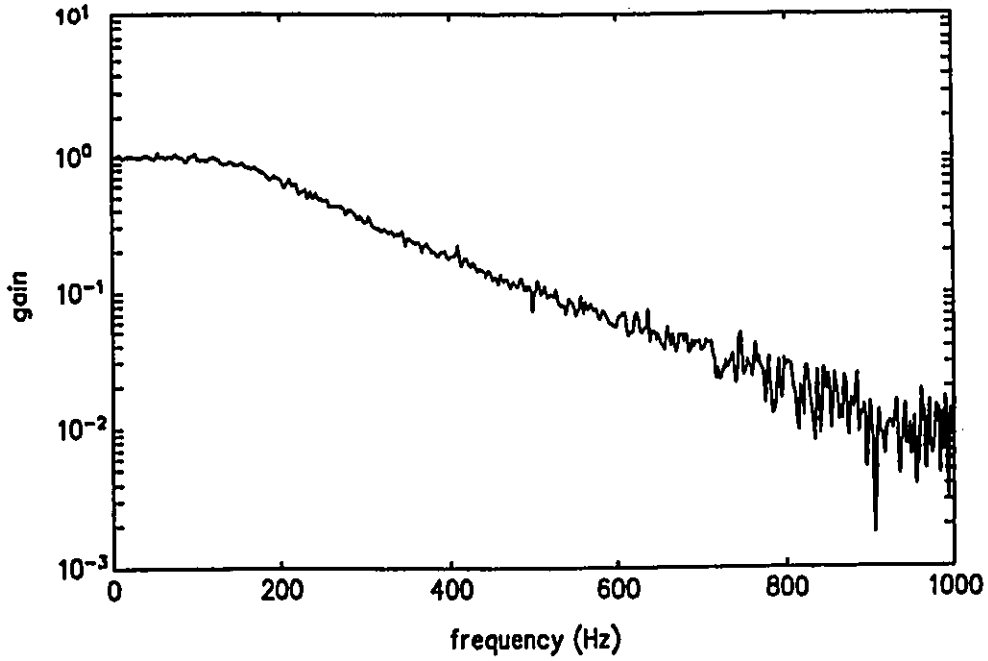


Fig.3.7 Gain Transfer Function of the Low-Pass Filter (190Hz)

Figure 3.7 shows the gain transfer function of the filter. We can see the filter cut-off frequency at 190Hz as we designed.

4. Control of the Robot

In order to control the force applied to the environment, y-position have to be controlled to keep in the same position and the angle of the end-effecter have to be controlled to keep perpendicular to the environment surface (Figure 4.1). In order to meet these requirement, hybrid position/force controller seems to be proper controller. However, since the robot arm is not initially at the desired position before hybrid control signals are sent to the motors, only position control should be applied to the arm to approach the environment. During position control, the manipulator is controlled to move 1mm every second. For position control, the resolved acceleration method is used. When the measured force (applied force) reaches 5N, automatically the hybrid position/force control is applied instead of only position control. Since the noise of the force sensor is almost ±2N and the evidence of contact is not clear when the applied force is low, the position where the applied force reac-

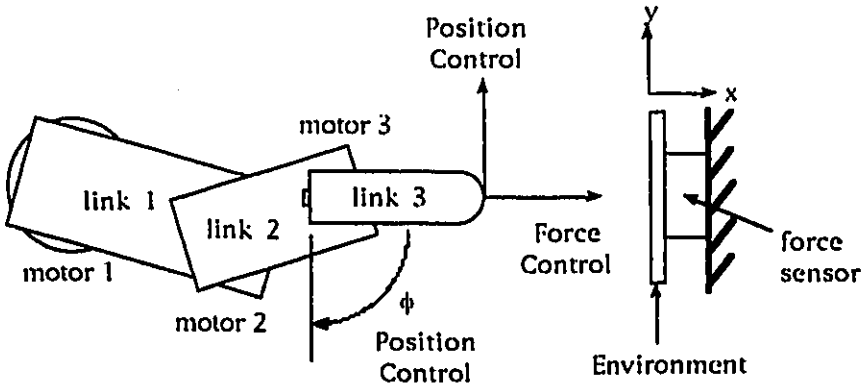


Fig.4.1 Control of the robot manipulator

ches SN is thought to be proper position for switching. Nonlinear terms like inertia, centrifugal, Coriolis and friction terms must be compensated for this control approach. In this chapter, the control algorithm of the robot is explained in detail.

4.1. Relations between Joint Torque and Cartesian Force

Suppose that the position of the end-effector is described as

$$\mathbf{x} = R(q) \tag{4.1}$$

where $\mathbf{x} = [x,y,\phi]^T$ is end-effector position in Cartesian coordinates and $q = [q1,q2,q3]^T$ is joint position in each link and direct-drive motor shaft (generalized coordinates).

From the principle of virtual work

$$\tau^T \delta q - f^T \delta x = 0 \tag{4.2}$$

where δq and δx are virtual displacement of q and x respectively. The definition of the Jacobian is

$$\delta x = J \delta q \tag{4.3}$$

Substituting into (4.2)

$$\tau^T \delta q - f^T J \delta q = (\tau - J^T f)\delta q = 0 \tag{4.4}$$

In equation (4.4) δq can take any value, such that we obtain

$$\tau = J^T f \tag{4.5}$$

This equation describes the relationship between joint torques and Cartesian forces.

4.2. Dynamics of the Robot

The dynamics equation of the robot can be written as [25].

$$M(q)\ddot{q} + h(q,\dot{q}) + g(q) + F_v\dot{q} + F_c\,\mathrm{sgn}(\dot{q}) = \tau - J^T f \qquad (4.6)$$

For the 3DOF planar robot (Figure 3.1) gravity effect does not have to be considered. Therefore gravity term can be erased from the equation. Viscous friction is assumed to be negligible because joint velocities are not high. The equation (4.6) can be rewritten to

$$M(q)\ddot{q} + h(q,\dot{q}) + F_c\,\mathrm{sgn}(\dot{q}) = \tau - J^T f \qquad (4.7)$$

The robot dynamics without friction and applied force can be obtained by solving Recursive Newton-Euler formulation or Lagrangian formulation. Since Lagrangian formulation needs more calculation than Recursive Newton-Euler formulation [2], Recursive Newton-Euler formulation is used for this thesis (see Appendix D for details).

4.3. Position Control

Position control has to be applied until the manipulator enters into contact with the environment. The resolved acceleration method is used for position control law.

From the equation (4.7), torque equation of the robot is

$$\tau = M(q)\ddot{q} + h(q,\dot{q}) + F_c\,sgn(\dot{q}) \qquad (4.8)$$

If the model is perfect and there exists no external disturbance, the end-effector is brought to the desired trajectory in Cartesian coordinates ($x = x_d$) and is kept on it, with the Cartesian acceleration command ux, translated into joint space using the equation(4.3),

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q} \qquad (4.9)$$

Equation (4.8) and (4.9) give the torque command for $u_x = \ddot{x}$, an open loop control law

$$\tau = M(q)J^{-1}(u_x - \dot{J}\dot{q}) + h(q,\dot{q}) + F_c\,sgn(\dot{q}) \qquad (4.10)$$

In practice, however, we can not avoid modeling and measuring errors as well as external disturbances. Therefore closed loop control compensator must be used for reducing the error.

$$ux = \ddot{x}_d + K_{pv}(\dot{x}_d - \dot{x}) + K_{pp}(x_d - x) \qquad (4.11)$$

We want to achieve in overdamped system because overshoot might damage the environment. To obtain overdamping, K_{pv} and K_{pp} for a linear system must satisfy the condition

$$K_{pv}^2 >= 4\,K_{pp} \qquad (4.12)$$

In fact, the system is nonlinear and discontinuous (the dry friction term) and position error result.

4.4. Hybrid Control

After the manipulator enters into contact with the environment (when the applied force reaches 5N), the force applied as well as the position and the angle of the end-effector must be controlled. Hybrid control is used to control position and force simultaneously. Figure 4.2 shows force data when controller is switched from position control to hybrid control.

Using the equation (4.10) and the diagonal selection matrix S which shows whether the force control or the position control is applied for each direction, the control law can be written as

$$\tau = M(q)J^{-1}[(I - S)u_x - \dot{J}\dot{q}] + h(q,\dot{q}) + F_c\,sgn(\dot{q}) + J^T f_d + J^T S u_r \qquad (4.13)$$

The command vectors for position and force control are

$$u_x = \ddot{x}_d + K_{pv}(\dot{x}_d - \dot{x}) + K_{pp}(x_d - x) \qquad (4.14)$$

$$u_r = K_{fp}(f_d - f) + K_{fi}\int(f_d - f)\,dt \qquad (4.15)$$

Figure 4.3 shows the block-diagram of the controller. Better results might be able to be obtained if a PID control law is used for the force control, but the signal from the force sensor is very noisy and can not be differentiated. The sample in figure 4.4 of the signal from the force sensor shows the noise content of the signal.

Craig [28] mentioned that we can obtain the derivative of the force on the environment as fe = kex if we know the environment stiffness ke. It is also

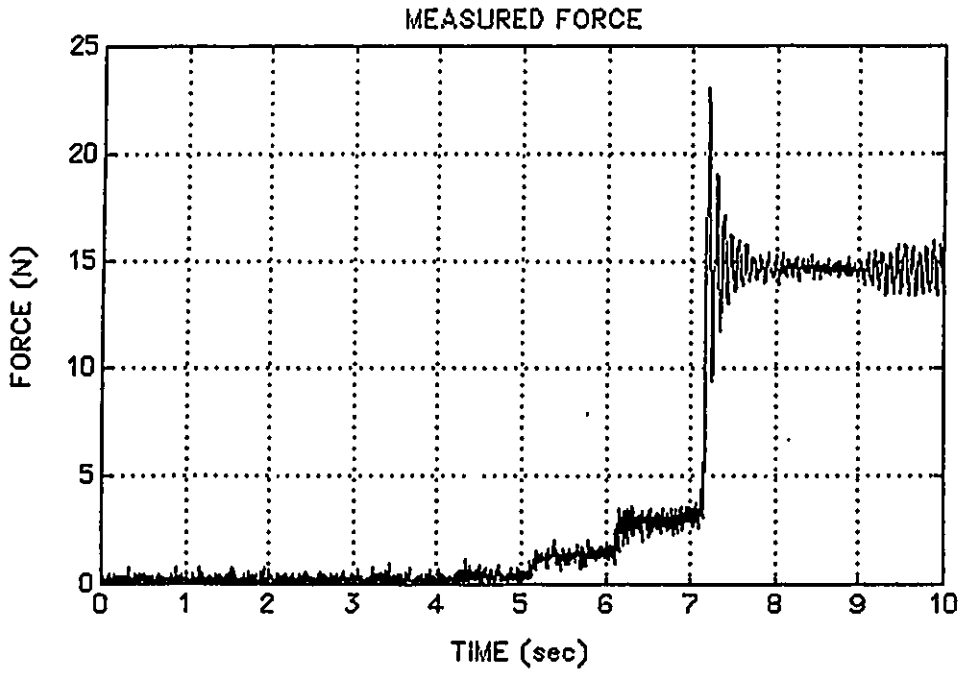


Fig. 4.2 Force Data from position control to hybrid control

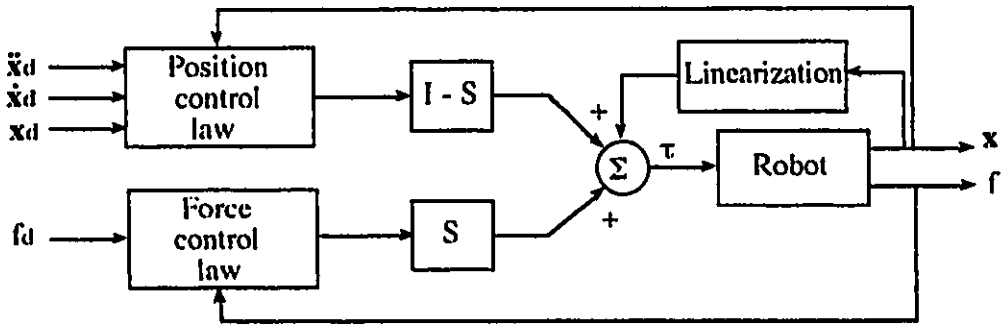


Fig.4.3 Block-diagram of the controller

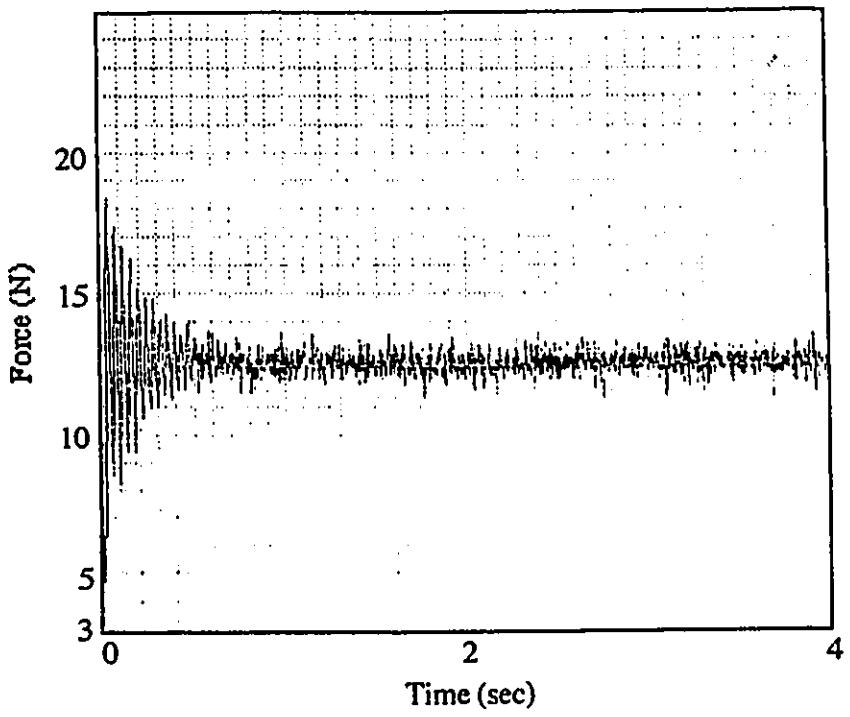


Fig.4.4 Signal from the Force Sensor

possible to estimate the force derivative using a state estimation if we know environment impedance [29]. We can not, however, use these methods before the environment impedance is identified. Another way of obtaining the force derivative is applying a filter to remove the noise from the measured force signal. A Butterworth low-pass filter with a low cut-off frequency creates a significant time lag and makes the system unstable, while high cut-off frequency leaves signal noisy. The equation of Nth order Butterworth filter for continuous system [30] is

$$|H(j\omega)| = 1/(1+(\omega/\omega_c)^{2N})^{1/2} \qquad (4.16)$$

where

ω_c : Cutoff Frequency

The position and force control loops mentioned above are coupled. The equation (4.13) can be changed to

$$M(q)\ddot{q} = M(q)J^{-1}[(I - S)u_x - \dot{J}\dot{q}] + J^T S u_f \qquad (4.17)$$

Using the equations (4.9),(4.14) and (4.15), this equation can be written as

$$\begin{aligned}(I - S)[(\ddot{x}_d - \ddot{x}) + K_{fv}(\dot{x}_d - \dot{x}) + K_{fp}(x_d - x)] - S\ddot{x} \\ + J M^{-1} J^T S[K_{fp}(f_d - f) + K_{fi}\int(f_d - f)\,dt] = 0\end{aligned} \qquad (4.18)$$

Usually $JM^{-1}J^T$ is not a identity matrix. Therefore the position control loop is affected by force control loop. This problem can be cleared using the another control law proposed in the literature [25] after the environment impedance identification. We can decouple the position and force control loops by using above mentioned environment spring constant. In the literature [25], following control law was proposed and proved that the position and force control loops are decoupled with this.

$$\tau = M(q)J^{-1}[(I - S)u_x + S k_e u_f - \dot{J}\dot{q}] + h(q,\dot{q}) + C_c\,sgn(\dot{q}) + J^T f_d \qquad (4.19)$$

$$u_x = \ddot{x}_d + K_{fv}(\dot{x}_d - \dot{x}) + K_{fp}(x_d - x) \qquad (4.20)$$

$$u_f = K_{fp}(f_d - f) \qquad (4.21)$$

4.5. Neural Networks Control

It is very difficult to obtain the desired force with the force control law of the hybrid control which is used for environment impedance identification mentioned before in this thesis, due to several reasons. First, the model is not perfect. Second, we can not use a high gain for the case of the unknown environment because its dynamics affects the system. Third, it is difficult to compensate friction perfectly. As far as the robot manipulator is concerned, friction depends also on configuration of the robot manipulator. Fourth, it is impossible to avoid external disturbance completely. Furthermore, the properties of some environments such as rubber, plastic, styrofoam, etc. is affected by temperature and humidity, such that the properties of the environment might change in time. In order to solve these problems, an adaptive scheme can be considered. In particular, the neural networks control seems to be useful to adapt the controller to unknown environments and to compensate for disturbances. The neural networks controller which does not require any previous environment impedance information is introduced in this chapter.

As mentioned in Chapter 2.2., there are several types of neural networks controllers. In this thesis, a direct neural network controller which seems the most effective way of using neural networks for controllers is applied to the robot. Figure 2.4 shows the typical direct neural networks controller. The conventional controller such as the hybrid controller can be designed without much difficulty, as shown in Chapter 2.4., and is built first. Then we can use the output of such a controller as the input of neural network controller. Furthermore, by using output of hybrid controller with output of the neural network controller as input torque commands to the robot, faster adaptation of the neural networks controller to unknown environment can be expected. Figure 4.5 shows the block-diagram of the controller.

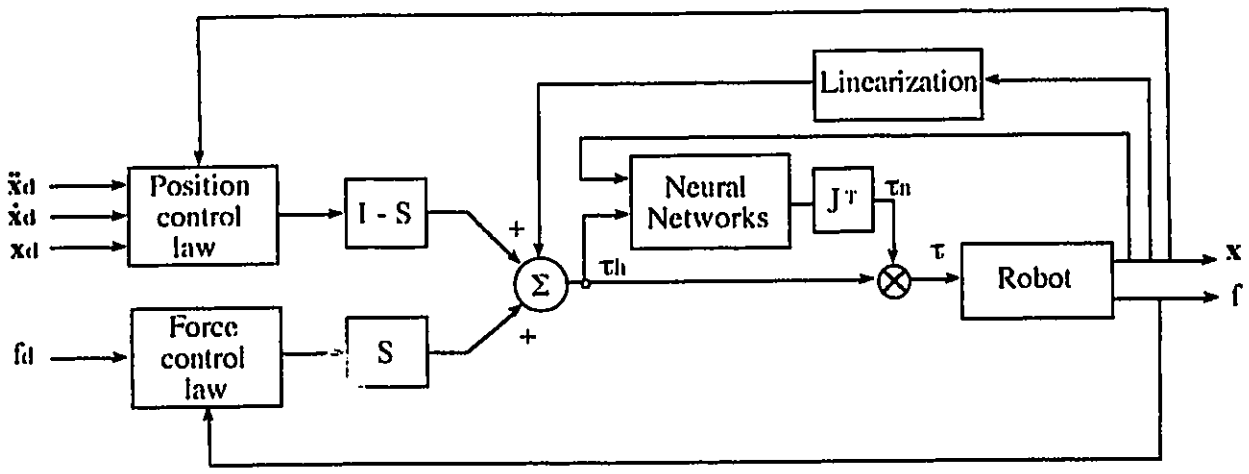


Fig.4.5 Neural Network Controller

In this control scheme, the neural networks is used to compensate the friction term, model errors and other external disturbances. Therefore, the hybrid controller used for this controller does not need to compensate the friction. In this case the equation (4.13) can be reduced to

$$\tau = M(q)J^{-1}[(I - S)u_x - \dot{J}\dot{q}] + h(q,\dot{q}) + J^T f_d + J^T S u_f \qquad (4.18)$$

The command vectors for this controller can be changed as

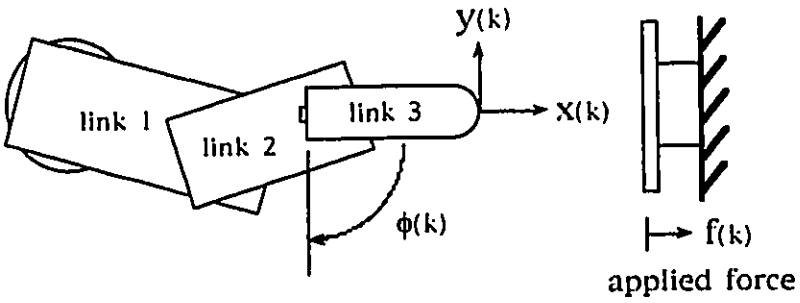
$$u_x = \ddot{x}_d + K_{pv}(\dot{x}_d - \dot{x}) + K_{pp}(x_d - x) \qquad (4.19)$$

$$u_f = K_{fp}(f_d - f) \qquad (4.20)$$

As we can see in the equation (4.20), the integral of the error is not needed any more because the neural network controller compensates the error of the hybrid controller. Furthermore, the gain in the equation (4.20) does not need to be changed for various environments since the neural network controller adjusts itself to unknown environments.

The architecture of the neural network is shown in Figure 4.6. Input signals to the neural network are the current and previous position of the robot manipulator in Cartesian coordinates and the computed torque commands for each motors by the hybrid controller. The configuration (position) and the velocity of the robot manipulator can be known from the current and the previous position (x,y and ϕ). The desired acceleration of the robot manipulator corresponding to the desired position/force can be obtained from the computed torque commands of the hybrid controller. Output signals from the neural network are force commands for robot manipulator in Cartesian coordinates. The force commands are converted to torque commands for each motors using the equation (4.5) and are added to the torque commands from the hybrid controller. The robot manipulator receives the sum of the torque commands from the controller. The result of the robot manipulator movement as response of the torque commands is measured by the encoders of each motors and by the force sensor. The weights of the neural network controller are adjusted every sampling time (1msec) using both position and force error. The back-propagation method is used as a learning law for the weights adjustment.

The neural network shown in Figure 4.6 is a nonlinear three-layer neural network controller. The number of the layer can be more than three [33], the minimum number of the layer is chosen in this thesis since the search of a better neural network architecture is not main purpose. The input layer consists of 9 neurons, the hidden layer consists of 12 neurons and the output layer consists of 3 neurons. Signals to the input layer are multiplied by the weights w_{ij} and summed in each neuron in the hidden layer. Each neuron is activated if the summed value yields a threshold value. Output signals from the hidden layer are multiplied by weights w_{jk} and summed in each neuron in



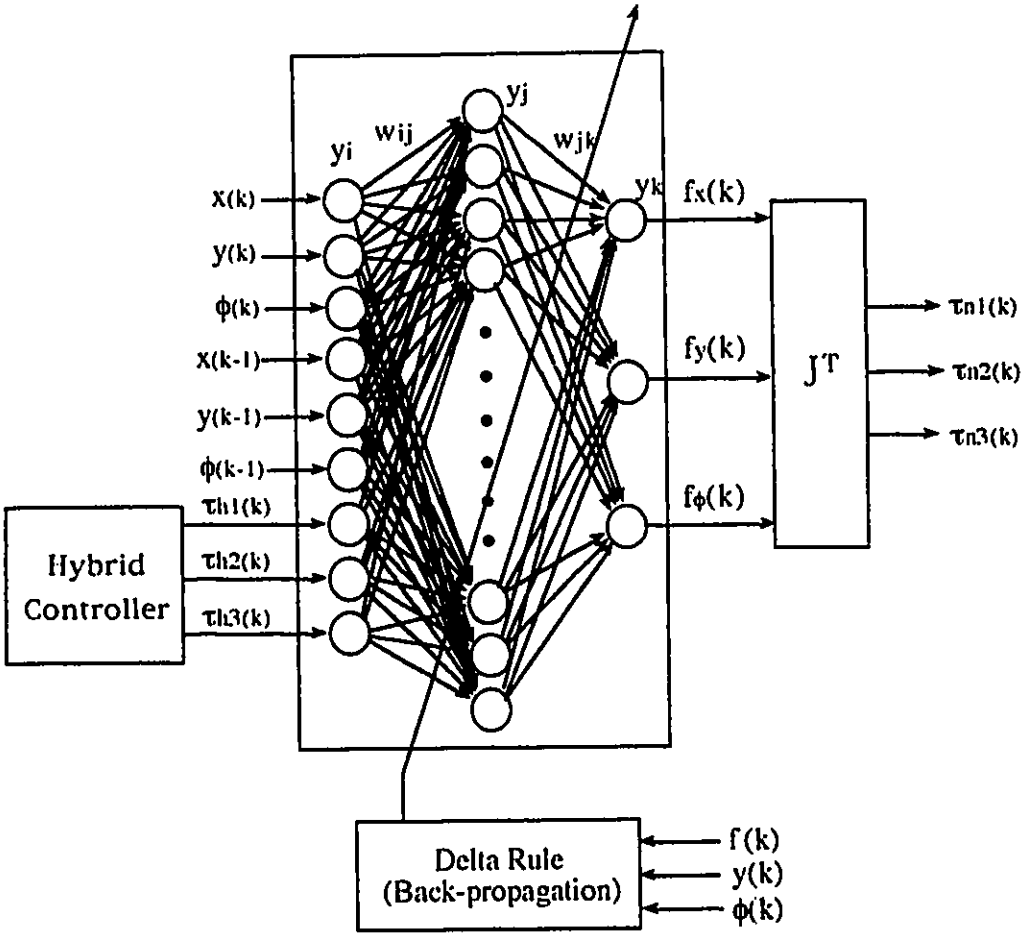


Fig.4.6 Architecture of the Neural Network

the output layer. If the summed value yields each threshold value, the neuron is activated. Signals from the output layer are force commands for robot manipulators in Cartesian coordinates. The weights and threshold values of the neural network is adjusted every sampling time (1msec).

The activation function (sigmoid function) [34] used in this neural network is written as

$$f(u) = 2/(1+e^{-u}) - 1 \qquad (4.21)$$

where u is input to the layer.

The back-propagation method is described as follows:

$$E = 1/2\Sigma(yd - y)^2 \qquad (4.22)$$

$$uj = \Sigma(wijyi) \qquad (4.23)$$

$$yj = f(uj) \qquad (4.24)$$

where

E : Squared Error

yd : Desired Output from the Neural Network

y : Output from the Neural Network

uj : Input to the Neuron j

yj : Output from the Neuron j

wij : Weight for the signal from the Neuron i to the Neuron j

The sigmoid function for the neuron j is

$$f(uj) = 2/(1+e^{-uj}) - 1 \qquad (4.25)$$

Using the squared errors given by the equation (4.22), the adjustment of the weights from the hidden layer to the output layer can be obtained as

$$\Delta w_{jk} = \eta \partial E/\partial w_{jk}$$
$$= \eta(\partial E/\partial y)(dy/du_k)(\partial u_k/\partial w_{jk})$$
$$= \eta\, e_k\, f'(u_k)\, y_j \qquad (4.26)$$

where

η : Learning Rate

The adjustment of the weights from the input layer to the hidden layer can be obtained as follows

$$\Delta w_{ij} = \eta \partial E/\partial w_{ij}$$
$$= \eta(\partial E/\partial y_j)(dy_j/du_j)(\partial u_j/\partial w_{ij})$$
$$= \eta(\Sigma(\partial E/\partial y)(dy/du_k)(\partial u_k/\partial y_j))(dy_j/du_j)(\partial u_j/\partial w_{ij})$$
$$= \eta\, \Sigma(e_k\, f'(u_k)\, w_{jk})\, f'(u_j)\, y_i \qquad (4.27)$$

From the equation (4.26) and (4.27), new weights are written as

$$w_{ij}(k) = w_{ij}(k-1) + \eta\Sigma(e_k(k)f'(u_k)w_{jk}(k-1))f'(u_j)y_i(k) \qquad (4.28)$$
$$w_{jk}(k) = w_{jk}(k-1) + \eta e_k(k)f'(u_k)y_j(k) \qquad (4.29)$$

In this neural network controller, the error in the equation (4.22) is the difference between the desired robot manipulator position/force and the measured robot manipulator position/force, therefore, tracking the desired position/force can be expected with no significant error with this algorithm.

5. Environment Impedance Identification

In this chapter, two kinds of environment impedance identification methods are described. First, environment impedance identification method using least square estimation and then environment impedance identification method using neural networks are presented.

If the environment is not rigid, it can be modeled as a lumped mass, a damper, and a spring system (see Figure 5.1).

$$Me\,\ddot{x} + Be\,\dot{x} + Ke\,x_e = f \tag{5.1}$$

where Me, Be and Ke are mass inertia coefficient, damping coefficient and spring coefficient respectively. They are assumed unknown parameters which depend on environment properties and $\ddot{x}$, $\dot{x}$, x_e and f are assumed measurable. Environment impedance identification means the evaluation of the unknown parameters Me, Be and Ke, explicitly or implicitly.

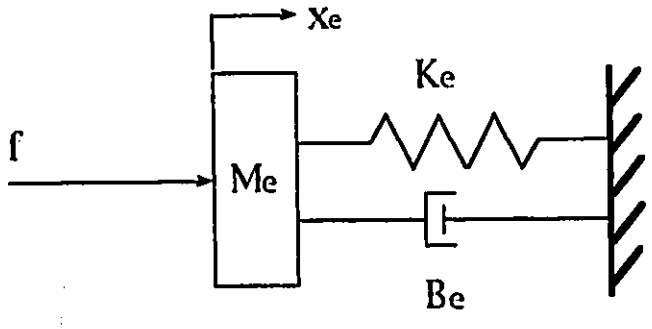


Fig.5.1 The Environment Model

5.1. Application of Least Square Estimation Method

The environment impedance coefficients Ke,Be and Me will be identified on line using the Recursive Least Squares Estimation Method. From the equation (5.1), reaction force equation can be rewritten as

$$f = \phi^T \theta \tag{5.2}$$

where

$$\phi^T(t) = [\ \ddot{x}(t)\ \ \dot{x}(t)\ \ xe(t)\ ]$$

$$\theta^T(t) = [\ Me\ \ Be\ \ Ke\ ]$$

The vector $\phi(t)$ contains measured values, while $\theta(t)$ is an vector of unknown parameters.

The basic idea of this method is to make the output, computed from the model in equation (5.2), agree in the sense of least squares as closely as possible with the measurement of $f(t)$ [31]. Let us define the error between estimated force and measured force as

$$\varepsilon(t) = f(t) - \hat{f}(t) = f(t) - \phi^T(t)\,\theta \tag{5.3}$$

We introduce the notation

$$F(t) = [f(1)\ f(2)\ \ldots\ f(t)]^T$$

$$E(t) = [\varepsilon(1)\ \varepsilon(2)\ \ldots\ \varepsilon(t)]^T$$

$$\Phi^T(t) = [\phi^T(1)\ \phi^T(2)\ \ldots\ \phi^T(t)]^T$$

The least-square error can be written as

$$L(\theta,t) = 1/2 \sum_{i=1}^{t} \varepsilon(i)^2 = 1/2 \sum_{i=1}^{t} (f(i) - \phi^T(i)\,\theta)^2 = 1/2\ \|E\|^2 \tag{5.4}$$

This function is minimal if the estimated parameters vector θ is given by

$$\Phi^T\Phi\hat{\theta} = \Phi^T F \tag{5.5}$$

If the matrix $\Phi^T\Phi$ is nonsingular, the minimum is unique and can written as

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T F \tag{5.6}$$

We can obtain the estimated coefficients vector $\theta(t)$ recursively. In this way, the results obtained at time t-1 can be used to calculate the estimated coefficients vector at time t.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + k(t)\varepsilon(t) \tag{5.7}$$

where

$$k(t) = p(t)\phi(t) \tag{5.8}$$

$$p(t) = (p(t-1)^{-1} + \phi(t)\phi^T(t))^{-1} \tag{5.9}$$

$$\varepsilon(t) = f(t) - \phi^T(t)\,\hat{\theta}(t-1) \tag{5.10}$$

In this method, an vector $\theta(t)$ of unknown parameters is assumed to be constant. If it is time-varying, a forgetting factor can be applied.

Ke,Be and Me can be obtained by other off-line or on-line methods under the limited condition.

$$Ke = f/x_\omega \qquad (\text{when } \dot{x} = 0,\ \ddot{x} = 0)$$

$$Be = (f - Kex_\omega)/\dot{x} \qquad (\text{when } \ddot{x} = 0 \text{ and Ke is known})$$

$$Me = (f - Kex_\omega - Be\dot{x})/\ddot{x} \qquad (\text{when Be and Ke is known})$$

Then we can compare the results with previous ones. However, it is difficult to obtain Be and Me with this method because velocity and acceleration signals are small and noisy.

5.2. Application of Neural Networks

Neural networks are known to be able to identify the system implicitly, and they can identify the system even if it is nonlinear because of their capability of taking into account of the nonlinearities implicitly.

The neural networks (different from the neural networks for the neural networks controller) are introduced for environment impedance identification in the force loop of the controller as shown in Figure 5.2. They are placed parallel with actual environment. Position, velocity and acceleration information of the robot manipulator, which is same information used for environment impedance identification using the least-square estimation, is sent to the neural networks. The neural networks output is the calculated force. The weights of the neural network are adjusted every sampling time (1msec) by the error between calculated force and measured force.

The architecture of the nonlinear three-layer neural networks for environment impedance identification is shown in Figure 5.3. The input layer

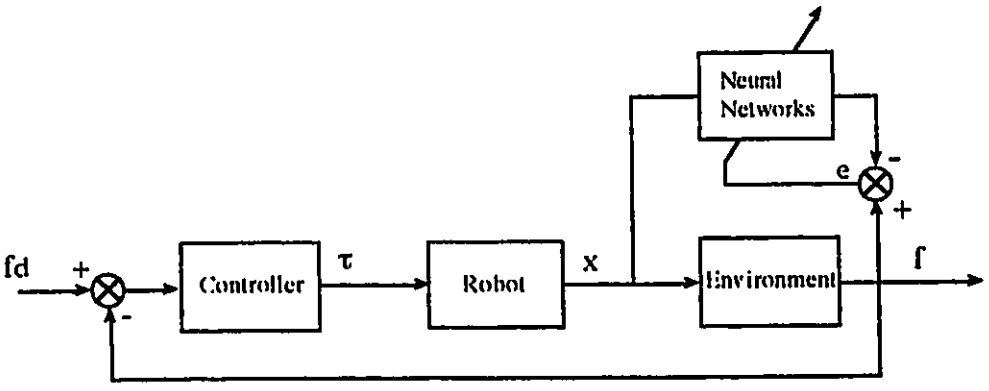


Fig.5.2 Environment Impedance Identification

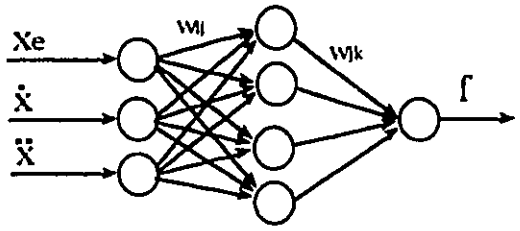


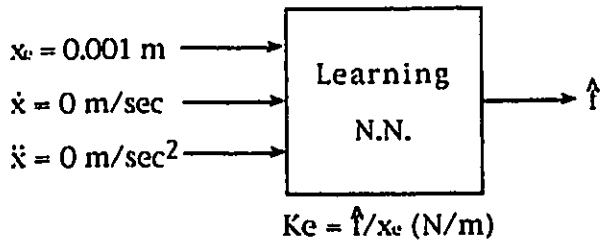
Fig.5.3 Three Layer Neural Network for Identification

consists of 3 neurons, the hidden layer consists of 4 neurons and the output layer consists of 1 neuron.

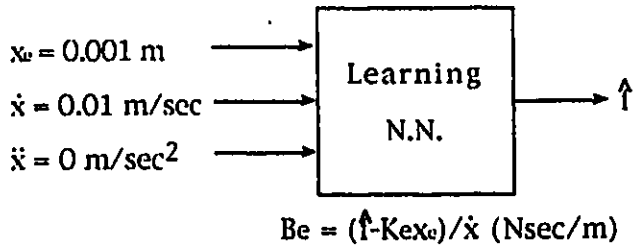
The process which is done in the neural network is same as that of the neural network controller explained in Chapter 4.5. The weights and the threshold values of the neural network are adjusted by back-propagation algorithm every sampling time (1msec).

After the neural network finished learning, or even during the learning process, it realizes the model of environment impedance in themselves implicitly because of their learning ability. However, if we input some dummy values which are close to learned values (actual input values) to the neural networks, it is possible to approximately obtain the coefficients explicitly by comparing dummy input values with output value. We must be careful for chosing dummy input values. If they are far different from learned values, the output of the neural network would be wrong. Because the neural networks store the relation between given inputs and output. Valid Output of the neural network can be obtained only when learned inputs are given. Therefore if we do not know learned input values, this explicit identification method can not be used. However, if we know learned input values this method is useful for explicit identification. For example, if we know

the robot manipulator push the environment about 1mm to obtain the desired force and its velocity and acceleration are small, we can input x_e = 0.001 m, $\dot{x}$ = 0 m/sec and $\ddot{x}$ = 0 m/sec^2 to the neural network. Then the spring coefficient Ke (N/m) which is realized in the neural network itself can be obtained approximately by dividing the estimated output force (N) by x_e.



Next, we input x = 0.001 mm, $\ddot{x}$ = 0 m/sec^2 and small velocity $\dot{x}$ = 0.01 m/sec to the neural network. The damping coefficient can thus be obtained approximately by subtracting the force which is produced by the displacement 0.001m from the estimated output force and dividing it by $\dot{x}$.



Furthermore the inertia coefficient can also be obtained approximately in the same way.

6. Experimental Results

Experimental study was performed with the configuration explained in Chapter 3. In this chapter, experimental results for environment impedance identification using least square estimation is first shown. Then experimental result of environment impedance identification using neural networks is shown. In the end, experimental results of the neural network controller are shown. The choice of numerical values for the gains of the hybrid controller for environment impedance identification is by trial and error and should be later justified by further studies. Remember, the controller used for environment impedance identification is different from the neural network controller.

Environment impedance identification was done with the hybrid controller explained in Chapter 4.4 since exact desired force control is not demanded for identification. One inch thickness styrofoam and rubber are used as environment. The gains in the equation (4.15) for the force control law are chosen as follows by trial and error

For Styrofoam : $K_{fp} = 1.0$, $K_{fi} = 0.003$

For Rubber : $K_{fp} = 0.25$, $K_{fi} = 0.0015$

The initial position of the robot manipulator for environment impedance identification is decided as the position where the robot manipulator applies a 5N force to the environment, since the initial position for identification is not evident when the force applied is low. For example, for a styrofoam environment, the surface of the environment is rough. so that

some force must be used for eliminating the gap between the environment and the force sensor and a gap in the force sensor itself. For this reason, the controller is switched from the position control to the hybrid control when the measured force becomes 5N. In the force control, 120 Hz (bandwidth of the force sensor) 2nd-order Butterworth low-pass filter is used for the measured force after sampling to the dSPACE to get rid of noise.

For identification, measured force f, robot manipulator position x, velocity $\dot{x}$ and acceleration $\ddot{x}$ are used; a 35Hz Butterworth low-pass filter is used only for the noisy acceleration signal.

The gains in the equation (4.14) for the position control law are chosen as follows by trial and error to achiving an overdamped system

$$K_{pp} = 400.0 \, , \, K_{pw} = 80.0$$

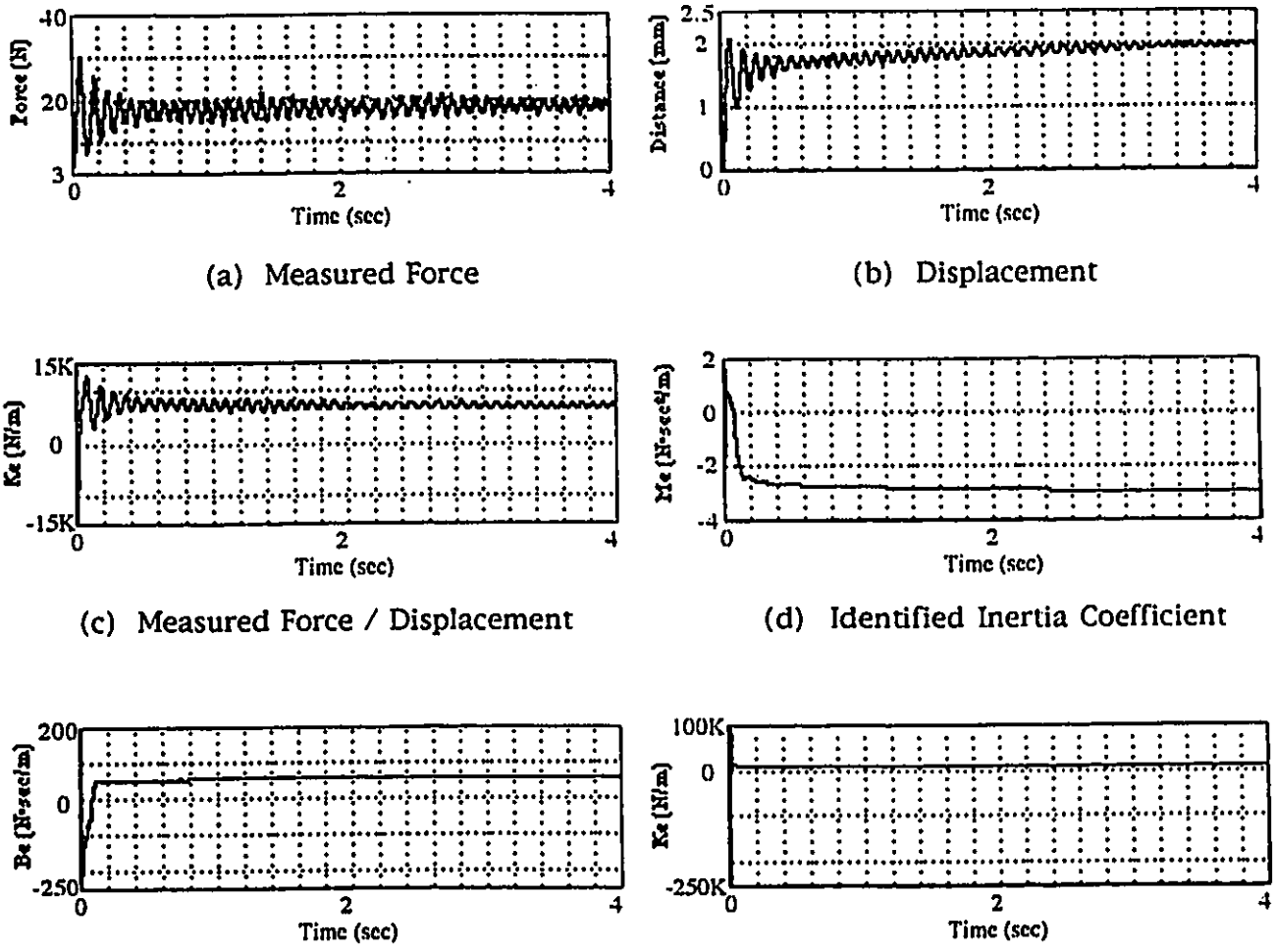
These gains are used for the position control law of the position controller, the hybrid controller and the neural network controller.

6.1. Identification using Least Square Estimation

The experimental results of identification using least square estimation (LSM) for the styrofoam and rubber are shown in Figure 6.1 to 6.5. These graphs show the measured force, the surface displacement of envirnment by the robot manipulator, the spring coefficient calculated from the measured force and the surface displacement of envirnment, identified coefficients Me, Be and Ke using the least square method for first 4 seconds of identification. The identification starts after the measured force reaches 5N, when the controller is switched from position control to hybrid control. The initial values for the coefficients are chosen as 0 $Nsec^2/m$, 0 Nsec/m and $2*10^5$ N/m

for Me,Be and Ke respectively. The desired force is 20N.

The results in Figure 6.1 show that the coefficients are practically identified during first 0.12 seconds. After first 0.12 seconds, the values of the coefficients change less significantly. These experimental results show that the environmental impedance can practically be identified in a very short time using the least square method. It is difficult to compare the estimated val-



(a) Measured Force

(b) Displacement

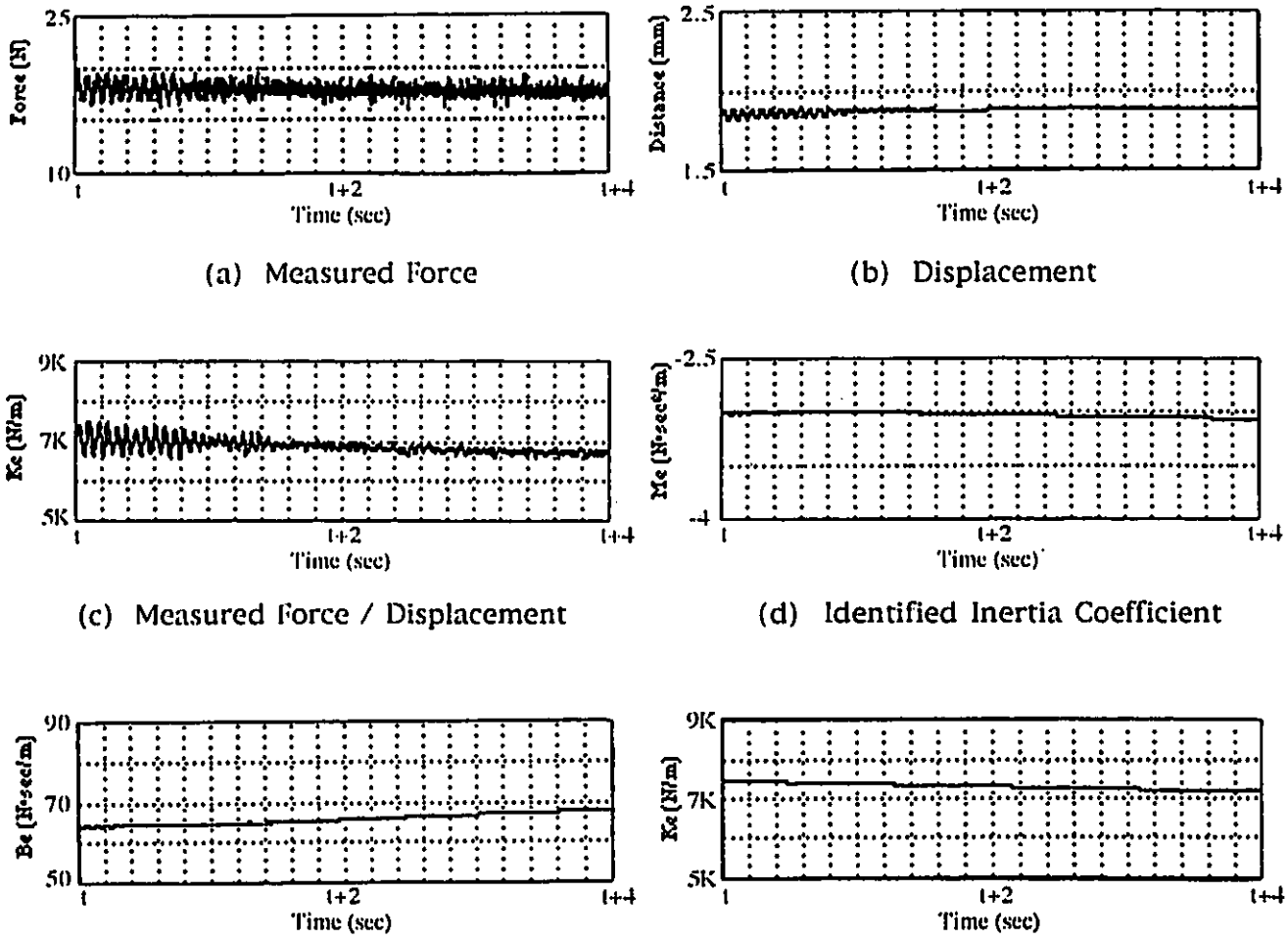
(c) Measured Force / Displacement

(d) Identified Inertia Coefficient

(e) Identified Damping Coefficient

(f) Identified Spring Coefficient

Fig.6.1 Identification Result of first 4sec for Styrofoam (LSM)



(a) Measured Force

(b) Displacement

(c) Measured Force / Displacement

(d) Identified Inertia Coefficient

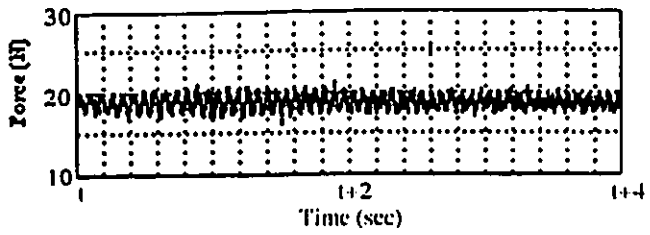
(e) Identified Damping Coefficient

(f) Identified Spring Coefficient

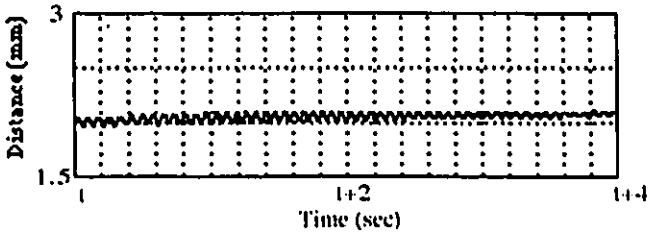
Fig.6.2 Identification Result for Styrofoam (LSM)

ues of Ke in (c) and (f) in this Figure since the values change a lot in the beginning of identification. Some time after the time slot shown in Figure 6.1 is shown in Figure 6.2 in order to compare the estimated values of Ke in (c) and (f) in the same scale.

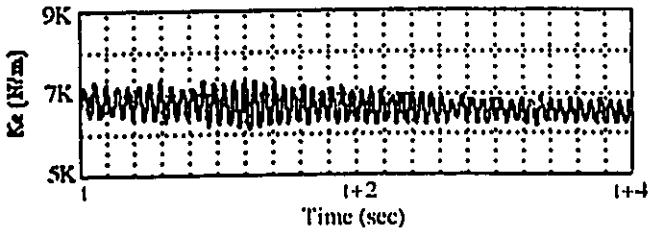
As we can see from this result, the spring coefficient Ke is a realistic value since the estimated values of Ke in (c) is 6.8K(N/m) and (f) is 7.2K(N/m)



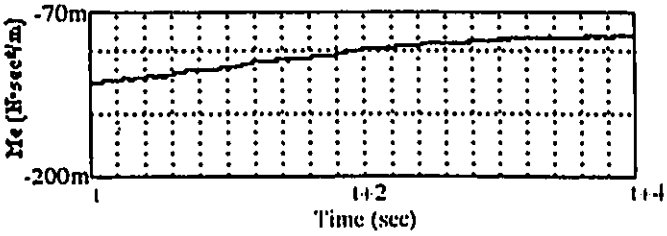
(a) Measured Force



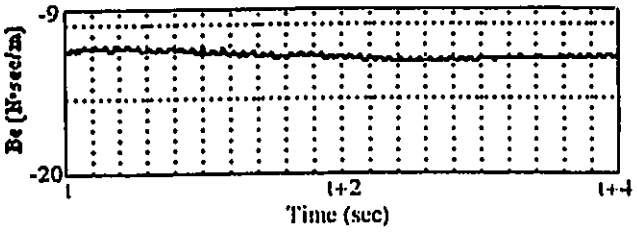
(b) Displacement



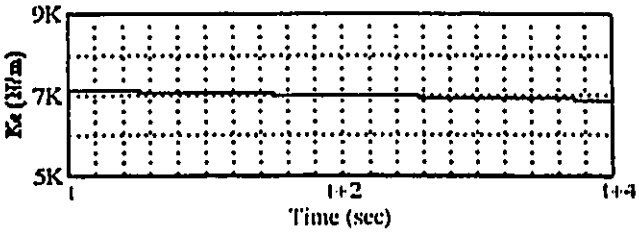
(c) Measured Force / Displacement



(d) Identified Inertia Coefficient



(e) Identified Damping Coefficient

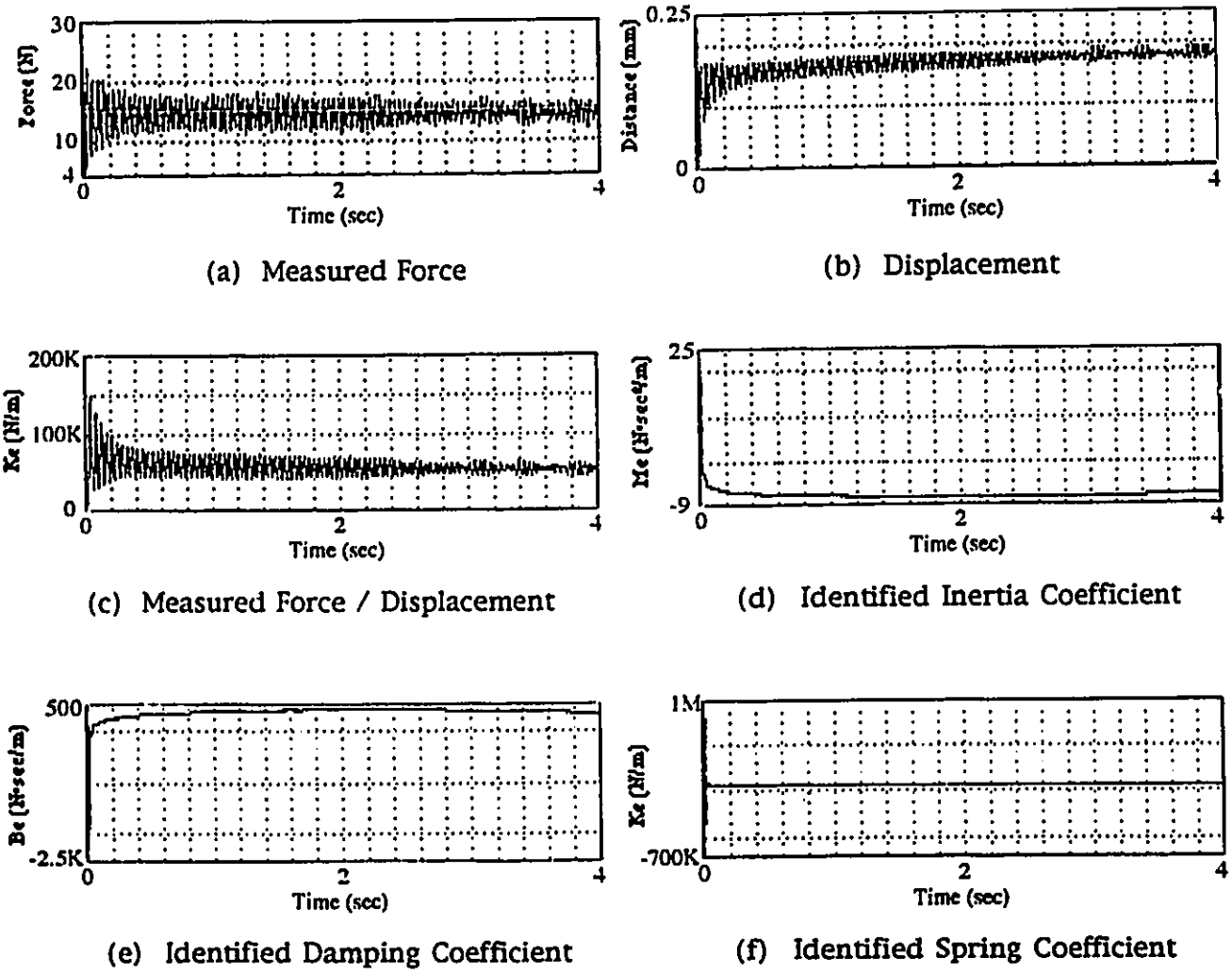


(f) Identified Spring Coefficient

Fig.6.3 Identification Result from raw input signals for Styrofoam (LSM)

which are almost same (less than 6% difference). The damping coefficient Be seems realistic. Unfortunately there is no way to prove this value is correct since it can not be estimated in another way in the same experiment and it is affected by experimental condition (temperature, humidity, etc.). The value of the inertia coefficient Me is unrealistic as it results negative. In this particular case, one of the reasons for this result is that the inertia coefficient

Me is very small coefficient. It is very difficult to estimate exactly Me. If we use another combination of unfiltered and filtered input signals for the identification the results become even more unrealistic. For example, if we use all raw input signals (unfiltered input signals), the measured force f, robot manipulator position x, velocity $\dot{x}$ and acceleration $\ddot{x}$ for identification, as shown in Figure 6.3, both the inertia coefficient Me and the damping coefficient Be result negative.



(a) Measured Force

(b) Displacement

(c) Measured Force / Displacement

(d) Identified Inertia Coefficient

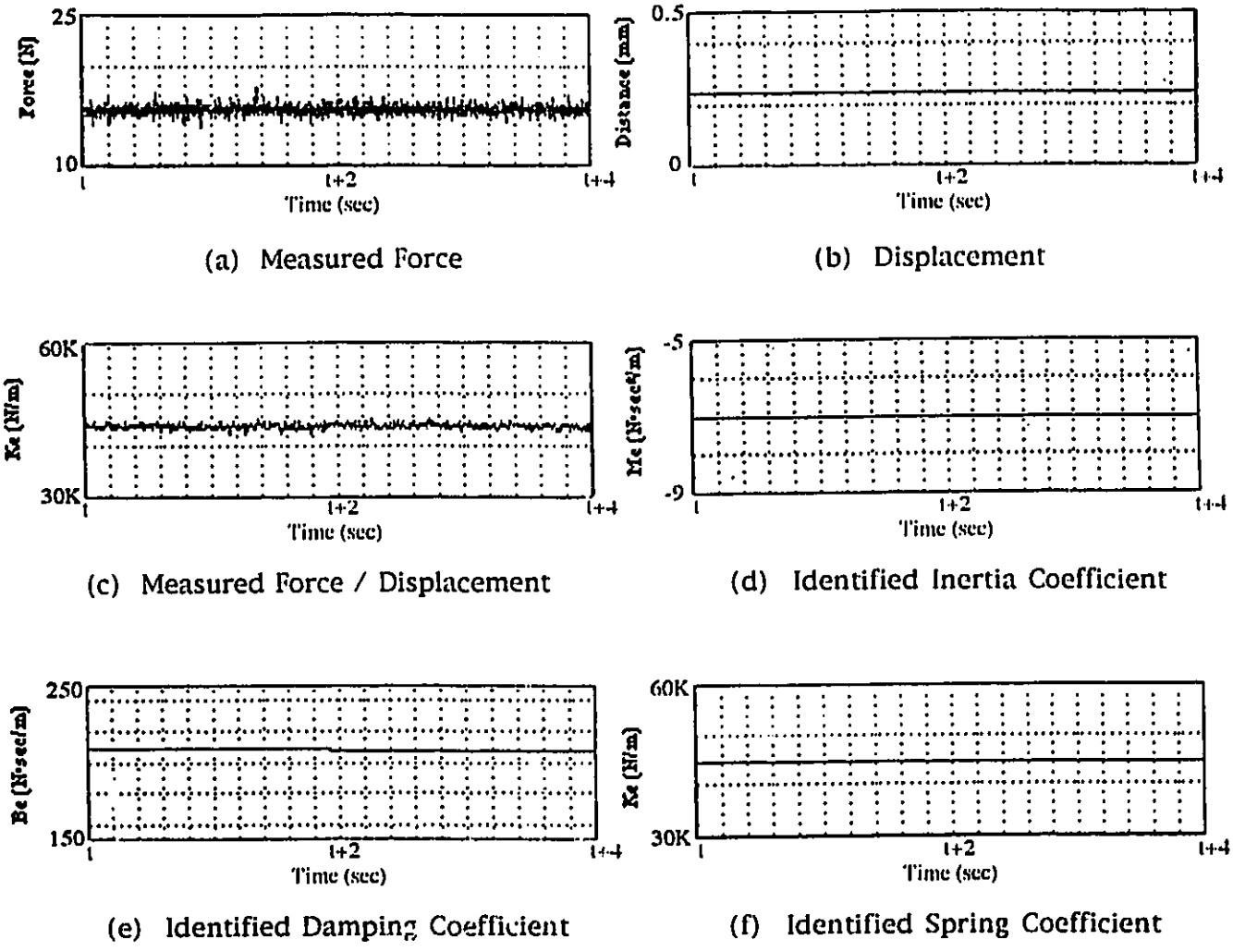
(e) Identified Damping Coefficient

(f) Identified Spring Coefficient

Fig.6.4 Identification Result of first 4sec for Rubber (LSM)

However, as far as the spring coefficient Ke is concerned, the estimated value is realistic since it is always almost the same as the calculated spring coefficient from the surface displacement of the environment and measured force measurements.

Next, the experimental result of identification using least square method (LSM) for the Rubber are shown (Figure 6.4). These graphs also show the measured force, pushed distance by the robot manipulator, the spring coeffici-



(a) Measured Force

(b) Displacement

(c) Measured Force / Displacement

(d) Identified Inertia Coefficient

(e) Identified Damping Coefficient

(f) Identified Spring Coefficient

Fig.6.5 Identification Result for Rubber (LSM)

ent calculated from the measured force and the surface displacement of the environment, identified coefficients Me,Be and Ke using the least square method.

The coefficients are practically identified during first 0.05 seconds. After first 0.05 seconds, the coefficients change only a little. It is difficult to compare the estimated values of Ke in (c) and (f) in this Figure since the values change a lot in the beginning of identification. Some time after the time slot shown in Figure 6.4 is shown in Figure 6.5 in order to compare the estimated values of Ke in (c) and (f) in the same scale. As we can see from this result, the spring coefficient Ke of the rubber is realistic value since the values of Ke in (c) is 44K(N/m) and (f) is also 44K(N/m) which are obtained in different ways. The damping coefficient Be seems realistic. Again, there is no way to prove this value is correct. The value of the inertia coefficient Me is unrealistic as it is negative.

6.2. Identification using Neural Networks

The identification using neural networks is an implicit method but explicit values for the coefficients can be obtained with proper input values (see Chapter 5.2). In these experiments, the desired force is 20N and the learning rate η of the neural network is set to 0.0001 after some experiment. If the learning rate becomes smaller, the neural networks take more time to learn (see Figure 6.6). If the learning rate is too large, learning may never converge. The experimental results for the identification using neural networks (N.N.) for Styrofoam are shown in Fig.6.7. The fixed dummy input values (surface displacement of the envirnment, its velocity and acceleration) to the neural networks for explicit identification for Styrofoam environment to obtain the coefficients are chosen as follows

Displacement	: x_e = 2mm
Displacing Velocity	: $\dot{x}$ = 10mm/sec
Displacing Acceleration	: $\ddot{x}$ = 1m/sec^2

The combination of these values has to be close to actual combination of input signals to the neural networks as explained in Chapter 5.2. In order to obtain

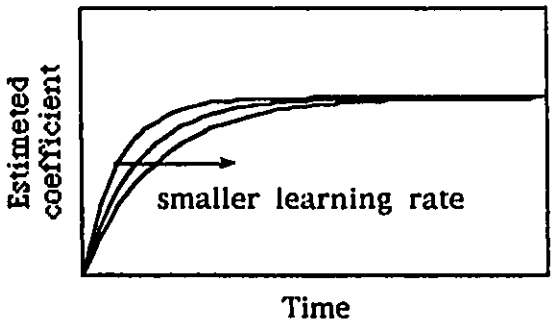


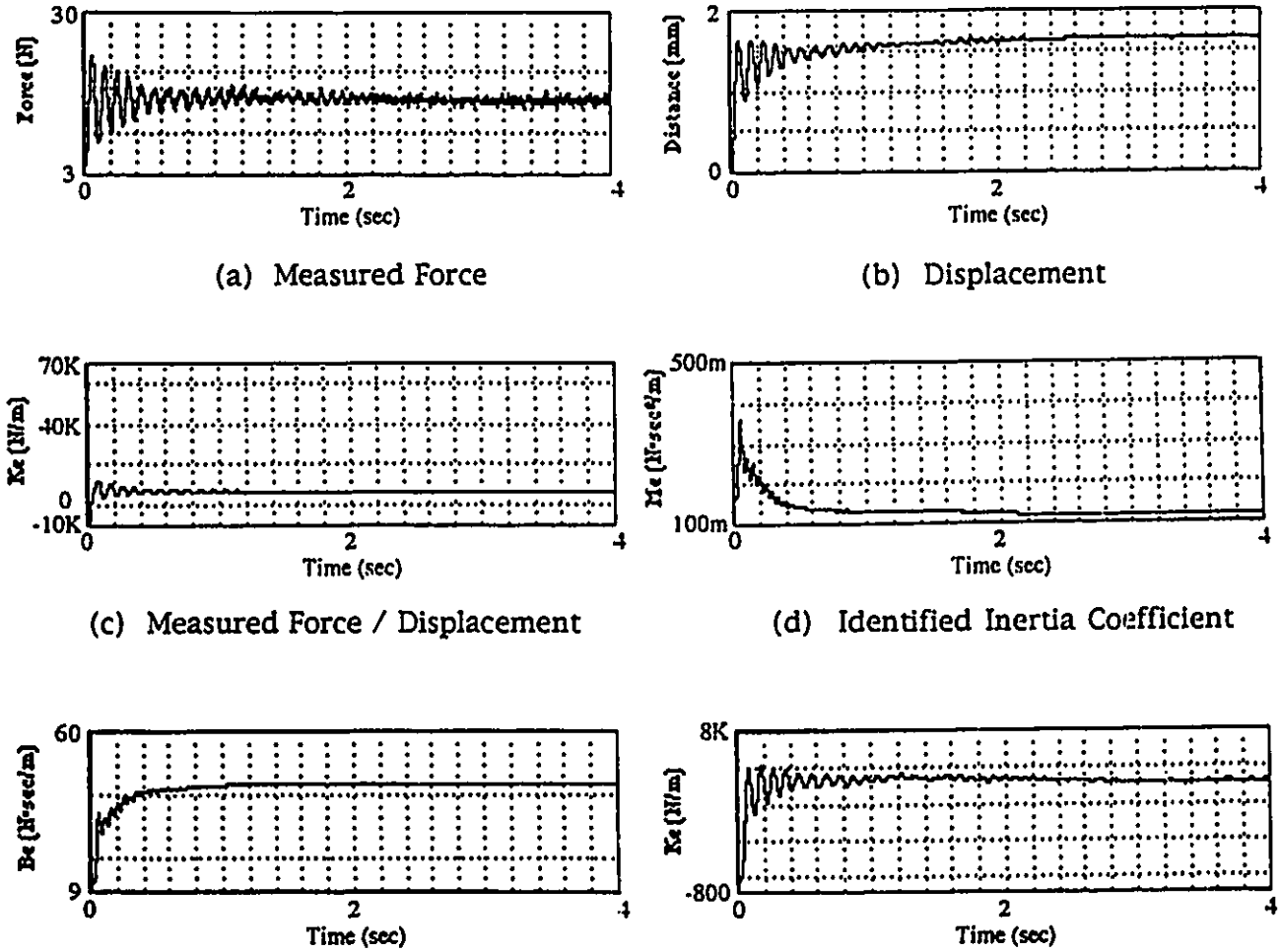
Fig.6.6 Effect of smaller learning rate

the spring coefficient Ke, the surface displacement of the environment (pushed distance) xe is set to 2mm and pushing velocity $\dot{x}$ and acceleration $\ddot{x}$ (velocity and acceleration of the robot manipulator) are set to 0. These signals are dummy inputs to the neural network. We can obtain the spring coefficient Ke (N/m) approximately by dividing output of the neural networks by x_e. For the damping coefficient Be, displacement x_e is set to 2mm, displacing velocity is set to 10mm/sec and displacing acceleration is set to $0m/sec^2$ as dummy input signals to the neural networks. Then we obtain the damping coefficient Be (Nsec/m) approximately by subtracting the force caused by x_e from the output of the neural network and dividing by $\dot{x}$. For the inertia coefficient Me, displacement x_e is set to 2mm, displacing velocity is set to 10m/sec and displacing acceleration is set to $1m/sec^2$ as input signals to the neural network. Then we obtain the inertia coefficient Me ($Nsec^2/m$) approximately by subtracting the force caused by the x_e and $\dot{x}$ from the output of the neural network and dividing by $\ddot{x}$.

If some of the input signals to the neural network are much larger than others, those input signals dominate the other input signals which are almost ignored. In order to avoid this situation, if we know the order of magnitude of the input signals a scaling is useful such that very small signals are multiplied by constants such that all inputs are in the close order of magnitude. For identification of Styrofoam environment, the displacement x is multiplied by 1000 and displacing velocity $\dot{x}$ is multiplied by 100 since they are small compared to other input signals.

Figure 6.7 shows the environmental impedance identification result using neural networks for Styrofoam. As we can see in Figure 6.7, after about 0.6sec the environment impedance is practically identified. Figure 6.8 shows the error of the neural networks output, i.e. the difference between the measured force and the estimated force by the neural networks. This data also

shows that it takes about 0.6sec for the error to become almost zero. The identified coefficients by the neural networks seem good since the spring coefficient Ke is almost same since calculated Ke from (c) is 6.7K(N/m) and estimated Ke from (f) is 6.6K(N/m). The damping coefficient Be is similar to the value which is estimated by the least square method and the inertia coefficient Me is a small positive value. These values seems realistic. However, there is no way to prove these values are correct.



(a) Measured Force

(b) Displacement

(c) Measured Force / Displacement

(d) Identified Inertia Coefficient

(e) Identified Damping Coefficient

(f) Identified Spring Coefficient

Fig.6.7 Identification Result of first 4sec for Styrofoam (N.N.)

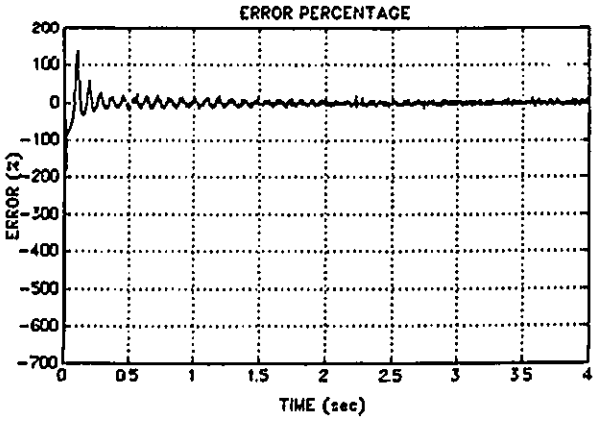


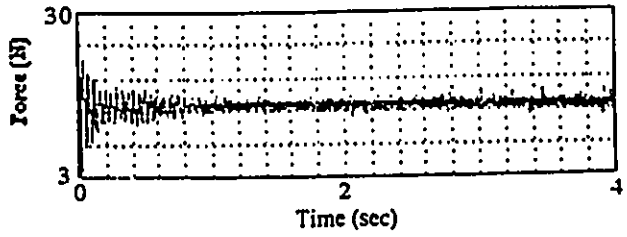
Fig.6.8 Error of the Neural Network Output for Styrofoam

The experimental results of identification using neural networks (N.N.) for Rubber are shown in Figure 6.9. The fixed dummy input values for explicit identification for Rubber environment are chosen as follows

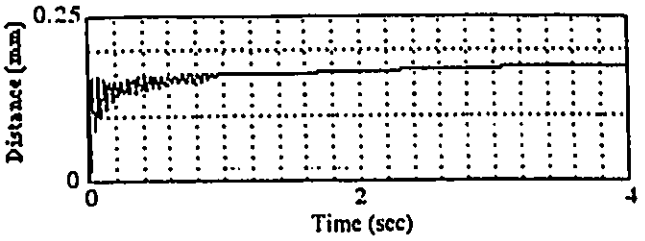
Displacement	: $x = 200\mu m$
Displacing Velocity	: $\dot{x} = 10mm/sec$
Displacing Acceleration	: $\ddot{x} = 0.5m/sec^2$

The coefficients can be obtained approximately in the same way as before. In order to avoid that some input signals dominate the input signal, displacement x is multiplied by 10,000 and displacing velocity $\dot{x}$ is multiplied by 100, since they are small compared to other input signals.

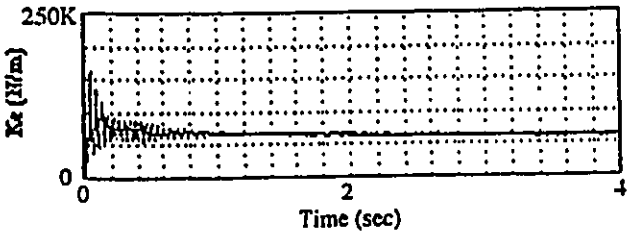
As we can see in Figure 6.9, it also takes about 0.6sec to identify environment impedance. The identified spring coefficient Ke for Rubber environment by neural network also seems realistic since calculated Ke from (c) is 62K(N/m) and estimated Ke from (f) is 56K(N/m). The damping coefficient Be results smaller than the value which is estimated by the least square method. The inertia coefficient Me also seems realistic since it is a small positive value.



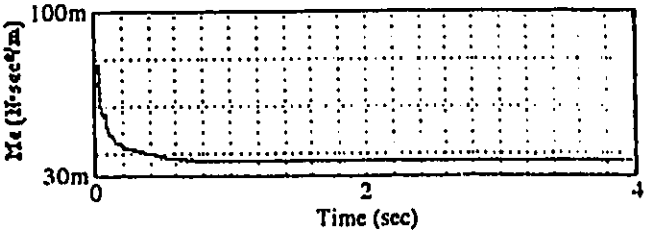
(a) Measured Force



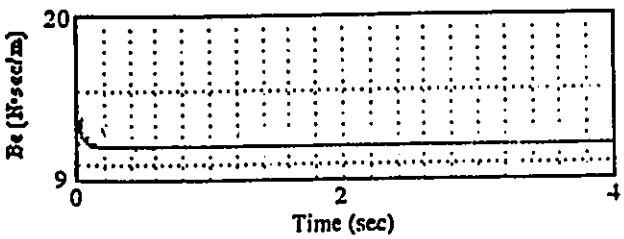
(b) Displacement



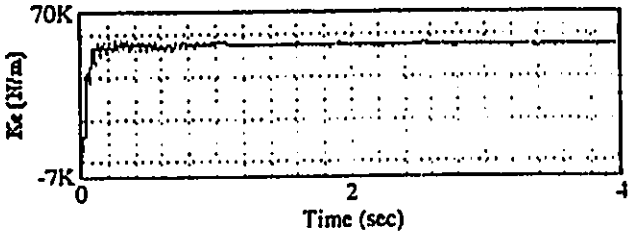
(c) Measured Force / Displacement



(d) Identified Inertia Coefficient



(e) Identified Damping Coefficient



(f) Identified Spring Coefficient

Fig.6.9 Identification Result of first 4sec for Rubber (N.N.)

This identification method can be improved following further study for the efficient tuning of the neural network. In this thesis the focus is on the comparative study of identification methods for environment impedance.

6.3. Force Control by the Neural Networks Controller

The neural network controller introduced in Chapter 4.5. controls the force applied to the environment, y position and the angle of the end-effector of the robot in Cartesian coordinates (see Figure 3.1). The purpose of this experiment is to verify the possibility and the effectiveness of the neural network controller for force control of the robot manipulator in contact with an unknown environment. For the unknown environment, Styrofoam and Rubber are used in this study. As we explained in the Chapter 6.1. and 6.2., the gains of the hybrid controller for environment impedance identification had to be changed depending on the property of the environment. However, the gains of the neural network controller do not have to be changed since this controller is adjusted to unknown environment. The gains in the equation (4.19) and (4.20) of the hybrid control part of the neural network controller are chosen as the same as the gains used for the above mentioned hybrid controller for the Styrofoam environment impedance identification

$$K_{pp} = 400.0 \, , \, K_{pv} = 80.0 \, , \, K_{fp} = 0.25$$

In this experiment, these gains are used for both Styrofoam and Rubber environment. In the force control, 120Hz (bandwidth of the force sensor) 2nd-order Butterworth low-pass filter is used for the measured force after sampling to the dSPACE to get rid of noise.

In order to show the neural networks effect in the neural networks controller, the force response of the robot manipulator by the hybrid controller without the neural networks is tested with the gains shown above. The experimental force response result is shown in Figure 6.10. The desired force (target force) is changed every 2.5 seconds between 10N and 20N. Since the gain for force control is low and there is no error integral (the hybrid

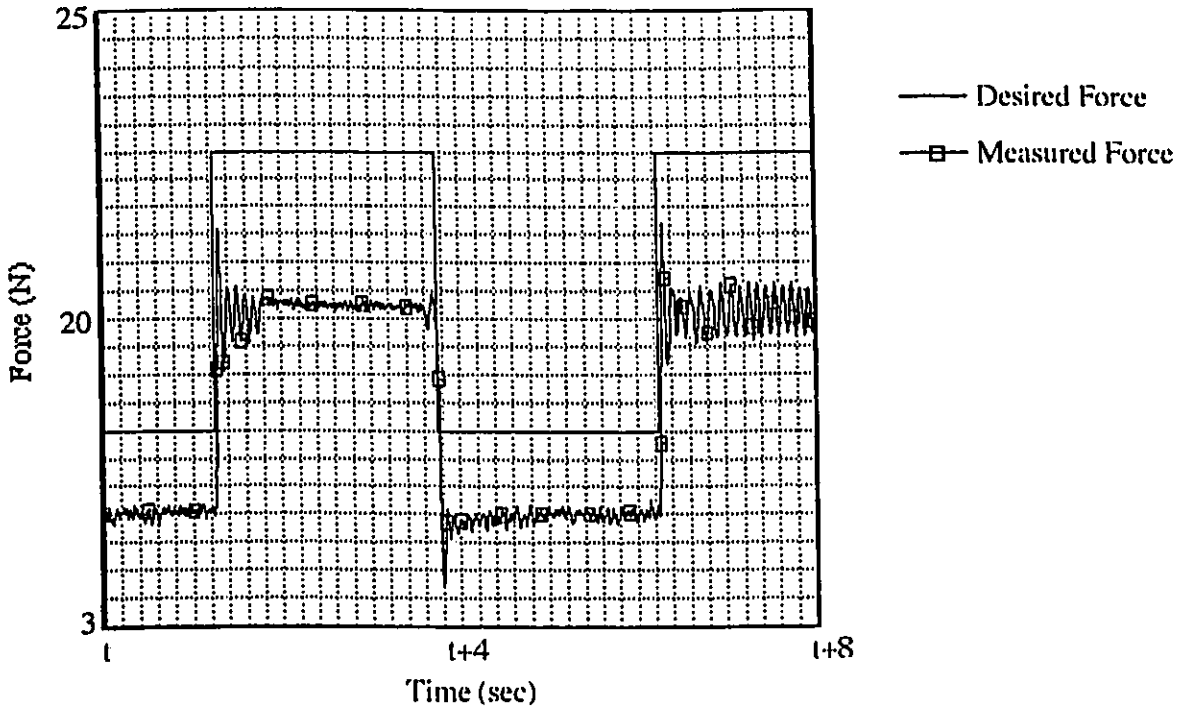


Fig.6.10 Force Response of the Hybrid Controller for Styrofoam

controller for environment impedance identification used error integral), the applied force (measured force) can not reach the desired force.

Figure 6.11 shows a 4sec time slot of the force response result of the robot manipulator for Styrofoam with the neural networks controller about 10 seconds after the experiment is started. The desired force (target force) is changed every 2.5 seconds between 10N and 20N. The learning rate η of the neural network is chosen 0.00005 after some experiment. The result shows that the robot manipulator follows acceptable well the desired force.

Figure 6.12 shows another experimental result for Styrofoam. The desired force is combination of step signal which is changed every 2.5 seconds between 10N and 20N and sinusoidal signal $4\sin(\pi t)$. The learning rate is same

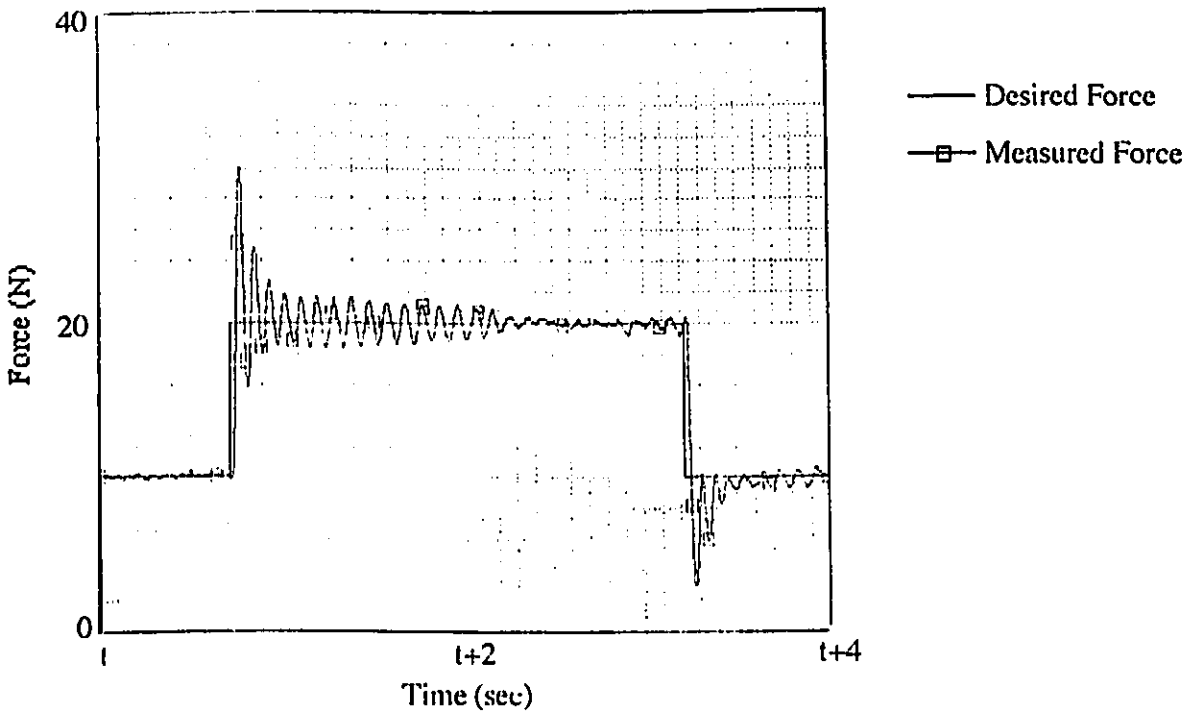


Fig.6.11 Force Response of the Neural Network Controller for Styrofoam - N0.1

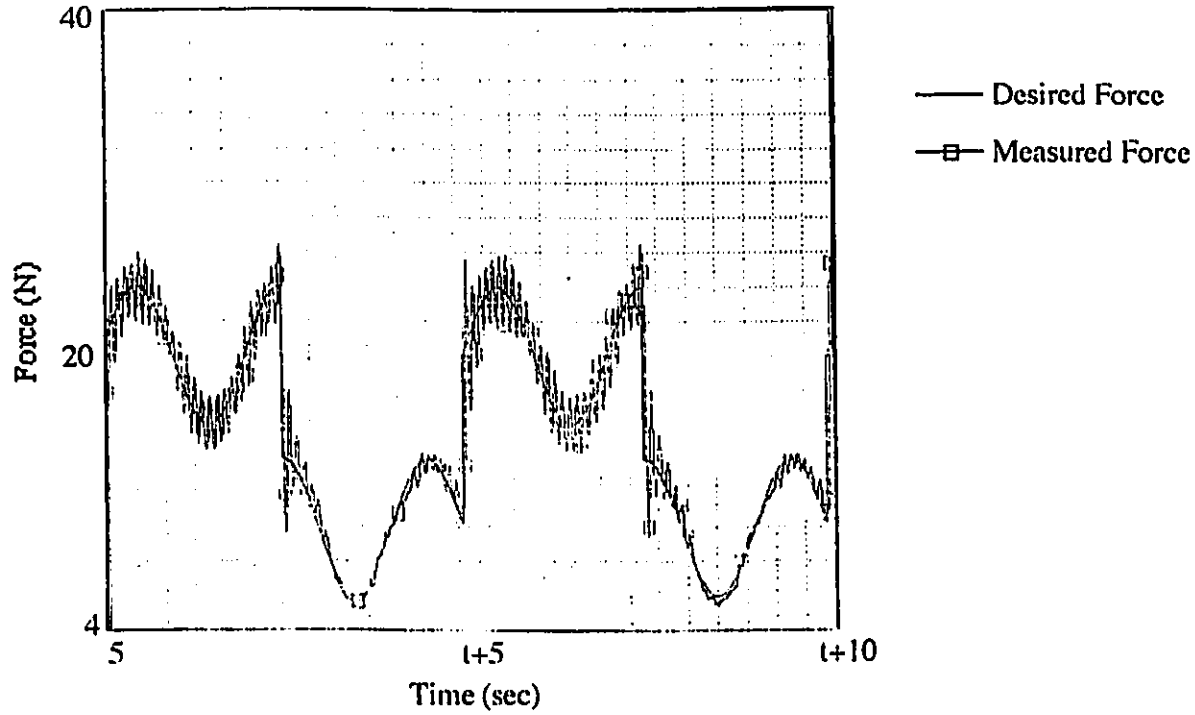


Fig.6.12 Force Response of the Neural Network Controller for Styrofoam - N0.2

as before. The result also shows that the robot manipulator follows acceptable well the desired force.

If the learning rate η is too small, the neural network controller can not adapt immediately to the desired force which changes every 2.5 seconds between 10N and 20N. Figure 6.13 shows the example of force response of the robot manipulator with a small learning rate (η = 0.00001). It can be seen that the force response can not follow the desired force immediately. However, because of the neural network ability of learning, the force output of the robot manipulator is able to follow the desired force after a few minutes, as shown in Figure 6.14. This means even if we use a small learning rate for the neural network, the desired force response can be obtained at a later time.

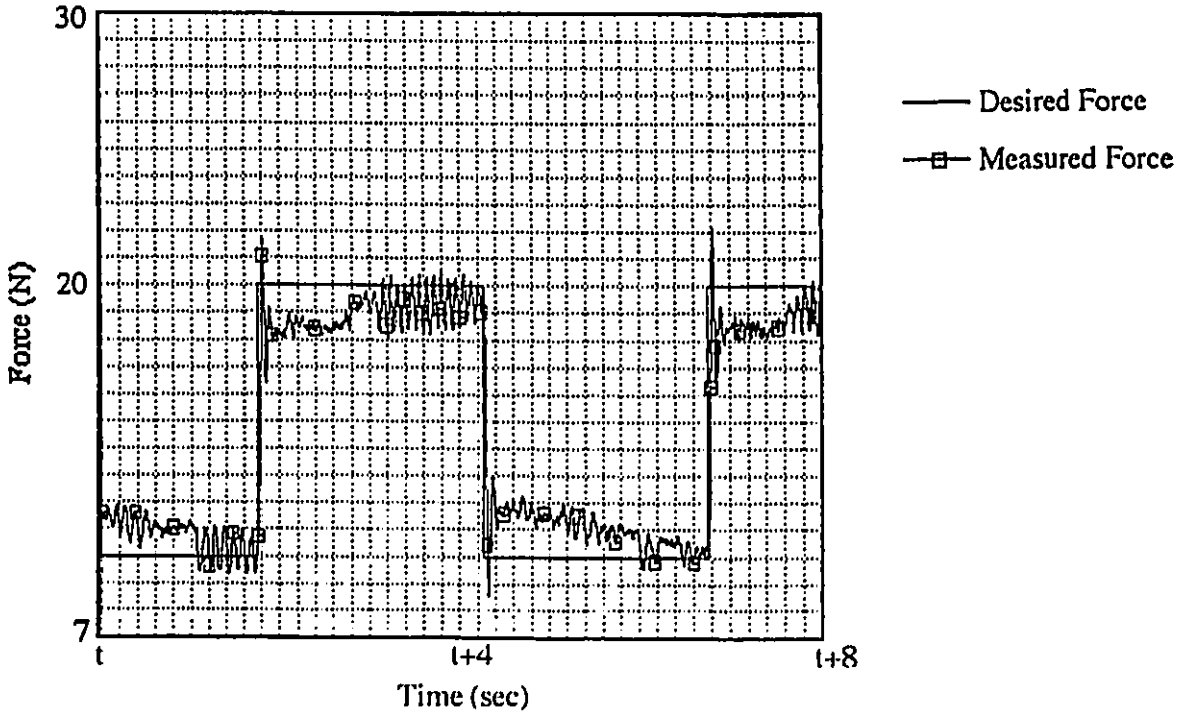


Fig.6.13 Force Response with a small learning rate

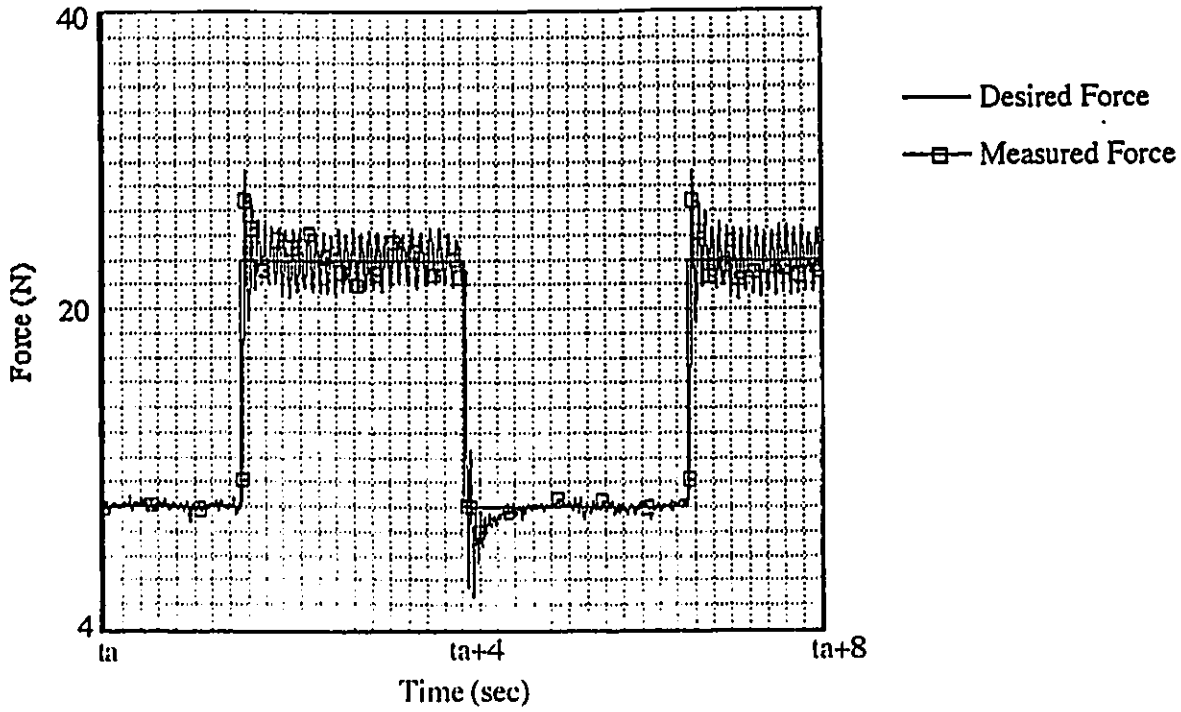
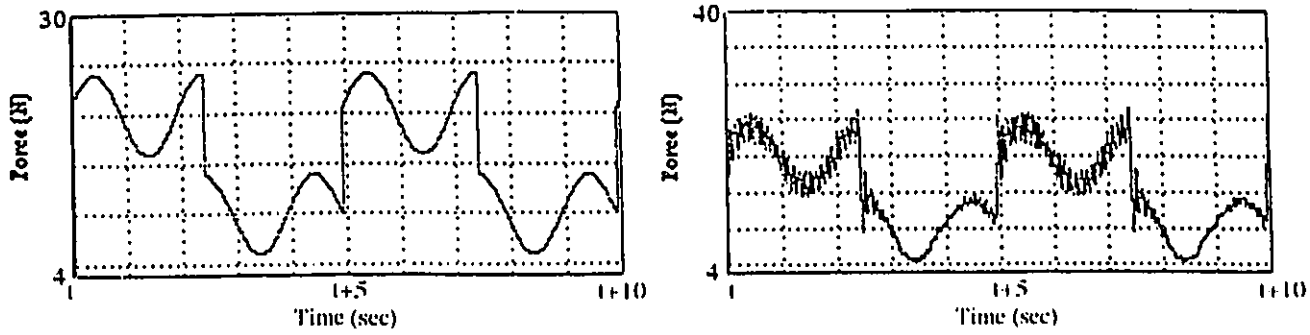


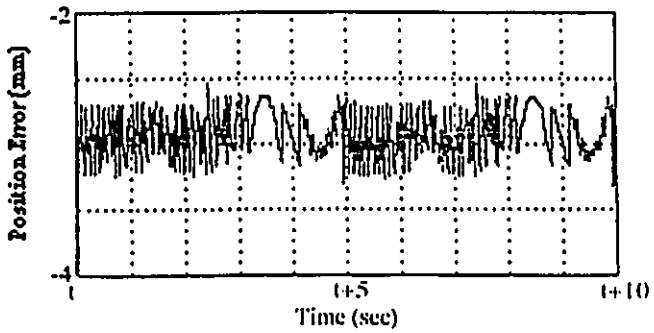
Fig.6.14 Force Response with a small learning rate after a few minutes

Figure 6.15 shows the desired force, measured force, measured y position and measured angle of the end-effector of the robot manipulator for Styrofoam environment. In this case, the desired y position and the desired angle are always zero. It can be seen, however, that the measured y position and angle are not zero. There are two major reasons. First, the position control which is executed before the neural network controller works is not accurate and therefore, there are some errors in y position and the angle of the robot manipulator before the neural network controller is executed. Second, the environment used for this experiment is soft and once force is applied, the end-effector sticks on the environment. In such a situation, it is difficult to position the end-effector accurately.

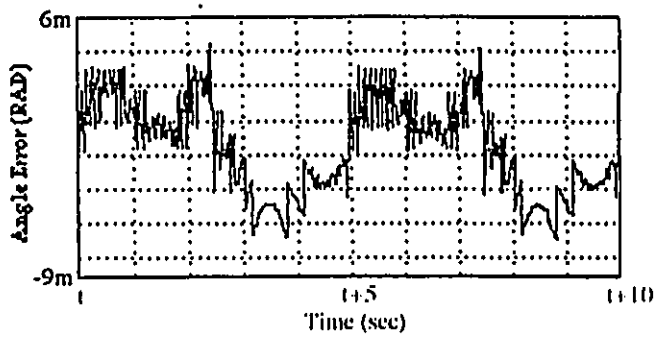


(a) Desired Force

(b) Measured Force



(c) y Position Error



(d) Angle Error

Fig.6.15 Experimental Result of the Neural Network Controller

Figure 6.16 shows the force response of the robot manipulator for Rubber environment with the neural network controller about 10 seconds after the experiment is started. The desired force (target force) is changed every 2.5 seconds between 10N and 20N as same as before. The learning rate η of the neural network is 0.00005. This result also shows the robot manipulator follows pretty well to the desired force.

Figure 6.17 shows the beginning of another experiment result for Rubber. The desired force is a combination of a step signal, which is changed every 2.5 seconds between 10N and 20N, and a sinusoidal signal $4\sin(\pi t)$. The learning rate is the same as before. The result also shows that the robot manipulator follows well to the desired force.

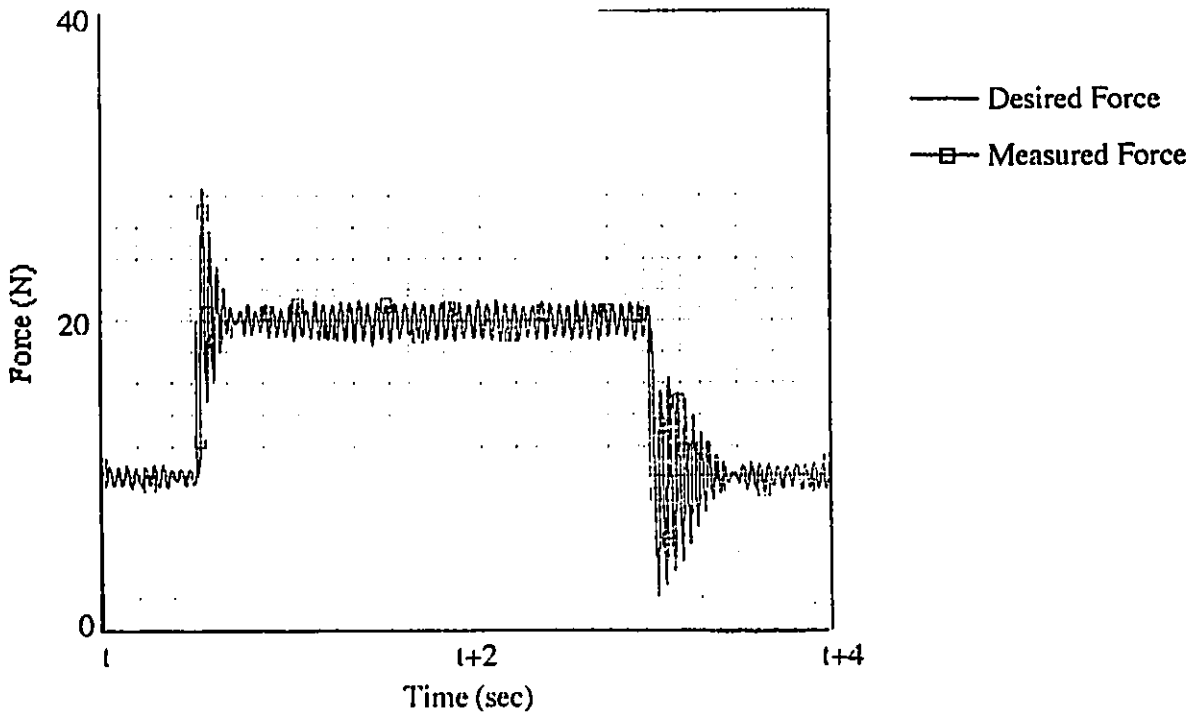


Fig.6.16 Force Response of the Neural Network Controller for Rubber - N0.1

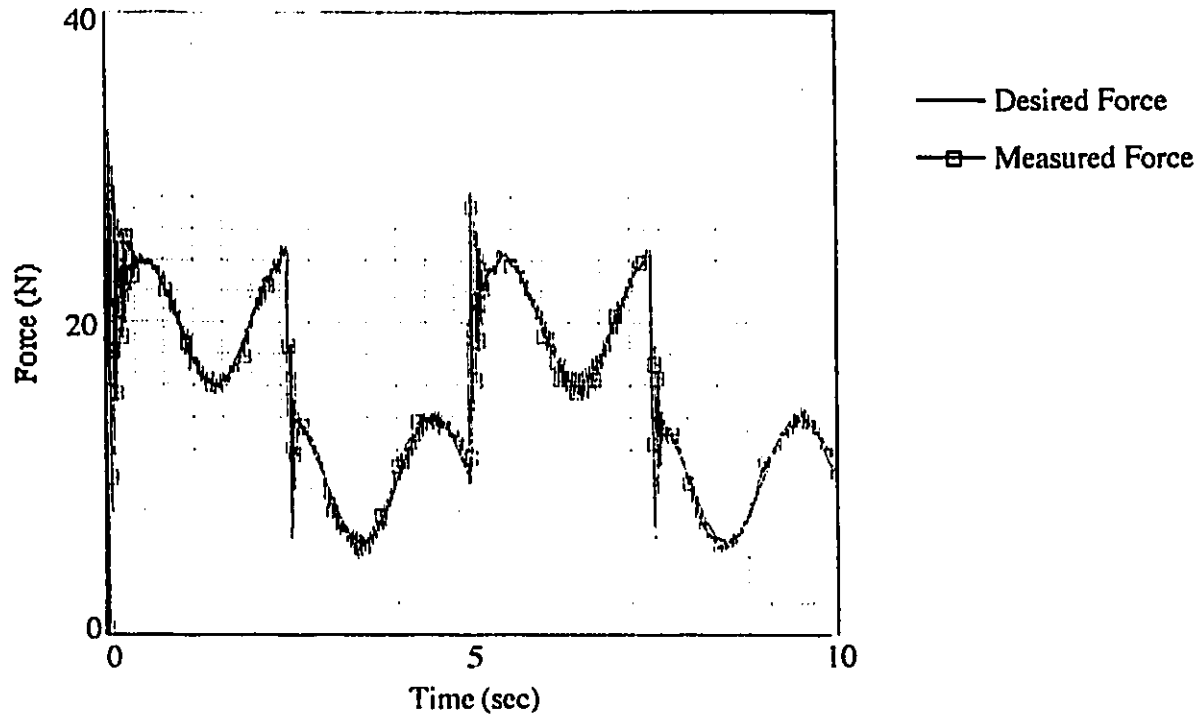


Fig.6.17 Force Response of the Neural Network Controller for Rubber - N0.2

This experimental results also show that the neural network controller can adapt in almost 5 seconds. This adapting time depends, however, on the learning rate of the neural network, as I mentioned before and if the learning rate is small, it takes more time for adaptation.

7. Conclusions

7.1. Conclusion for Environment Impedance Identification

Two kinds of environment impedance identification methods are analysed in this thesis. The identification of the spring coefficient Ke, both methods give realistic results. For damping coefficient Be, both methods seem to give realistic values. For inertia coefficient Me, the least square method gives a negative value while the neural networks give a realistic result. Therefore the identification method using neural networks seem to be able to give suitable values for the coefficients compared to the identification method using least square estimation.

The coefficients with the least square method are obtained, however, faster than with neural network method. The input signals to the neural networks have to be similar values to actual input signals and scaled for the similar order of magnitude, which sometimes becomes an inconvenience, especially for the identification of unknown environment.

This thesis shows that the feasibility of two kinds of on-line environment impedance identification methods for practical use, though both of them have a little drawback. It depends on the situation that which method is preferred to be used. For example, if identification of the inertia and damping coefficients is not very important and quicker identification is required, the identification method using least square estimation is preferred, and if identification of all coefficients is important, the identification method using neural networks is preferred.

7.2. Conclusion and Future Work for Neural Network Controller

The application of neural networks to robot control is proposed and experimentally tested in this thesis. The experimental results show that the neural

networks controller proposed in this thesis is able to obtain the desired force in Cartesian coordinates. This ability is not only for fixed desired force, but also for time varied desired force. However, this thesis analyses only the feasibility and the effectiveness of the application of neural networks to robot control in order to obtain desired force in Cartesian coordinates and the search of a better neural networks controller is not main purpose of this thesis. Therefore it is necessary to study further the best learning rate, how many hidden layers are required and how many neurons are required to perform a better control, since the neural networks controller proposed in this thesis is not the best tuned controller. It is also important to study the stability of the controller and select a better architecture of neural networks from the variety of neural networks.

References

1. McCulloch,W..S., Pitts,W., "A Logical Calculus of the Idea Immanent in Nervous Activity", Bull. Math. Biophysics, 5, 1943, pp.115-133.

2. Luh,J.Y.S., Walker,M.W., Paul,R.P.C., "On-line Computational Scheme for Mechanical Manipulators", ASME Journal of Dynamic Systems, Measurement, and Control, Vol.102, 1980, pp.69-76.

3. Raibert,M.H., Craig,J.J. ,"Hybrid Position/Force Control of Manipulators", Trans. ASME Journal of Dynamic Systems, Measurement, and control, Vol.102, 1981, pp.126-133.

4. Mason,M.T., "Compliance and Force Control for Computer Controlled Manipulators", IEEE Trans. on Systems, Man, and Cybernetics, SMC-11, 6, 1981, pp.418-432.

5. Duffy,J., "The fallacy of Modern Hybrid Control Theory that is Based on Orthogonal Complements of Twist and Wrench Space", Journal of Robotic Systems, Vol.7, No.2, 1990, pp.139-144.

6. Sugimoto,K., "Theoretical Study on Hybrid Control of Position and Force", Journal of the Robotics Society of Japan, Vol.11, No.3, 1993, pp.151-158. (in Japanese)

7. Yoshikawa,T.,Sugie,T.,Tanaka,M., "Dynamic Hybrid Control of Robot Manipulators

- Controller Design and Experiment -", Proc. of IEEE Int. Conf. on Robotics and Automation, 1987, pp.2005-2010.

8. Khatib,O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation", IEEE Journal of Robotics and Automation, Vol.RA-3, No.1, 1987, pp.43-53.

9. Yabuta,T., Chona,A.J., Beni,G., "On the Asymptotic Stability of the Hybrid Position /Force Control Scheme for Robot Manipulators", Proc. of IEEE Int. Conf. on Robotics and Automation, 1988, pp.338-343.

10. Wen,J.T., Murphy,S., "Stability Analysis of Position and Force Control Probrems for Robot Arms", Proc. of IEEE Int. Conf. on Robotics and Automation, 1990, pp.252-257.

11. Yabuta,T., "Nonlinear Basic Stability Concept of the Hybrid Position/Force Control Scheme for Robot Manipulators", IEEE Trans. on Robotics and Automation, Vol.8, No.5, 1992, pp.663-670.

12. Salisbury,J.K., "Active Stiffness Control of Manipulator Fine Motions", Proc. of the 19th IEEE Conf. on Decision and Control, 1980, pp.95-100.

13. Whitney,D.E., "Force Feedback Control of Manipulator Fine Motions", ASME Journal of Dynamic Systems, Measurement, and Control, June 1977, pp.91-97.

14. Hogan,N., "Impedance Control: An Approach to Manipulation: Part I - III", ASME Journal of Dynamic Systems, Measurement, and Control, Vol.107,1985,pp1-24.

15. Widrow,B., "Generalization and Information Storage in Network of Adaline

'Neurons'", Self-Organizing Systems - 1962, pp.435-462, Spartan Books, 1962.

16. Kawato,M., Furukawa,K., Suzuki,R., "A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement", Biological Cybernetics, 57, 1987, pp.169-185.

17. Kawato,M., Uno,Y., Isobe,M., Suzuki,R., "Hierarchical Neural Network Model for Voluntary Movement With Application to Robotics", IEEE Contr. Syst. Mag., Vol.8, 1988, pp.8-16.

18. Narendra,K., Parthasarathy,K., "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. on Neural Networks, Vol.1, No.1, 1990, pp.4-27.

19. Yamada,T., Yabuta,T., "Direct Controller using Neural Network", Trans. SICE'89, 1989, pp.391-392. (in Japanese)

20. Yabuta,T., Yamada,T., "Possibility of Neural Networks Controller for Robot Manipulators", Proc. of IEEE Int. Conf. on Robotics and Automation, 1990, pp.1686-1691.

21. Yamada,T., Yabuta,T., "Some Remarks on Characteristics of Direct Neuro-Controller with Regard to Adaptive Control",Trans. SICE, Vol.27, No.7, 1991, pp.784-791. (in Japanese)

22. Fukuda,T., Kurihara,T., Tokita,M., Mitsuoka,T., "Position and Force Hybrid Control of Robotic Manipulator by Neural Networks (1st Report, Application of Neural Servo Controller to Stabbing Controll)", Trans. JSME, 56-527, C, 1990, pp.210-216. (in Japanese)

23. Fukuda,T., Shibata,T., Tokita,M., Mitsuoka,T., "Position and Force Hybrid Control of Robotic Manipulator by Neural Networks (2nd Report, Control of 2DOF Manipulators by a Neural network having Time Delay Elements)", Trans. JSME, 57-535, C, 1991, pp.866-873. (in Japanese)

24. Fukada,T., Shibata,T., "Neural Network Applications for Robotic Motion Control", Journal of Robotics and Mechatromics, Vol.2, No3, 1991, pp.21-25.

25. Xu,J., Harashima,F., Sim,K., Hashimoto,H., "Resolved Acceleration Position/Force Control of Robot Arm Using Fuzzy Compensation", Journal of the Robotics Society of Japan, Vol.9, No.3, 1991, pp.16-26. (in Japanese)

26. Gomi,H., Kawato,M., "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning", ATR Technical Report, TR-A-0114, 1991, pp.1-35.

27. Cohen,M., Flash,T., "Learning Impedance Parameters for Robot Control Using an Associative Search Network", IEEE Trans. on Robotics and Automation, Vol.7, No.3, 1991, pp382-390.

28. Craig,J.J., **Introduction to ROBOTICS - Mechanics and Control**, Addison-Wesley Publishing Company, 1986.

29. Qian,H.P. , De Schutter,J. , "Introducing Active Linear and Nonlinear Damping to Enable Stable High Gain Force Control in case of Stiff Contact" , Proc. of the 1992 IEEE Int. Conf. on Robotics and Automation, pp.1374-1379.

30. Jackson,L., **Digital Filters and Signal Processing**, Kluwer Academic Publishers, 1986.

31. Astrom,K., Wittenmark,B., **Adaptive Control**, Addison-Wesley Publishing Company, 1989.

32. Kalaycioglu,S., Brown,A., "On-Line Identification of Environmental Parameters for Force Control of Space Robots", CCASI Conference, Nov., 1992.

33. Irie,B., Kawato,M., "Acquisition of Internal Representation by Multi-Layered Perceptrons", Trans. of IEICE, D-II, Vol.J73-D-II, No8, 1990, pp.1173-1178. (in Japanese)

34. Nakano,K., **An Introduction to Neurocomputing**, Corona Publishing Co., 1990. (in Japanese)

35. **TMS320C3x User's Guide**, Texas Instruments., 1990.

36. **DS1002 User's Guide**, dSPACE digital signal processing and control engineering GmbH, 1990.

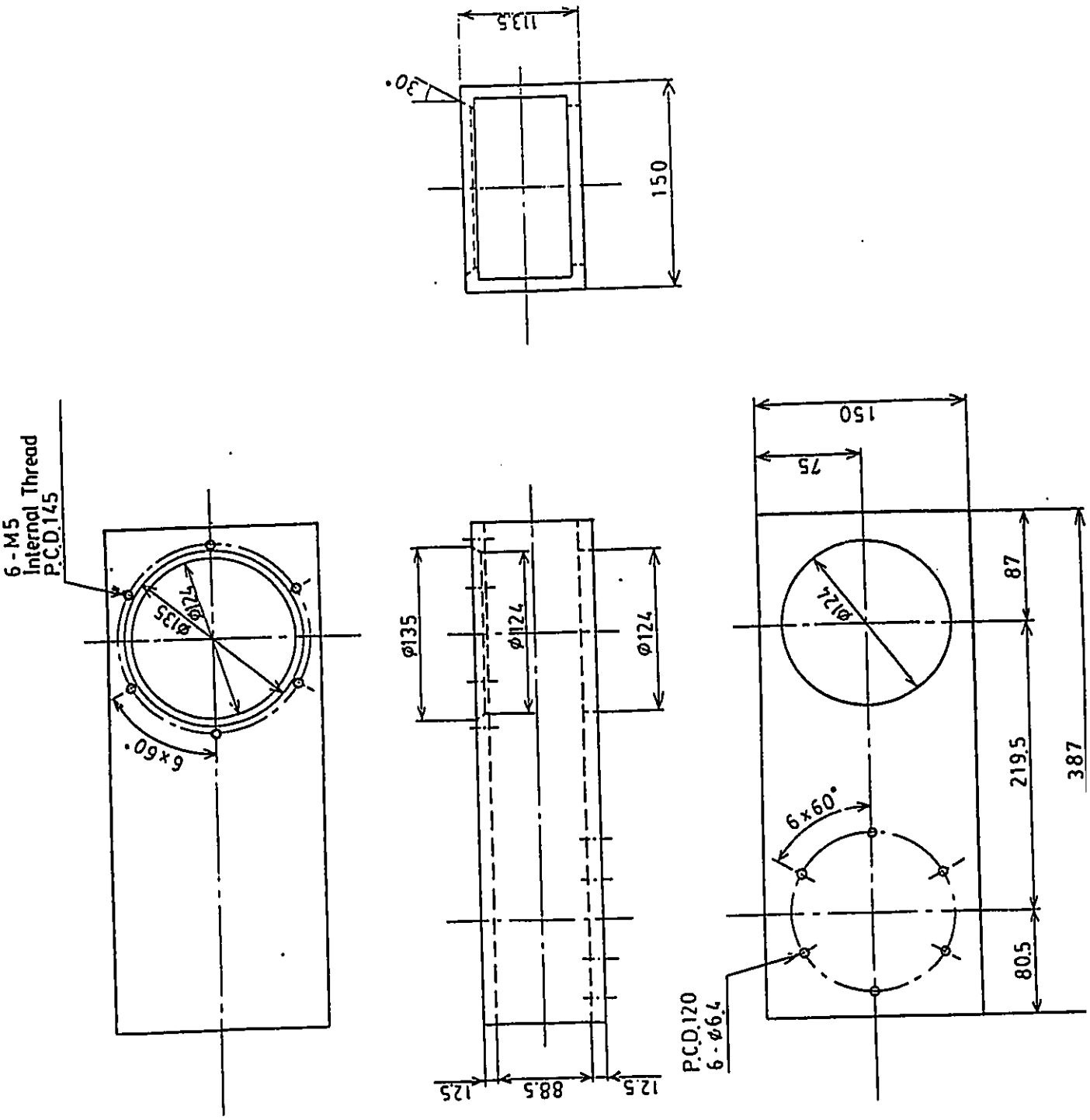
37. **FS6-120A 6-AXIS FORCE SENSOR SOFTWARE MANUAL**, Barry Wright Co., 1982.

38. Hostetter,G., **Engineering Network Analysis**, Library of Congress Catalog-ing In Publication Data, 1984.

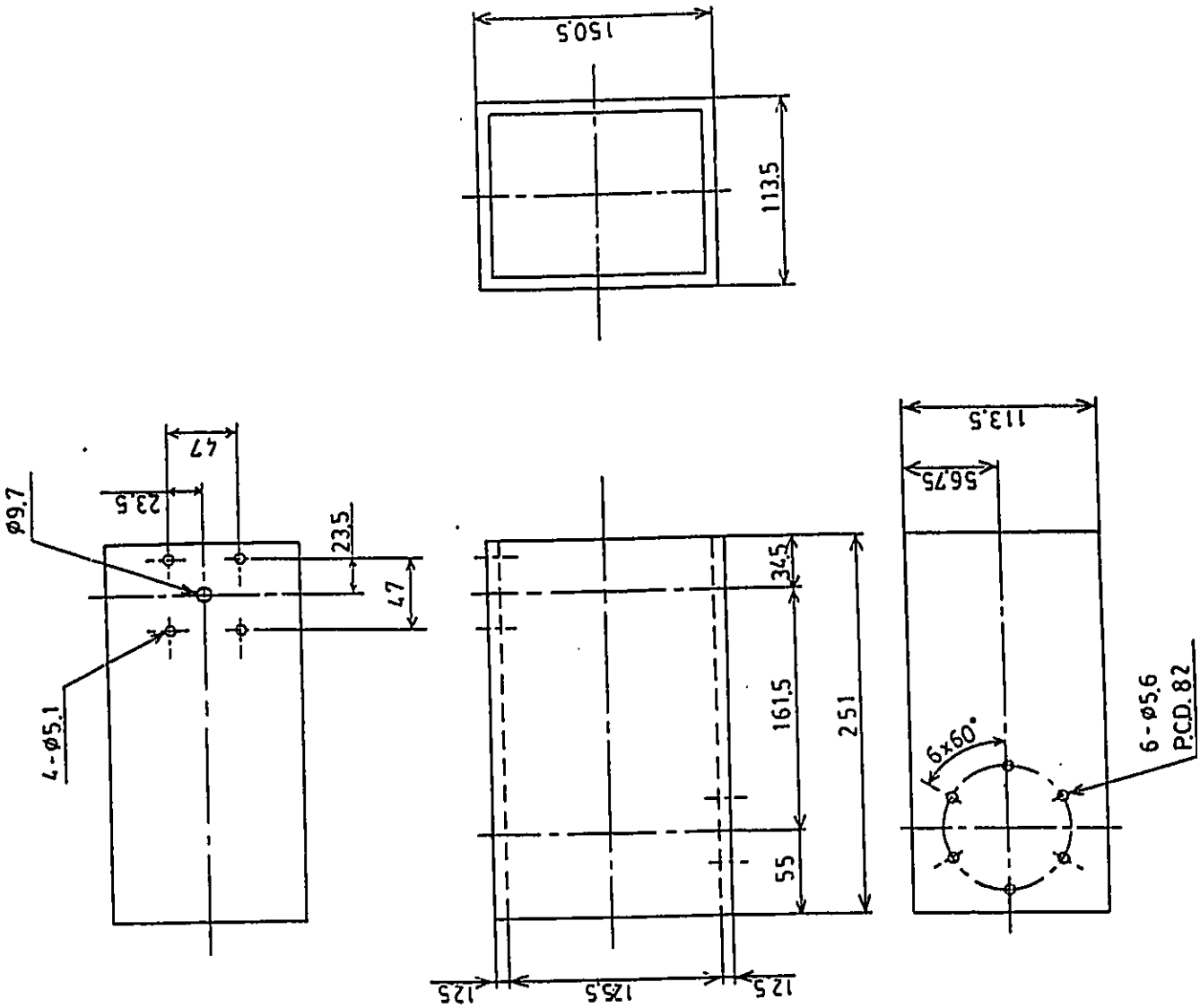
APPENDIX A:

MANIPULATOR LINK DIMENSIONS

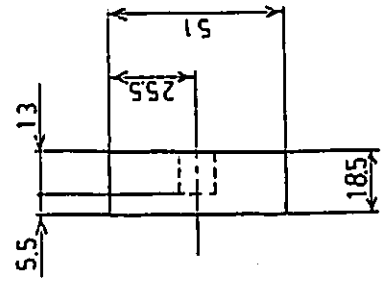
Link 1

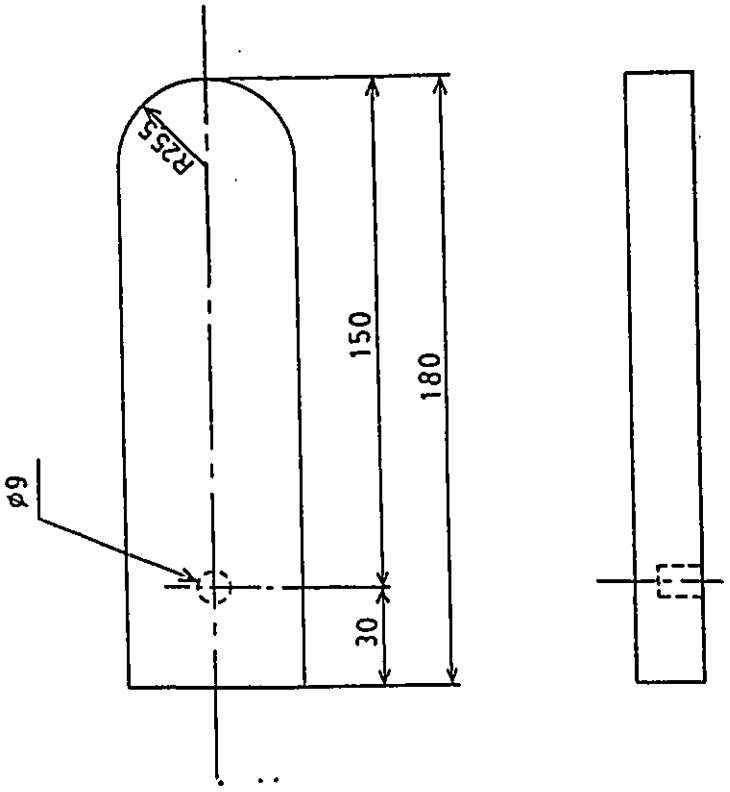

113.5
30°
150
6-M5
Internal Thread
P.C.D.145
φ135
φ124
6×60°
φ135
φ124
φ124
12.5
88.5
12.5
150
75
87
φ124
219.5
387
6×60°
80.5
P.C.D.120
6-φ6.4

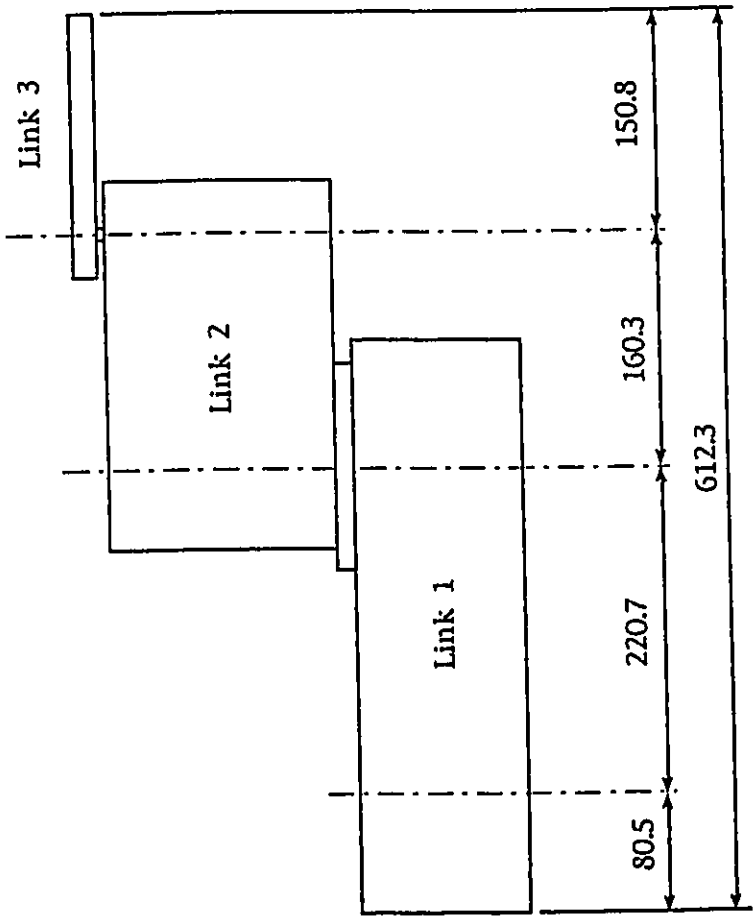
Link 2


150.5
113.5
47
23.5
ø9.7
23.5
47
4-ø5.1
34.5
161.5
251
55
125
125.5
125
113.5
56.75
6x60°
6-ø5.6
P.C.D. 82

Link 3







Dimension of the Robot Manipulator

APPENDIX B:

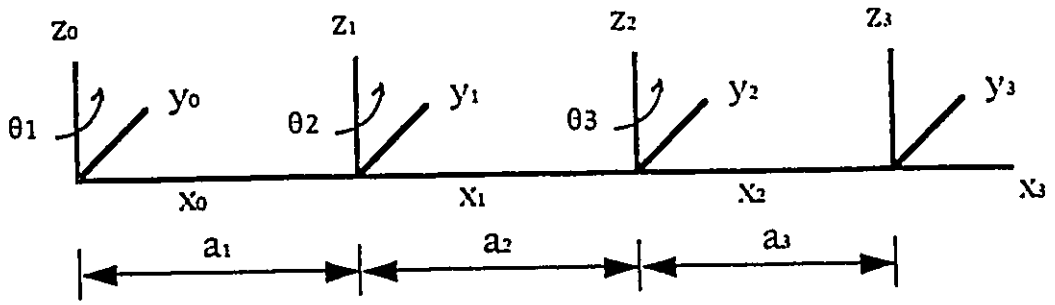
SPECIFICATION OF MOTORS

Item	NSK 0408	NSK 0810	System 7 7521.250
Maximum Torque	9.8 Nm	88.2 Nm	2 Nm
Maximum Friction Torque	98 Ncm	441 Ncm	-
Axial Load Capacity	1764 N	4508 N	-
Moment Load Capacity	19.6 Nm	78.4 Nm	-
Rotor Inertia	0.009 $kgfm^2$	0.084 $kgfm^2$	0.28 $kgcm^2$
Weight	6.5 kgf	24 kgf	1.9 kgf
Resolver Resolution	614,400 or 153,600 count/rev	409,600 or 102,400 count/rev	2,000 count/rev
Environment	0 - 40° C operating temperature 20 - 80% humidity		-

APPENDIX C:

MANIPULATOR KINEMATICS

C1. Manipulator Kinematics



D-H Transformations
Table of Link Parameters

variable	link-i	frame	θi	di	ai	αi	Cθ	Sθ	Cα	Sα
θ1	1	0 - 1	θ1	0	a1	0	Cθ1	Sθ1	1	0
θ2	2	1 - 2	θ2	0	a2	0	Cθ2	Sθ2	1	0
θ3	3	2 - 3	θ3	0	a3	0	Cθ3	Sθ3	1	0

$$A_i = Rz(\theta_i)Trans(0,0,d_i)Trans(a_i,0,0)Rx(\alpha_i)$$

$$= \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha & aC\theta \\ S\theta & C\theta C\alpha & -C\theta S\alpha & aS\theta \\ 0 & S\alpha & C\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1 = T_1 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & a_1C\theta_1 \\ S\theta_1 & C\theta_1 & 0 & a_1S\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}1)$$

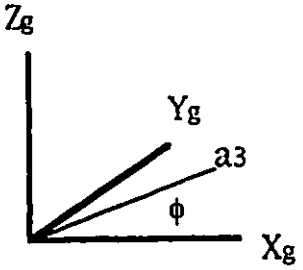
$$A_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & a_3C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & a_3S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = A_1A_2 = \begin{bmatrix} C_{12} & -S_{12} & 0 & a_2C_{12}+a_1C_1 \\ S_{12} & C_{12} & 0 & a_2S_{12}+a_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}2)$$

$$T_3 = T_2A_3 = \begin{bmatrix} C_{123} & -S_{123} & 0 & a_3C_{123}+a_2C_{12}+a_1C_1 \\ S_{123} & C_{123} & 0 & a_3S_{123}+a_2S_{12}+a_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}3)$$

Task Space Values

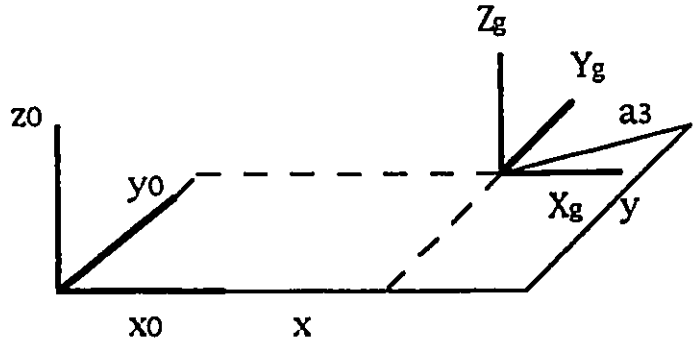


$$\text{Euler } G(\phi,\theta,\psi) = R_z(\phi)R_y(\theta)R_z(\psi)$$

$$\theta \,\&\, \psi = 0$$

$$\text{Euler } G(\phi,\theta,\psi) = R_z(\phi) = \begin{bmatrix} C\phi & -S\phi & 0 & 0 \\ S\phi & C\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_3 = \left[\text{Trans}(x,y,z)\right]\left[\text{Euler } G\right] = \begin{bmatrix} C\phi & -S\phi & 0 & x \\ S\phi & C\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}4)$$

$$T_2 = T_3 A_3^{-1} = \begin{bmatrix} C_{12} & -S_{12} & 0 & -a_3C\phi+x \\ S_{12} & C_{12} & 0 & -a_3S\phi+y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}5)$$

$$T_1 = T_2 A_2^{-1} = A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & -a_2C_{12}-a_3C\phi+x \\ S_1 & C_1 & 0 & -a_2S_{12}-a_3S\phi+y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}6)$$

Equate T_3 (Eqn.(1-3)&(1-4)) t_{11}

$$\boxed{\theta_1+\theta_2+\theta_3 = \phi} \qquad (1\text{-}7)$$

Equate T_2 (Eqn.(1-2)&(1-5)) t_{14} & t_{24}

$$a_2C_{12}+a_1C_1 = -a_3C\phi+x$$
$$a_2S_{12}+a_1S_1 = -a_3S\phi+y$$

square these two equations,

$$a_2^2C_{12}^2 + 2a_1a_2C_{12}C_1 + a_1^2C_1^2 = (-a_3C\phi+x)^2$$
$$a_2^2S_{12}^2 + 2a_1a_2S_{12}S_1 + a_1^2S_1^2 = (-a_3S\phi+y)^2$$

add these two equations,

$$a_2^2+a_1^2+2a_1a_2C_2 = (-a_3C\phi+x)^2+(-a_3S\phi+y)^2$$

$$C_2 = \frac{-a_2^2-a_1^2+(-a_3C\phi+x)^2+(-a_3S\phi+y)^2}{2a_1a_2}$$

$$\boxed{\theta_2 = \text{acos}\left(\frac{-a_2^2-a_1^2+(-a_3C\phi+x)^2+(-a_3S\phi+y)^2}{2a_1a_2}\right)} \qquad (1\text{-}8)$$

Equate T_1 (Eqn.(1-1)&(1-6)) t_{14} & t_{24}

$$-a_2C_{12}-a_3C\phi+x = a_1C_1$$
$$-a_2S_{12}-a_3S\phi+y = a_1S_1$$

then

$$(-a_1-a_2C_2)C_1+a_2S_1S_2 = a_3C\phi-x$$
$$(-a_1-a_2C_2)S_1-a_2C_1S_2 = a_3S\phi-y$$

So

$$\left(-a_1-a_2C_2+\frac{a_3C\phi-x}{a_3S\phi-y}a_2S_2\right)C_1 = \left(-a_2S_2+\frac{a_3C\phi-x}{a_3S\phi-y}(-a_1-a_2C_2)\right)S_1$$

$$\frac{S_1}{C_1} = \frac{((a_3S\phi-y)(-a_1-a_2C_2)+(a_3C\phi-x)a_2S_2)}{((a_3C\phi-x)(-a_1-a_2C_2)-(a_3S\phi-y)a_2S_2)}$$

$$\boxed{\theta_1 = \text{atan2}(((a_3S\phi-y)(-a_1-a_2C_2)+(a_3C\phi-x)a_2S_2),((a_3C\phi-x)(-a_1-a_2C_2)-(a_3S\phi-y)a_2S_2))} \quad (1\text{-}9)$$

From Eqn.(1-7),(1-8),(1-9),

$$\boxed{\theta_3 = \phi-\theta_1-\theta_2} \quad (1\text{-}10)$$

C2. Manipulator Jacobian

$$x = a_3C_{123}+a_2C_{12}+a_1C_1$$
$$y = a_3S_{123}+a_2S_{12}+a_1S_1$$
$$\phi = \theta_1+\theta_2+\theta_3$$

$$J = \begin{bmatrix} -a_3S_{123}-a_2S_{12}-a_1S_1 & -a_3S_{123}-a_2S_{12} & -a_3S_{123} \\ a_3C_{123}+a_2C_{12}+a_1C_1 & a_3C_{123}+a_2C_{12} & a_3C_{123} \\ 1 & 1 & 1 \end{bmatrix} \quad (2\text{-}1)$$

The inverse Jacobian matrix is

$$J^{-1} = \frac{1}{a_1a_2S_2}\begin{bmatrix} a_2C_{12} & a_2S_{12} & a_3a_2S_3 \\ -a_2C_{12}-a_1C_1 & -a_3S_{12}-a_1S_1 & -a_3(a_2S_3+a_1S_{23}) \\ a_1C_1 & a_1S_1 & a_3a_1S_{23}+a_1a_2S_2 \end{bmatrix} \quad (2\text{-}2)$$

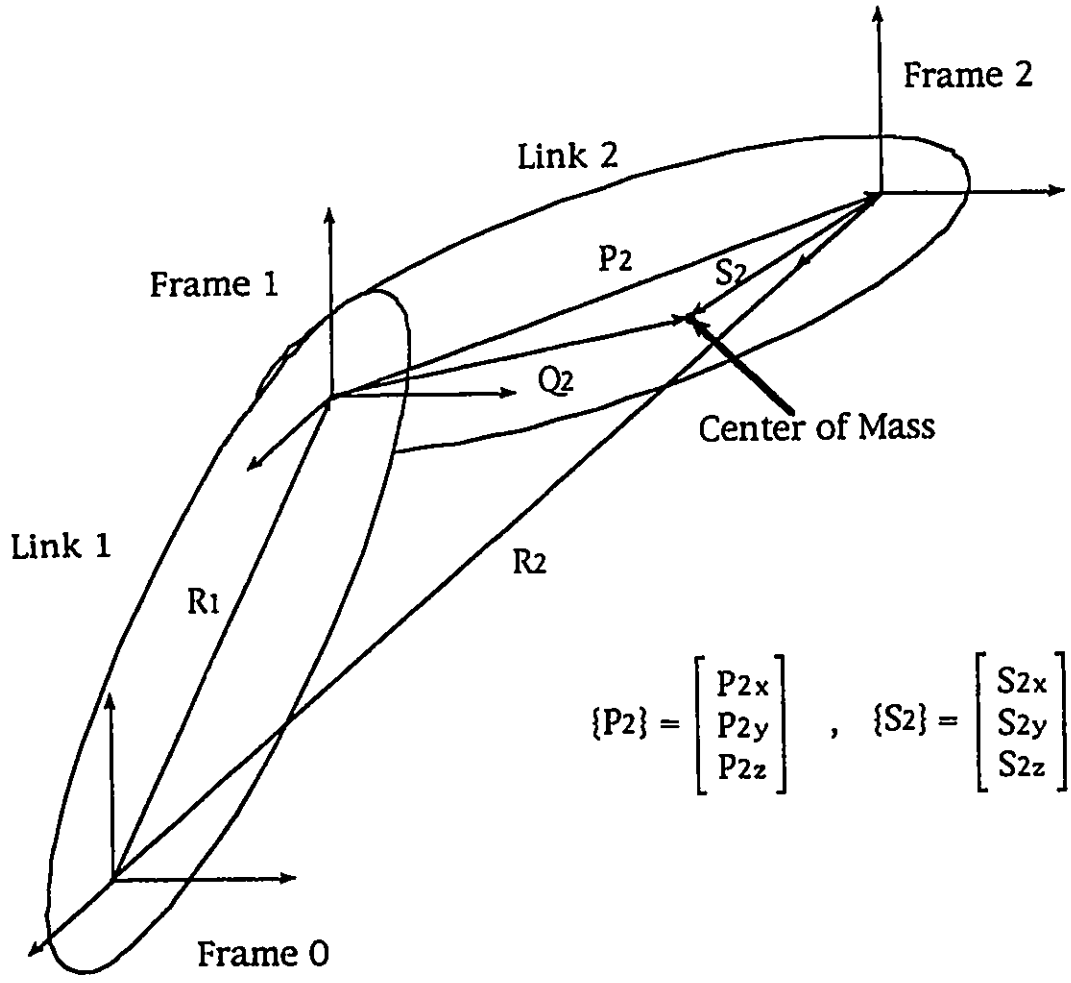
Derivative of Jacobian matrix is

$$\dot{J} = \begin{bmatrix} -a_3C_{123}-a_2C_{12}-a_1C_1 & -a_3C_{123}-a_2C_{12} & -a_3C_{123} \\ -a_3S_{123}-a_2S_{12}-a_1S_1 & -a_3S_{123}-a_2S_{12} & -a_3S_{123} \\ 0 & 0 & 0 \end{bmatrix} \quad (2\text{-}3)$$

APPENDIX D:

MANIPULATOR DYNAMICS

Notation



$$\{P_2\} = \begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \end{bmatrix} \quad , \quad \{S_2\} = \begin{bmatrix} S_{2x} \\ S_{2y} \\ S_{2z} \end{bmatrix}$$

Manipulator Dynamics

Manipulator dynamics can be obtained by solving Recursive Newton-Euler Equations.

Definitions

$$[Ii] = \begin{bmatrix} Ixx & -Ixy & -Ixz \\ -Iyx & Iyy & -Iyz \\ -Izx & -Izy & Izz \end{bmatrix} \qquad [Dev\ C] = \begin{bmatrix} 0 & -Cz & Cy \\ Cz & 0 & -Cx \\ -Cy & Cx & 0 \end{bmatrix}$$

$$[Dev\ \Delta i] = \begin{bmatrix} 0 & -\dot{\Omega}z & \dot{\Omega}y \\ \dot{\Omega}z & 0 & -\dot{\Omega}x \\ -\dot{\Omega}y & \dot{\Omega}x & 0 \end{bmatrix} + \begin{bmatrix} 0 & -\Omega z & \Omega y \\ \Omega z & 0 & -\Omega x \\ -\Omega y & \Omega x & 0 \end{bmatrix} \begin{bmatrix} 0 & -\Omega z & \Omega y \\ \Omega z & 0 & -\Omega x \\ -\Omega y & \Omega x & 0 \end{bmatrix}$$

Starting Values for Forward Equation

Angular velocity

$$\Omega 0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Angular acceleration

$$\dot{\Omega}0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Velocity of Origin of Frame 0

$$V0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Acceleration of Origin of Frame 0

$$\dot{V}0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

D1. Forward Equations (Link 1)

D1-1. Angular velocity of link 1

$$\Omega 1 = [Rot1]^t(\{\Omega 0\} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}) = \begin{bmatrix} C1 & S1 & 0 \\ -S1 & C1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

D1-2. Angular acceleration of link 1

$$\dot{\Omega}_1 = [Rot1]^t(\{\dot{\Omega}_0\} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} + [Dev\ \Omega_0] \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix})$$

$$= \begin{bmatrix} C1 & S1 & 0 \\ -S1 & C1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix}$$

D1-3. Acceleration of origin of Frame 1

$$\dot{V}_1 = [Rot1]^t\{\dot{V}_0\} + [Dev\ \Delta_1]\ \{P_1\}$$

$$= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \begin{bmatrix} 0 & -\ddot{\theta}_1 & 0 \\ \ddot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\theta}_1 & 0 \\ \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -\dot{\theta}_1 & 0 \\ \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2 \\ a_1\ddot{\theta}_1 \\ -g \end{bmatrix}$$

D1-4. Acceleration of the center of mass of link 1

$$a_1 = \{\dot{V}_1\} + [Dev\ \Delta_1] \begin{bmatrix} S_{1x} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2 \\ a_1\ddot{\theta}_1 \\ -g \end{bmatrix} + \begin{bmatrix} 0 \\ S_{1x}\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} -S_{1x}\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -(a_1+S_{1x})\dot{\theta}_1^2 \\ (a_1+S_{1x})\ddot{\theta}_1 \\ -g \end{bmatrix}$$

D1-5. Effective force

$$F_1 = m_1\{a_1\} = \begin{bmatrix} -m_1(a_1+S_{1x})\dot{\theta}_1^2 \\ m_1(a_1+S_{1x})\ddot{\theta}_1 \\ -m_1 g \end{bmatrix}$$

D1-6. Effective torque

$$N_1 = [I_1]\{\dot{\Omega}_1\} + [\text{Dev}\,\Omega_1]\,[I_1]\{\Omega_1\}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{1zz} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\theta}_1 & 0 \\ \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ I_{1zz}\dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I_{1zz}\ddot{\theta}_1 \end{bmatrix}$$

D2. Forward Equations (Link 2)

D2-1. Angular velocity of link 2

$$\Omega_2 = [\text{Rot2}]^t(\{\Omega_1\} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix}) = \begin{bmatrix} C2 & S2 & 0 \\ -S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix}$$

D2-2. Angular acceleration of link 2

$$\dot{\Omega}_2 = [\text{Rot2}]^t(\{\dot{\Omega}_1\} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_2 \end{bmatrix} + [\text{Dev}\,\Omega_1]\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix})$$

$$= \begin{bmatrix} C2 & S2 & 0 \\ -S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 \end{bmatrix}$$

D2-3. Acceleration of origin of Frame 2

$$\dot{V}_2 = [Rot2]^t\{\dot{V}_1\} + [Dev\ \Delta 2]\ \{P_2\}$$

$$= \begin{bmatrix} C2 & S2 & 0 \\ -S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -a_1\dot{\theta}_1^2 \\ a_1\ddot{\theta}_1 \\ -g \end{bmatrix} + \begin{bmatrix} 0 & -(\ddot{\theta}_1+\ddot{\theta}_2) & 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ 0 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ \dot{\theta}_1+\dot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ \dot{\theta}_1+\dot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-a_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ a_1\dot{\theta}_1^2S2+a_1\ddot{\theta}_1C2+a_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ -g \end{bmatrix}$$

D2-4. Acceleration of the center of mass of link 2

$$a_2 = \{\dot{V}_2\} + [Dev\ \Delta 2] \begin{bmatrix} S_{2x} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-a_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ a_1\dot{\theta}_1^2S2+a_1\ddot{\theta}_1C2-a_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ -g \end{bmatrix} + \begin{bmatrix} 0 & -(\ddot{\theta}_1+\ddot{\theta}_2) & 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} S_{2x} \\ 0 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ \dot{\theta}_1+\dot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ \dot{\theta}_1+\dot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} S_{2x} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-(a_2+S_{1x})(\dot{\theta}_1+\dot{\theta}_2)^2 \\ a_1\dot{\theta}_1^2S2+a_1\ddot{0}_1C2+(a_2+S_{1x})(\ddot{\theta}_1+\ddot{\theta}_2) \\ -g \end{bmatrix}$$

D2-5. Effective force

$$F_2 = m_2\{a_2\} = m_2\begin{bmatrix} -a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-(a_2+S_{1x})(\dot{\theta}_1+\dot{\theta}_2)^2 \\ a_1\dot{\theta}_1^2S2+a_1\ddot{\theta}_1C2+(a_2+S_{1x})(\ddot{\theta}_1+\ddot{\theta}_2) \\ -g \end{bmatrix}$$

$$= \begin{bmatrix} m_2(-a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-(a_2+S_{1x})(\dot{\theta}_1+\dot{\theta}_2)^2) \\ m_2(a_1\dot{\theta}_1^2S2+a_1\ddot{\theta}_1C2+(a_2+S_{1x})(\ddot{\theta}_1+\ddot{\theta}_2)) \\ -m_2g \end{bmatrix}$$

D2-6. Effective torque

$$N_2 = [I_2]\{\dot{\Omega}_2\} + [\text{Dev }\Omega_2]\,[I_2]\{\Omega_2\}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{2zz} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ \dot{\theta}_1+\dot{\theta}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ I_{2zz}(\dot{\theta}_1+\dot{\theta}_2) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ I_{2zz}(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix}$$

D3. Forward Equations (Link 3)

D3-1. Angular velocity of link 3

$$\Omega_3 = [\text{Rot3}]^t(\{\Omega_2\} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix}) = \begin{bmatrix} C3 & S3 & 0 \\ -S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 \end{bmatrix}$$

D3-2. Angular acceleration of link 3

$$\dot{\Omega}_3 = [\mathrm{Rot3}]^t(\{\dot{\Omega}_2\} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_3 \end{bmatrix} + [\mathrm{Dev}\ \Omega_2] \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix})$$

$$= \begin{bmatrix} C3 & S3 & 0 \\ -S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3 \end{bmatrix}$$

D3-3. Acceleration of origin of Frame 3

$$\dot{V}_3 = [\mathrm{Rot3}]^t\{\dot{V}_2\} + [\mathrm{Dev}\ \Delta_3]\ \{P_3\}$$

$$= \begin{bmatrix} C3 & S3 & 0 \\ -S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -a_1\dot{\theta}_1^2C2+a_1\ddot{\theta}_1S2-a_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ a_1\dot{\theta}_1^2S2+a_1\ddot{\theta}_1C2+a_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ -g \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) & 0 \\ \ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ 0 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2C23+a_1\ddot{\theta}_1S23-a_2C3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2S3(\ddot{\theta}_1+\ddot{\theta}_2)-a_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 \\ a_1\dot{\theta}_1^2S23+a_1\ddot{\theta}_1C23+a_2S3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2C3(\ddot{\theta}_1+\ddot{\theta}_2)+a_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) \\ -g \end{bmatrix}$$

D3-4. Acceleration of the center of mass of link 3

$$a_3 = \{\dot{V}_3\} + [\text{Dev}\,\Delta_3]\begin{bmatrix} S_{3x} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \dot{V}_{3x} \\ \dot{V}_{3y} \\ \dot{V}_{3z} \end{bmatrix} + \begin{bmatrix} 0 & -(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) & 0 \\ \ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} S_{3x} \\ 0 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} S_{3x} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1\dot{\theta}_1^2C_{23}+a_1\ddot{\theta}_1S_{23}-a_2C_3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2S_3(\ddot{\theta}_1+\ddot{\theta}_2)-(a_3+S_{3x})(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 \\ a_1\dot{\theta}_1^2S_{23}+a_1\ddot{\theta}_1C_{23}+a_2S_3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2C_3(\ddot{\theta}_1+\ddot{\theta}_2)+(a_3+S_{3x})(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) \\ -g \end{bmatrix}$$

D3-5. Effective force

$$F_3 = m_3\{a_3\}$$

$$= \begin{bmatrix} m_3(-a_1\dot{\theta}_1^2C_{23}+a_1\ddot{\theta}_1S_{23}-a_2C_3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2S_3(\ddot{\theta}_1+\ddot{\theta}_2)-(a_3+S_{3x})(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2) \\ m_3(a_1\dot{\theta}_1^2S_{23}+a_1\ddot{\theta}_1C_{23}+a_2S_3(\dot{\theta}_1+\dot{\theta}_2)^2+a_2C_3(\ddot{\theta}_1+\ddot{\theta}_2)+(a_3+S_{3x})(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)) \\ -m_3g \end{bmatrix}$$

D3-6. Effective torque

$$N_3 = [I_3]\{\dot{\Omega}_3\} + [\text{Dev}\,\Omega_3]\,[I_3]\{\Omega_3\}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{3zz} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ \dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ I_{3zz}(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ I_{3zz}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) \end{bmatrix}$$

Starting Values for Backward Equation

Force Exerted on the Environment by link 3

$$f_g = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Torque Exerted on the Environment by link 3

$$n_g = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

D4. Backward Equations (Link 3)

D4-1. Total force exerted on link 3 by link 2 at joint 3

$$f_3 = [Rot4]\{f_0\} + \{F_3\} = \begin{bmatrix} F_{3x} \\ F_{3y} \\ F_{3z} \end{bmatrix}$$

D4-2. Vector to conter of mass

$$Q_3 = \{P_3\} + \{S_3\} = \begin{bmatrix} a_3+S_{3x} \\ 0 \\ 0 \end{bmatrix}$$

D4-3. Total torque exerted on link 3 by link 2 at joint 3

$$n_3 = [Rot4]\{n_g\} + [DevP_3][Rot4]\{f_g\} + [DevQ_3]\{F_3\} + \{N_3\}$$

$$= \begin{bmatrix} 0 \\ -(a_3+S_{3x})F_{3z} \\ (a_3+S_{3x})F_{3y}+I_{3z}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) \end{bmatrix}$$

D4-4. Torque applied by motor

$$\tau_3 = \{n_3\}^t[Rot3]^t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \{n_3\}^t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = (a_3+S_{3x})F_{3y}+I_{3zz}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)$$

D5. Backward Equations (Link 2)

D5-1. Total force exerted on link 2 by link 1 at joint 2

$$f_2 = [Rot3]\{f_3\} + \{F_2\}$$

$$= \begin{bmatrix} C3 & -S3 & 0 \\ S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} F_{3x} \\ F_{3y} \\ F_{3z} \end{bmatrix} + \begin{bmatrix} F_{2x} \\ F_{2y} \\ F_{2z} \end{bmatrix} = \begin{bmatrix} F_{3x}C3-F_{3y}S3+F_{2x} \\ F_{3x}S3+F_{3y}C3+F_{2y} \\ F_{3z}+F_{2z} \end{bmatrix}$$

D5-2. Vector to conter of mass

$$Q_2 = \{P_2\} + \{S_2\} = \begin{bmatrix} a_2-S_{2x} \\ 0 \\ 0 \end{bmatrix}$$

D5-3. Total torque exerted on link 2 by link 1 at joint 2

$$n_2 = [Rot3]\{n_3\} + [DevP_2][Rot3]\{f_3\} + [DevQ_2]\{F_2\} + \{N_2\}$$

$$= \begin{bmatrix} C3 & -S3 & 0 \\ S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ -(a_3+S_{3x})F_{3z} \\ (a_3+S_{3x})F_{3y}+I_{3zz}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) \end{bmatrix}$$

$$+\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a_2 \\ 0 & a_2 & 0 \end{bmatrix}\begin{bmatrix} F_{3x}C3-F_{3y}S3 \\ F_{3x}S3+F_{3y}C3 \\ F_{3z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -(a_2+S_{2x}) \\ 0 & a_2+S_{2x} & 0 \end{bmatrix}\begin{bmatrix} F_{2x} \\ F_{2y} \\ F_{2z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ I_{2zz}(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix}$$

$$= \begin{bmatrix} (a_3+S_{3x})F_{3z}S_3 \\ -(a_3+S_{3x})F_{3z}C_3-a_2F_{3z}-(a_2+S_{2x})F_{2z} \\ (a_3+S_{3x})F_{3y}+I_{3zz}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+a_2(F_{3x}S_3+F_{3y}C_3)+(a_2+S_{2x})F_{2y}+I_{2zz}(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix}$$

D5-4. Torque applied by motor

$$\tau_2 = \{n_2\}^t[Rot2]^t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \{n_2\}^t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= (a_3+S_{3x})F_{3y}+I_{3zz}(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+a_2(F_{3x}S_3+F_{3y}C_3)+(a_2+S_{2x})F_{2y}+I_{2zz}(\ddot{\theta}_1+\ddot{\theta}_2)$$

D6. Backward Equations (Link 1)

D6-1. Total force exerted on link 1 by link 0 at joint 1

$$f_1 = [Rot2]\{f_2\} + \{F_1\}$$

$$= \begin{bmatrix} C_2 & -S_2 & 0 \\ S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{2x} \\ f_{2y} \\ f_{2z} \end{bmatrix} + \begin{bmatrix} F_{1x} \\ F_{1y} \\ F_{1z} \end{bmatrix} = \begin{bmatrix} f_{2x}C_2-f_{2y}S_2+F_{1x} \\ f_{2x}S_2+f_{2y}C_2+F_{1y} \\ f_{2z}+F_{1z} \end{bmatrix}$$

D6-2. Vector to conter of mass

$$Q_1 = \{P_1\} + \{S_1\} = \begin{bmatrix} a_1-S_{1x} \\ 0 \\ 0 \end{bmatrix}$$

D6-3. Total torque exerted on link 1 by link 0 at joint 1

$$n_1 = [Rot2]\{n_2\} + [DevP_1][Rot2]\{f_2\} + [DevQ_1]\{F_1\} + \{N_1\}$$

$$= \begin{bmatrix} C2 & -S2 & 0 \\ S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} n_{2x} \\ n_{2y} \\ n_{2z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a_1 \\ 0 & a_1 & 0 \end{bmatrix}\begin{bmatrix} f_{2x}C2-f_{2y}S2 \\ f_{2x}S2+f_{2y}C2 \\ f_{2z} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -(a_1+S_{1x}) \\ 0 & a_1+S_{1x} & 0 \end{bmatrix}\begin{bmatrix} F_{1x} \\ F_{1y} \\ F_{1z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ I_{1zz}\ddot{\theta}_1 \end{bmatrix}$$

$$= \begin{bmatrix} n_{2x}C2-n_{2y}S2 \\ n_{2x}S2+n_{2x}C2-a_1f_{2z}-(a_1+S_{1x})F_{1z} \\ n_{2z}+a_1(f_{2x}S2+f_{2y}C2)+(a_1+S_{1x})F_{1y}+I_{1zz}\ddot{\theta}_1 \end{bmatrix}$$

D6-4. Torque applied by motor

$$\tau_1 = \{n_1\}^t[Rot1]^t\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \{n_1\}^t\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= n_{2z}+a_1(f_{2x}S2+f_{2y}C2)+(a_1+S_{1x})F_{1y}+I_{1zz}\ddot{\theta}_1$$

where

$m1 = 12.88$ kg , $I_{1zz} = 0.138841$ kgm^2
$m2 = 5.95$ kg , $I_{2zz} = 0.039523$ kgm^2
$m3 = 0.44$ kg , $I_{3zz} = 1.339*10^{-3}$ kgm^2
$a1 = 0.2207$ m , $S_{1x} = -0.0476$ m
$a2 = 0.1603$ m , $S_{2x} = -0.0619$ m
$a3 = 0.1508$ m , $S_{3x} = -0.0918$ m

APPENDIX E:

COMPUTER CODE OF NEURAL NETWORK CONTROLLER

```
/* N.N. Error Compensation Cartesian Controller for Unkown Environment */
/* Programmed by Kazuo Kiguchi */

/* declaration of input / output functions */

float ds3001(long base, long channel);
float ds2002(long base, long channel);
void ds2101(long base, long channel, float value);

#include <d:\c30tools\math.h>

/* declaration of gains for the controller */

float k1 =  4.0000000E+02;          /* gain Kpp */
float b1 = -0.8000000E+02;          /* gain Kpv */
float k2 =  0.2500000E+00;          /* gain Kfp */

/* declaration of variables */

float m1 = 1.2880000E+01;           /*  mass 1 (kg) */
float m2 = 5.9500000E+00;           /*  mass 2 (kg) */
float m3 = 0.4400000E+00;           /*  mass 3 (kg) */
float a1 = 0.2207000E+00;           /*  link 1 (m)  */
float a2 = 0.1603000E+00;           /*  link 2 (m)  */
float a3 = 0.1508000E+00;           /*  link 3 (m)  */
float I1zz = 0.1388410E+00;
float I2zz = 0.0395230E+00;
float I3zz = 1.3390000E-03;
float S1x = -0.0476000E+00;
float S2x = -0.0619000E+00;
float S3x = -0.0918000E+00;

float x1 = 0.4540000E+00;           /*  x direction displacement  */
float x2 = 0.0000000E+00;           /*  y direction displacement  */
float x3 = 0.0000000E+00;           /*  angle fai  */

float xc1 = 0.0000000E+00;          /*  x position control input  */
float xc2 = 0.0000000E+00;          /*  y position control input  */
float xc3 = 0.0000000E+00;          /*  fai position control input  */
float fc = 0.0000000E+00;           /*  force control input  */

float M[3][3];                      /* Inertia Matrix */
float OMG[3];                       /* Corillis etc. */
float J[3][3];                      /*  Jacobian Matrix  */
float IJ[3][3];                     /*  Inverse Jacobian Matrix  */
float JD[3][3];                     /*  derivative of Jacobian  */
float x[3];

float Mc = 0.0000000E+00;           /* equivalent mass coefficient */
float Bc = 0.0000000E+00;           /* equivalent damping coefficient */
float Kc = 2.0000000E+04;           /* equivalent spring coefficient */
float Mc_1,Bc_1,Kc_1;               /* previous coefficients */
float Kem = 0.;                     /* equivalent spring coefficient calculated from uff/x0 */
```

```
/* initial variables */

float x1d = 4.5400000E-01;          /* desired x position */
float x2d = 0.0000000E+00;          /* desired y position */
float x3d = 0.0000000E+00;          /* desired fai angle */
float u1 = -0.5377671E+00;          /* motor 1 measured angle */
float u2 =  1.3205011E+00;          /* motor 2 measured angle */
float u3 = -0.7827340E+00;          /* motor 3 */

float ufd = 2.0000000E+01;          /* desired force */
float uf = 0.0000000E+00;           /* measured force from force senser */

/* output variables */

float y1 =  0.0000000E+00;          /* output of the hybrid controller for motor1 */
float y2 =  0.0000000E+00;          /* output of the hybrid controller for motor2 */
float y3 =  0.0000000E+00;          /* output of the hybrid controller for motor3 */

/* other variables  */

float st = 1.0000E-03;              /* sampring time */

float to1 = 0.;                     /* torque to apply motor 1 for force control */
float to2 = 0.;                     /* torque to apply motor 2 for force control */
float to3 = 0.;                     /* torque to apply motor 3 for force control */
float uf_1 = 0.;
float uf_2 = 0.;
float uff = 0.;                     /* filtered measured force */
float uff_1 = 0.;
float uff_2 = 0.;
float x1_1 = 0.;
float x2_1 = 0.;
float x3_1 = 0.;
float x1v = 0.;                     /* x direction velocity */
float x2v = 0.;                     /* y direction velocity */
float x3v = 0.;                     /* angle velocity */
float x1v_1 = 0.;
float x2v_1 = 0.;
float x3v_1 = 0.;
float x1v_2 = 0.;
float x2v_2 = 0.;
float x3v_2 = 0.;
float x1vf = 0.;                    /* x direction filtered velocity */
float x2vf = 0.;                    /* y direction filtered velocity */
float x3vf = 0.;                    /* angle filtered velocity */
float x1vf_1 = 0.;
float x2vf_1 = 0.;
float x3vf_1 = 0.;
float x1vf_2 = 0.;
float x2vf_2 = 0.;
float x3vf_2 = 0.;
float x1a = 0.;                     /* x direction acceleration */
float x1a_1 = 0.;
float x1a_2 = 0.;
```

```
float x1af = 0.;                    /* x direction filtered acceleration */
float x1af_1 = 0.;
float x1af_2 = 0.;
float ufdt = 0.;                    /* instant desired force */
float u1_1 = 0.;
float u2_1 = 0.;
float u3_1 = 0.;
float u3_2 = 0.;
float u3f = 0.;
float u3f_1 = 0.;
float u3f_2 = 0.;
float u1v = 0.;                     /*  angular velocity of link 1  */
float u2v = 0.;                     /*  angular velocity of link 2  */
float u3v = 0.;                     /*  angular velocity of link 3  */
float u1v_1 = 0.;
float u2v_1 = 0.;
float u3v_1 = 0.;
float u3v_2 = 0.;
float u3vf = 0.;
float u3vf_1 = 0.;
float u3vf_2 = 0.;
float u1a = 0.;                     /* angular acceleration of link1 */
float u2a = 0.;                     /* angular acceleration of link2 */
float u3a = 0.;                     /* angular acceleration of link3 */
float u1a_1 = 0.;
float u1a_2 = 0.;
float u1af = 0.;
float u1af_1 = 0.;
float u1af_2 = 0.;
float u2a_1 = 0.;
float u2a_2 = 0.;
float u2af = 0.;
float u2af_1 = 0.;
float u2af_2 = 0.;

int N = 2;                          /* counter for plotting */
int N1 = 0;                         /* counter for position control */
int N3 = 0;                         /* counter for changing desired force */
int i,j;
int s = 0;                          /*  force/position mode  */

float s1,c1,s2,c2,s3,c3;            /*  temporary variables  */
float s12,c12,s23,c23,s123,c123;    /*  temporary variables  */
float ta,tb,tc,tuv;                 /*  temporary variables  */
float den;
float uf_i ;                        /*  initial force */
float x0;                           /*  initial position of force control */
float x1e;                          /*  pushed distance */
float co1 = 0.;                     /*  coefficient of friction comp.  */
float co2 = 0.;                     /*  coefficient of friction comp.  */
float sgn1;                         /*  sign for friction comp. Motor 1  */
float sgn2;                         /*  sign for friction comp. Motor 2  */
```

```
/* Initial Weights of Neural Network */

float w01 = -1.000E-03;
float w11 =  1.439E-03;
float w21 =  1.275E-03;
float w31 =  2.550E-03;
float w41 = -3.500E-03;
float w51 =  1.300E-03;
float w61 =  1.915E-03;
float w71 = -4.170E-03;
float w81 =  1.750E-03;
float w91 =  4.100E-03;
float w02 = -1.100E-03;
float w12 =  6.700E-03;
float w22 =  7.500E-03;
float w32 =  1.200E-03;
float w42 = -2.878E-03;
float w52 =  1.732E-03;
float w62 =  1.238E-03;
float w72 = -1.937E-03;
float w82 =  8.100E-03;
float w92 =  2.580E-03;
float w03 = -1.200E-03;
float w13 = -1.300E-03;
float w23 =  1.532E-03;
float w33 =  1.400E-03;
float w43 =  5.120E-03;
float w53 =  1.097E-03;
float w63 =  2.005E-03;
float w73 =  5.800E-03;
float w83 =  1.178E-03;
float w93 = -1.800E-03;
float w04 = -1.300E-03;
float w14 =  3.927E-03;
float w24 =  4.190E-03;
float w34 = -8.450E-03;
float w44 = -8.475E-03;
float w54 =  3.938E-03;
float w64 = -1.838E-03;
float w74 = -1.622E-03;
float w84 = -2.750E-03;
float w94 =  0.172E-03;
float w05 = -1.400E-03;
float w15 =  1.476E-03;
float w25 =  0.900E-03;
float w35 =  0.667E-03;
float w45 =  4.619E-03;
float w55 =  0.889E-03;
float w65 =  1.000E-03;
float w75 =  4.600E-03;
float w85 =  1.525E-03;
float w95 =  0.105E-03;
float w06 = -1.500E-03;
float w16 =  1.333E-03;
```

```
float w26 =  1.698E-03;
float w36 =  1.250E-03;
float w46 =  1.333E-03;
float w56 =  1.067E-03;
float w66 =  1.524E-03;
float w76 =  1.750E-03;
float w86 =  3.800E-03;
float w96 =  0.106E-03;
float w07 = -1.700E-03;
float w17 =  1.381E-03;
float w27 =  0.921E-03;
float w37 = -6.550E-03;
float w47 = -4.286E-03;
float w57 =  1.090E-03;
float w67 =  0.778E-03;
float w77 =  1.850E-03;
float w87 =  1.120E-03;
float w97 =  0.107E-03;
float w08 = -1.700E-03;
float w18 =  1.500E-03;
float w28 =  1.548E-03;
float w38 =  1.403E-03;
float w48 =  1.387E-03;
float w58 =  1.500E-03;
float w68 =  1.806E-03;
float w78 =  1.700E-03;
float w88 =  1.225E-03;
float w98 =  0.108E-03;
float w09 = -1.800E-03;
float w19 =  1.502E-03;
float w29 =  1.542E-03;
float w39 =  1.402E-03;
float w49 =  1.382E-03;
float w59 =  1.502E-03;
float w69 =  1.802E-03;
float w79 =  1.702E-03;
float w89 =  1.222E-03;
float w99 =  0.102E-03;
float w0x = -1.900E-03;
float w1x =  1.503E-03;
float w2x =  1.543E-03;
float w3x =  1.403E-03;
float w4x =  1.383E-03;
float w5x =  1.503E-03;
float w6x =  1.803E-03;
float w7x =  1.703E-03;
float w8x =  1.223E-03;
float w9x =  0.103E-03;
float w0x1 = -1.133E-03;
float w1x1 =  1.504E-03;
float w2x1 =  1.544E-03;
float w3x1 =  1.404E-03;
float w4x1 =  1.384E-03;
float w5x1 =  1.504E-03;
```

```
float w6x1 =  1.804E-03;
float w7x1 =  1.704E-03;
float w8x1 =  1.224E-03;
float w9x1 =  0.104E-03;
float w0x2 = -1.234E-03;
float w1x2 =  1.505E-03;
float w2x2 =  1.545E-03;
float w3x2 =  1.405E-03;
float w4x2 =  1.385E-03;
float w5x2 =  1.505E-03;
float w6x2 =  1.805E-03;
float w7x2 =  1.705E-03;
float w8x2 =  1.225E-03;
float w9x2 =  0.105E-03;

float v01 = -1.050E-03;
float v11 = -3.415E-03;
float v21 = -1.570E-03;
float v31 =  3.138E-03;
float v41 = -6.800E-03;
float v51 = -2.212E-03;
float v61 =  1.976E-03;
float v71 = -9.512E-03;
float v81 =  8.026E-03;
float v91 = -4.162E-03;
float vx1 =  3.532E-03;
float vx11 = -1.203E-03;
float vx21 = -1.650E-03;
float v02 = -1.150E-03;
float v12 =  1.075E-03;
float v22 = -6.950E-03;
float v32 =  7.500E-03;
float v42 = -4.200E-03;
float v52 = -8.750E-03;
float v62 = -1.039E-03;
float v72 =  5.650E-03;
float v82 = -2.756E-03;
float v92 = -5.352E-03;
float vx2 = -8.803E-03;
float vx12 = -6.244E-03;
float vx22 = -5.537E-03;
float v03 = -1.658E-03;
float v13 =  1.242E-03;
float v23 = -2.742E-03;
float v33 = -1.030E-03;
float v43 =  2.921E-03;
float v53 = -6.420E-03;
float v63 = -4.429E-03;
float v73 =  2.921E-03;
float v83 =  8.418E-03;
float v93 =  8.935E-03;
float vx3 =  9.695E-03;
float vx13 = -5.594E-03;
float vx23 =  8.905E-03;
```

```
float h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,t1,t2,t3;
float dif1,dif2,dif3;
float sh1,sh2,sh3,sh4,sh5,sh6,sh7,sh8,sh9,sh10,sh11,sh12,st1,st2,st3;
float sh1d,sh2d,sh3d,sh4d,sh5d,sh6d,sh7d,sh8d,sh9d,sh10d,sh11d,sh12d,st1d,st2d,st3d;
float st1_1,st2_1,st3_1,iv1,iv2,iv3,iv4,iv5,iv6,iv7,iv8,iv9,vden;
float er1,er2,er3,er4,er5,er6,er7,er8,er9,er10,er11,er12;
float for1,for2,for3;                    /* Torque commands for the motor */
float alf = -1.000E-04;                  /* Learning Rate */

c_int10()

{
 uf_2=uf_1;
 uf_1=uf;
 uff_2=uff_1;
 uff_1=uff;

 uf = -200*(ds2(0)2(0x0000000)20,0x0(0000001f)-uf_i);
 uff = 0.0913*uf + 0.1826*uf_1 + 0.0913*uf_2 + 0.9824*uff_1 - 0.3477*uff_2;          /* 120Hz */

 if (s == 0){
  if (N1 == 5000){
          x1d = 0.456;
          x2d = -0.0;
          x3d = 0.0;
          }
  if (N1 == 6000){
          x1d = x1d+0.001;
          x2d = 0.0;
          x3d = 0.0;
          N1 = 5000;
          }
 N1++;
 }

 x1_1=x1;
 x2_1=x2;
 x3_1=x3;
 x1v_2=x1v_1;
 x1v_1=x1v;
 x1vf_2=x1vf_1;
 x1vf_1=x1vf;
 x2v_2=x2v_1;
 x2v_1=x2v;
 x2vf_2=x2vf_1;
 x2vf_1=x2vf;
 x3v_2=x3v_1;
 x3v_1=x3v;
 x3vf_2=x3vf_1;
 x3vf_1=x3vf;
 uf_1=uf;
 u1_1=u1;
 u2_1=u2;
```

```
u3_2=u3_1;
u3_1=u3;
u3f_2=u3f_1;
u3f_1=u3f;
u1v_1=u1v;
u2v_1=u2v;
u3v_2=u3v_1;
u3v_1=u3v;
u1a_2=u1a_1;
u1a_1=u1a;
u1af_2=u1af_1;
u1af_1=u1af;
u2a_2=u2a_1;
u2a_1=u2a;
u2af_2=u2af_1;
u2af_1=u2af;

u1 = (-0.5377671E+00)-1*ds3001(0x00000040, 0x00000001)/.0116578;
u2 = (1.3205011E+00)-1*ds3001(0x00000040, 0x00000002)/.00776;
u3 = (-0.7827340E+00)-1*ds3001(0x00000050, 0x00000003)/3.794865E-05;
u3f = 0.0104*u3 + 0.0209*u3_1 + 0.0104*u3_2 + 1.691*u3f_1 - 0.7327*u3f_2;

s1 = sin(u1);
c1 = cos(u1);
s2 = sin(u2);
c2 = cos(u2);
s3 = sin(u3f);
c3 = cos(u3f);
s12 = sin(u1+u2);
c12 = cos(u1+u2);
s23 = sin(u2+u3f);
c23 = cos(u2+u3f);
s123 = sin(u1+u2+u3f);
c123 = cos(u1+u2+u3f);

x1 = a3*c123 + a2*c12 + a1*c1;
x2 = a3*s123 + a2*s12 + a1*s1;
x3 = u1+u2+u3f;

x1v = (x1-x1_1)/st;
x1vf = 0.0104*x1v + 0.0209*x1v_1 + 0.0104*x1v_2 + 1.691*x1vf_1 - 0.7327*x1vf_2;
x2v = (x2-x2_1)/st;
x2vf = 0.0104*x2v + 0.0209*x2v_1 + 0.0104*x2v_2 + 1.691*x2vf_1 - 0.7327*x2vf_2;
x3v = (x3-x3_1)/st;
x3vf = 0.0104*x3v + 0.0209*x3v_1 + 0.0104*x3v_2 + 1.691*x3vf_1 - 0.7327*x3vf_2;

x1a_2 = x1a_1;
x1a_1 = x1a;
x1af_2 = x1af_1;
x1af_1 = x1af;
x1a = (x1v-x1v_1)/st;
x1af = 0.0104*x1a + 0.0209*x1a_1 + 0.0104*x1a_2 + 1.691*x1af_1 - 0.7327*x1af_2;

u1v = (u1-u1_1)/st;
```

```
u2v = (u2-u2_1)/st;
u3v = (u3-u3_1)/st;
u3vf = 0.0104*u3v + 0.0209*u3v_1 + 0.0104*u3v_2 + 1.691*u3vf_1 - 0.7327*u3vf_2;
u1a = (u1v-u1v_1)/st;
u2a = (u2v-u2v_1)/st;
u3a = (u3vf-u3vf_1)/st;
u1af = 0.0104*u1a + 0.0209*u1a_1 + 0.0104*u1a_2 + 1.691*u1af_1 - 0.7327*u1af_2;
u2af = 0.0104*u2a + 0.0209*u2a_1 + 0.0104*u2a_2 + 1.691*u2af_1 - 0.7327*u2af_2;

if (uff > 0.80){
  if (s == 0) x0 = x1;
  s = 1;
  }

if (uff < 0.00 && s == 1){
  s = 0;
  N1 = 4000;
  }

if (s == 1){
  if(N3 == 2500) {
          ufd = 10.0;
          x2d = 0.0;
          }
  if(N3 == 5000) {
          ufd = 20.0;
          x2d = 0.0;
          N3 = 0;
          }
  ufdt = ufd; /* + 4*sin(3.141592654*st*N3); */

  if (N == 2)
        {
     x1e = x1-x0;
        if(x1e != 0) Kem = uff/x1e;

        asm("       trapu   27");           /* call TRACE30 */
        N = 0;
        }
  N++;
  N3++;
  }

if(u1v > 0) sgn1 = 1.;
else if(u1v < 0) sgn1 = -1.;
else sgn1 = 0.;

if(fabs(u1v) <= 0.002) co1 = fabs(u1v)/0.002;

if(u2v > 0) sgn2 = 1.;
else if(u2v < 0) sgn2 = -1.;
else sgn2 = 0.;

if(fabs(u2v) <= 0.002) co2 = fabs(u2v)/0.002;
```

```
J[0][0] = -a3*s123-a2*s12-a1*s1;
J[0][1] = -a3*s123-a2*s12;
J[0][2] = -a3*s123;
J[1][0] = a3*c123+a2*c12+a1*c1;
J[1][1] = a3*c123+a2*c12;
J[1][2] = a3*c123;
J[2][0] = 1.;
J[2][1] = 1.;
J[2][2] = 1.;

den = a1*a2*s2;
if (fabs(den) > 0.001) {
IJ[0][0] = a2*c12/den;
IJ[0][1] = a2*s12/den;
IJ[0][2] = a2*a3*s3/den;
IJ[1][0] = (-a2*c12-a1*c1)/den;
IJ[1][1] = (-a2*s12-a1*s1)/den;
IJ[1][2] = -a3*(a2*s3+a1*s23)/den;
IJ[2][0] = a1*c1/den;
IJ[2][1] = a1*s1/den;
IJ[2][2] = a1*(a3*s23+a2*s2)/den;
}

JD[0][0] = -a3*c123-a2*c12-a1*c1;
JD[0][1] = -a3*c123-a2*c12;
JD[0][2] = -a3*c123;
JD[1][0] = -a3*s123-a2*s12-a1*s1;
JD[1][1] = -a3*s123-a2*s12;
JD[1][2] = -a3*s123;

ta = a2*c3+a1*c23+a3+S3x;
tb = a2*s3+a1*s23;
tc = a2+a1*c2+S2x;
M[0][0] = I1zz+I2zz+I3zz+m3*ta*(a1*c23+a2*c3+a3+S3x)+m3*tb*(a1*s23+a2*s3)+m2*tc*tc+m2*a1*a1*s2*s2
+m1*(a1+S1x)*(a1*S1x);
M[0][1] = I2zz+I3zz+m3*ta*(a2*c3+a3+S3x)+m3*a2*s3*tb+m2*(a2+S2x)*tc;
M[0][2] = I3zz+m3*(a3+S3x)*ta;
M[1][0] = I2zz+I3zz+m3*(a1*c23+a2*c3+a3+S3x)*(a2*c3+a3+S3x)+m3*a2*c3*(a1*s23+a2*s3)+m2*(a2+S2x)
*(a1*c2+a2+S2x);
M[1][1] = I2zz+I3zz+m3*(a2*c3+a3+S3x)*(a2*c3+a3+S3x)+m3*a2*a2*s3*s3+m2*(a2+S2x)*(a2+S2x);
M[1][2] = I3zz+m3*(a3+S3x)*(a2*c3+a3+S3x);
M[2][0] = I3zz+m3*(a3+S3x)*(a1*c23+a2*c3+a3+S3x);
M[2][1] = I3zz+m3*(a3+S3x)*(a3+S3x)+m3*a2*c3;
M[2][2] = I3zz+m3*(a3+S3x)*(a3+S3x);

tuv = (u1v+u2v+u3vf)*(u1v+u2v+u3vf);
OMG[0] = (m3*a1*s23*ta-m3*a1*c23*tb+m2*a1*s2*tc-m2*a1*a1*a1*s2*s2*c2)*u1v*u1v + (m3*a2*s3*ta-
m3*a2*c3*tb-m2*a1*a1*s2*s2*(a2+S2x))*(u1v+u2v)*(u1v+u2v) - m3*tb*(a3+S3x)*tuv;
OMG[1] = (m3*(a2*c3+a3+S3x)*a1*s23-m3*a1*a2*s3*c23+m2*a1*s2*(a2+S2x))*u1v*u1v + (m3*a2*s3
*(a2*c3+a3+S3x)-m3*a2*a2*s3*c3)*(u1v+u2v)*(u1v+u2v) - m3*a2*s3*(a3+S3x)*tuv;
OMG[2] = m3*(a3+S3x)*(a1*s23*u1v*u1v + a2*s3*(u1v+u2v)*(u1v+u2v));

xc1 = k1*(x1d-x1)+b1*x1vf;          /*  control law for x  */
```

```
  xc2 = k1*(x2d-x2)+b1*x2vf;          /*  control law for y  */
  xc3 = k1*(x3d-x3)+b1*x3vf;          /*  control law for fai  */
  if(s == 1){
    fc = k2*(ufdt-uff);
  }

  x[0] = xc1-(JD[0][0]*u1v+JD[0][1]*u2v+JD[0][2]*u3vf);
  x[1] = xc2-(JD[1][0]*u1v+JD[1][1]*u2v+JD[1][2]*u3vf);
  x[2] = xc3-(JD[2][0]*u1v+JD[2][1]*u2v+JD[2][2]*u3vf);

  to1 = J[0][0]*(fc+ufdt);
  to2 = J[0][1]*(fc+ufdt);
  to3 = J[0][2]*(fc+ufdt);

  if (s == 0) {
  y1 = M[0][0]*(IJ[0][0]*x[0]+IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[0][1]*(IJ[1][0]*x[0]+IJ[1][1]*x[1]+IJ[1][2]*x[2])
+M[0][2]*(IJ[2][0]*x[0]+IJ[2][1]*x[1]+IJ[2][2]*x[2])+OMG[0]+sgn1*co1*0.9*9.8;

  y2 = M[1][0]*(IJ[0][0]*x[0]+IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[1][1]*(IJ[1][0]*x[0]+IJ[1][1]*x[1]+IJ[1][2]*x[2])
+M[1][2]*(IJ[2][0]*x[0]+IJ[2][1]*x[1]+IJ[2][2]*x[2])+OMG[1]+sgn2*co2*0.3;

  y3 = -1*(M[2][0]*(IJ[0][0]*x[0]+IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[2][1]*(IJ[1][0]*x[0]+IJ[1][1]*x[1]+IJ[1][2]*x[2])
+M[2][2]*(IJ[2][0]*x[0]+IJ[2][1]*x[1]+IJ[2][2]*x[2])+OMG[2]);
/*  y3 = -1*(M[2][0]*u1af+M[2][1]*u2af+M[2][2]*(IJ[2][0]*x[0]+IJ[2][1]*x[1]+IJ[2][2]*x[2])+OMG[2]);  */
  }

  if (s == 1) {
  y1 = M[0][0]*(IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[0][1]*(IJ[1][1]*x[1]+IJ[1][2]*x[2])+M[0][2]*(IJ[2][1]*x[1]+IJ[2][2]
*x[2])+OMG[0]+to1;
  y2 = M[1][0]*(IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[1][1]*(IJ[1][1]*x[1]+IJ[1][2]*x[2])+M[1][2]*(IJ[2][1]*x[1]+IJ[2][2]
*x[2])+OMG[1]+to2;
  y3 = -1*(M[2][0]*(IJ[0][1]*x[1]+IJ[0][2]*x[2])+M[2][1]*(IJ[1][1]*x[1]+IJ[1][2]*x[2])+M[2][2]*(IJ[2][1]*x[1]
+IJ[2][2]*x[2])+OMG[2]+to3);
  }


  /* Neural Networks */

  if (s == 1) {

  iv1 = x1;
  iv2 = x2+0.01;
  iv3 = x3+0.01;
  iv4 = x1_1;
  iv5 = x2_1+0.01;
  iv6 = x3_1+0.01;
  iv7 = y1;
  iv8 = y2;
  iv9 = y3;

  h1  = w01+w11*iv1+w21*iv2+w31*iv3+w41*iv4+w51*iv5+w61*iv6+w71*iv7+w81*iv8+w91*iv9;
  h2  = w02+w12*iv1+w22*iv2+w32*iv3+w42*iv4+w52*iv5+w62*iv6+w72*iv7+w82*iv8+w92*iv9;
  h3  = w03+w13*iv1+w23*iv2+w33*iv3+w43*iv4+w53*iv5+w63*iv6+w73*iv7+w83*iv8+w93*iv9;
  h4  = w04+w14*iv1+w24*iv2+w34*iv3+w44*iv4+w54*iv5+w64*iv6+w74*iv7+w84*iv8+w94*iv9;
```

```
  h5  = w05+w15*iv1+w25*iv2+w35*iv3+w45*iv4+w55*iv5+w65*iv6+w75*iv7+w85*iv8+w95*iv9;
  h6  = w06+w16*iv1+w26*iv2+w36*iv3+w46*iv4+w56*iv5+w66*iv6+w76*iv7+w86*iv8+w96*iv9;
  h7  = w07+w17*iv1+w27*iv2+w37*iv3+w47*iv4+w57*iv5+w67*iv6+w77*iv7+w87*iv8+w97*iv9;
  h8  = w08+w18*iv1+w28*iv2+w38*iv3+w48*iv4+w58*iv5+w68*iv6+w78*iv7+w88*iv8+w98*iv9;
  h9  = w09+w19*iv1+w29*iv2+w39*iv3+w49*iv4+w59*iv5+w69*iv6+w79*iv7+w89*iv8+w99*iv9;
  h10 = w0x+w1x*iv1+w2x*iv2+w3x*iv3+w4x*iv4+w5x*iv5+w6x*iv6+w7x*iv7+w8x*iv8+w9x*iv9;
  h11 = w0x1+w1x1*iv1+w2x1*iv2+w3x1*iv3+w4x1*iv4+w5x1*iv5+w6x1*iv6+w7x1*iv7+w8x1*iv8+w9x1
*iv9;
  h12 = w0x2+w1x2*iv1+w2x2*iv2+w3x2*iv3+w4x2*iv4+w5x2*iv5+w6x2*iv6+w7x2*iv7+w8x2*iv8+w9x2
*iv9;

  sh1 = 2/(1+exp(-h1))-1.;
  sh2 = 2/(1+exp(-h2))-1.;
  sh3 = 2/(1+exp(-h3))-1.;
  sh4 = 2/(1+exp(-h4))-1.;
  sh5 = 2/(1+exp(-h5))-1.;
  sh6 = 2/(1+exp(-h6))-1.;
  sh7 = 2/(1+exp(-h7))-1.;
  sh8 = 2/(1+exp(-h8))-1.;
  sh9 = 2/(1+exp(-h9))-1.;
  sh10 = 2/(1+exp(-h10))-1.;
  sh11 = 2/(1+exp(-h11))-1.;
  sh12 = 2/(1+exp(-h12))-1.;

  t1 = v01+v11*h1+v21*h2+v31*h3+v41*h4+v51*h5+v61*h6+v71*h7+v81*h8+v91*h9+vx1*h10+vx11*h11
+vx21*h12;
  t2 = v02+v12*h1+v22*h2+v32*h3+v42*h4+v52*h5+v62*h6+v72*h7+v82*h8+v92*h9+vx2*h10+vx12*h11
+vx22*h12;
  t3 = v03+v13*h1+v23*h2+v33*h3+v43*h4+v53*h5+v63*h6+v73*h7+v83*h8+v93*h9+vx3*h10+vx13*h11
+vx23*h12;

  st1 = 40/(1+exp(-t1))-20.0;
  st2 = 20/(1+exp(-t2))-10.0;
  st3 = 2/(1+exp(-t3))-1.0;

  for1 = y1+J[0][0]*st1+J[1][0]*st2+J[2][0]*st3;
  for2 = y2+J[0][1]*st1+J[1][1]*st2+J[2][1]*st3;
  for3 = y3+J[0][2]*st1+J[1][2]*st2+J[2][2]*st3;

  if (for1 > 16.0) for1=16.0;                    /* torque saturation for motor 1 */
  else if (for1 < -16.0) for1 = -16.0;

  if (for2 > 2.5) for2=2.5;                      /* torque saturation for motor 2 */
  else if (for2<-2.5) for2 = -2.5;

  if (for3 > 0.6) for3=0.6;                      /* torque saturation for motor 3 */
  else if (for3<-0.6) for3 = -0.6;

  ds2101(0x00000080, 0x00000001, for1/88.2);
  ds2101(0x00000080, 0x00000002, for2/9.8);
  ds2101(0x00000090, 0x00000003, for3/2.0);

  dif1 = ufdt-uff;
  dif2 = x2d-x2;
```

```
dif3 = x3d-x3;

st1d = 40*exp(-t1)/((1+exp(-t1))*(1+exp(-t1)));
st2d = 20*exp(-t2)/((1+exp(-t2))*(1+exp(-t2)));
st3d = 2*exp(-t3)/((1+exp(-t3))*(1+exp(-t3)));

sh1d = 2*exp(-h1)/((1+exp(-h1))*(1+exp(-h1)));
sh2d = 2*exp(-h2)/((1+exp(-h2))*(1+exp(-h2)));
sh3d = 2*exp(-h3)/((1+exp(-h3))*(1+exp(-h3)));
sh4d = 2*exp(-h4)/((1+exp(-h4))*(1+exp(-h4)));
sh5d = 2*exp(-h5)/((1+exp(-h5))*(1+exp(-h5)));
sh6d = 2*exp(-h6)/((1+exp(-h6))*(1+exp(-h6)));
sh7d = 2*exp(-h7)/((1+exp(-h7))*(1+exp(-h7)));
sh8d = 2*exp(-h8)/((1+exp(-h8))*(1+exp(-h8)));
sh9d = 2*exp(-h9)/((1+exp(-h9))*(1+exp(-h9)));
sh10d = 2*exp(-h10)/((1+exp(-h10))*(1+exp(-h10)));
sh11d = 2*exp(-h11)/((1+exp(-h11))*(1+exp(-h11)));
sh12d = 2*exp(-h12)/((1+exp(-h12))*(1+exp(-h12)));

er1 = dif1*st1d*v11+dif2*st2d*v12+dif3*st3d*v13;
er2 = dif1*st1d*v21+dif2*st2d*v22+dif3*st3d*v23;
er3 = dif1*st1d*v31+dif2*st2d*v32+dif3*st3d*v33;
er4 = dif1*st1d*v41+dif2*st2d*v42+dif3*st3d*v43;
er5 = dif1*st1d*v51+dif2*st2d*v52+dif3*st3d*v53;
er6 = dif1*st1d*v61+dif2*st2d*v62+dif3*st3d*v63;
er7 = dif1*st1d*v71+dif2*st2d*v72+dif3*st3d*v73;
er8 = dif1*st1d*v81+dif2*st2d*v82+dif3*st3d*v83;
er9 = dif1*st1d*v91+dif2*st2d*v92+dif3*st3d*v93;
er10 = dif1*st1d*vx1+dif2*st2d*vx2+dif3*st3d*vx3;
er11 = dif1*st1d*vx11+dif2*st2d*vx12+dif3*st3d*vx13;
er12 = dif1*st1d*vx21+dif2*st2d*vx22+dif3*st3d*vx23;

w01 = w01-alf*er1*sh1d;
w02 = w02-alf*er2*sh2d;
w03 = w03-alf*er3*sh3d;
w04 = w04-alf*er4*sh4d;
w05 = w05-alf*er5*sh5d;
w06 = w06-alf*er6*sh6d;
w07 = w07-alf*er7*sh7d;
w08 = w08-alf*er8*sh8d;
w09 = w09-alf*er9*sh9d;
w0x = w0x-alf*er10*sh10d;
w0x1 = w0x1-alf*er11*sh11d;
w0x2 = w0x2-alf*er12*sh12d;

w11 = w11-alf*er1*sh1d*iv1;
w12 = w12-alf*er2*sh2d*iv1;
w13 = w13-alf*er3*sh3d*iv1;
w14 = w14-alf*er4*sh4d*iv1;
w15 = w15-alf*er5*sh5d*iv1;
w16 = w16-alf*er6*sh6d*iv1;
w17 = w17-alf*er7*sh7d*iv1;
w18 = w18-alf*er8*sh8d*iv1;
w19 = w19-alf*er9*sh9d*iv1;
```

```
w1x = w1x-alf*er10*sh10d*iv1;
w1x1 = w1x1-alf*er11*sh11d*iv1;
w1x2 = w1x2-alf*er12*sh12d*iv1;

w21 = w21-alf*er1*sh1d*iv2;
w22 = w22-alf*er2*sh2d*iv2;
w23 = w23-alf*er3*sh3d*iv2;
w24 = w24-alf*er4*sh4d*iv2;
w25 = w25-alf*er5*sh5d*iv2;
w26 = w26-alf*er6*sh6d*iv2;
w27 = w27-alf*er7*sh7d*iv2;
w28 = w28-alf*er8*sh8d*iv2;
w29 = w29-alf*er9*sh9d*iv2;
w2x = w2x-alf*er10*sh10d*iv2;
w2x1 = w2x1-alf*er11*sh11d*iv2;
w2x2 = w2x2-alf*er12*sh12d*iv2;

w31 = w31-alf*er1*sh1d*iv3;
w32 = w32-alf*er2*sh2d*iv3;
w33 = w33-alf*er3*sh3d*iv3;
w34 = w34-alf*er4*sh4d*iv3;
w35 = w35-alf*er5*sh5d*iv3;
w36 = w36-alf*er6*sh6d*iv3;
w37 = w37-alf*er7*sh7d*iv3;
w38 = w38-alf*er8*sh8d*iv3;
w39 = w39-alf*er9*sh9d*iv3;
w3x = w3x-alf*er10*sh10d*iv3;
w3x1 = w3x1-alf*er11*sh11d*iv3;
w3x2 = w3x2-alf*er12*sh12d*iv3;

w41 = w41-alf*er1*sh1d*iv4;
w42 = w42-alf*er2*sh2d*iv4;
w43 = w43-alf*er3*sh3d*iv4;
w44 = w44-alf*er4*sh4d*iv4;
w45 = w45-alf*er5*sh5d*iv4;
w46 = w46-alf*er6*sh6d*iv4;
w47 = w47-alf*er7*sh7d*iv4;
w48 = w48-alf*er8*sh8d*iv4;
w49 = w49-alf*er9*sh9d*iv4;
w4x = w4x-alf*er10*sh10d*iv4;
w4x1 = w4x1-alf*er11*sh11d*iv4;
w4x2 = w4x2-alf*er12*sh12d*iv4;

w51 = w51-alf*er1*sh1d*iv5;
w52 = w52-alf*er2*sh2d*iv5;
w53 = w53-alf*er3*sh3d*iv5;
w54 = w54-alf*er4*sh4d*iv5;
w55 = w55-alf*er5*sh5d*iv5;
w56 = w56-alf*er6*sh6d*iv5;
w57 = w57-alf*er7*sh7d*iv5;
w58 = w58-alf*er8*sh8d*iv5;
w59 = w59-alf*er9*sh9d*iv5;
w5x = w5x-alf*er10*sh10d*iv5;
w5x1 = w5x1-alf*er11*sh11d*iv5;
```

```
w5x2 = w5x2-alf*cr12*sh12d*iv5;

w61 = w61-alf*cr1*sh1d*iv6;
w62 = w62-alf*cr2*sh2d*iv6;
w63 = w63-alf*cr3*sh3d*iv6;
w64 = w64-alf*cr4*sh4d*iv6;
w65 = w65-alf*cr5*sh5d*iv6;
w66 = w66-alf*cr6*sh6d*iv6;
w67 = w67-alf*cr7*sh7d*iv6;
w68 = w68-alf*cr8*sh8d*iv6;
w69 = w69-alf*cr9*sh9d*iv6;
w6x = w6x-alf*cr10*sh10d*iv6;
w6x1 = w6x1-alf*cr11*sh11d*iv6;
w6x2 = w6x2-alf*cr12*sh12d*iv6;

w71 = w71-alf*cr1*sh1d*iv7;
w72 = w72-alf*cr2*sh2d*iv7;
w73 = w73-alf*cr3*sh3d*iv7;
w74 = w74-alf*cr4*sh4d*iv7;
w75 = w75-alf*cr5*sh5d*iv7;
w76 = w76-alf*cr6*sh6d*iv7;
w77 = w77-alf*cr7*sh7d*iv7;
w78 = w78-alf*cr8*sh8d*iv7;
w79 = w79-alf*cr9*sh9d*iv7;
w7x = w7x-alf*cr10*sh10d*iv7;
w7x1 = w7x1-alf*cr11*sh11d*iv7;
w7x2 = w7x2-alf*cr12*sh12d*iv7;

w81 = w81-alf*cr1*sh1d*iv8;
w82 = w82-alf*cr2*sh2d*iv8;
w83 = w83-alf*cr3*sh3d*iv8;
w84 = w84-alf*cr4*sh4d*iv8;
w85 = w85-alf*cr5*sh5d*iv8;
w86 = w86-alf*cr6*sh6d*iv8;
w87 = w87-alf*cr7*sh7d*iv8;
w88 = w88-alf*cr8*sh8d*iv8;
w89 = w89-alf*cr9*sh9d*iv8;
w8x = w8x-alf*cr10*sh10d*iv8;
w8x1 = w8x1-alf*cr11*sh11d*iv8;
w8x2 = w8x2-alf*cr12*sh12d*iv8;

w91 = w91-alf*cr1*sh1d*iv9;
w92 = w92-alf*cr2*sh2d*iv9;
w93 = w93-alf*cr3*sh3d*iv9;
w94 = w94-alf*cr4*sh4d*iv9;
w95 = w95-alf*cr5*sh5d*iv9;
w96 = w96-alf*cr6*sh6d*iv9;
w97 = w97-alf*cr7*sh7d*iv9;
w98 = w98-alf*cr8*sh8d*iv9;
w99 = w99-alf*cr9*sh9d*iv9;
w9x = w9x-alf*cr10*sh10d*iv9;
w9x1 = w9x1-alf*cr11*sh11d*iv9;
w9x2 = w9x2-alf*cr12*sh12d*iv9;
```

```
v01 = v01-alf*dif1*st1d;
v11 = v11-alf*dif1*st1d*sh1;
v21 = v21-alf*dif1*st1d*sh2;
v31 = v31-alf*dif1*st1d*sh3;
v41 = v41-alf*dif1*st1d*sh4;
v51 = v51-alf*dif1*st1d*sh5;
v61 = v61-alf*dif1*st1d*sh6;
v71 = v71-alf*dif1*st1d*sh7;
v81 = v81-alf*dif1*st1d*sh8;
v91 = v91-alf*dif1*st1d*sh9;
vx1 = vx1-alf*dif1*st1d*sh10;
vx11 = vx11-alf*dif1*st1d*sh11;
vx21 = vx21-alf*dif1*st1d*sh12;

v02 = v02-alf*dif2*st2d;
v12 = v12-alf*dif2*st2d*sh1;
v22 = v22-alf*dif2*st2d*sh2;
v32 = v32-alf*dif2*st2d*sh3;
v42 = v42-alf*dif2*st2d*sh4;
v52 = v52-alf*dif2*st2d*sh5;
v62 = v62-alf*dif2*st2d*sh6;
v72 = v72-alf*dif2*st2d*sh7;
v82 = v82-alf*dif2*st2d*sh8;
v92 = v92-alf*dif2*st2d*sh9;
vx2 = vx2-alf*dif2*st2d*sh10;
vx12 = vx12-alf*dif2*st2d*sh11;
vx22 = vx22-alf*dif2*st2d*sh12;

v02 = v03-alf*dif3*st3d;
v13 = v13-alf*dif3*st3d*sh1;
v23 = v23-alf*dif3*st3d*sh2;
v33 = v33-alf*dif3*st3d*sh3;
v43 = v43-alf*dif3*st3d*sh4;
v53 = v53-alf*dif3*st3d*sh5;
v63 = v63-alf*dif3*st3d*sh6;
v73 = v73-alf*dif3*st3d*sh7;
v83 = v83-alf*dif3*st3d*sh8;
v93 = v93-alf*dif3*st3d*sh9;
vx3 = vx3-alf*dif3*st3d*sh10;
vx13 = vx13-alf*dif3*st3d*sh11;
vx23 = vx23-alf*dif3*st3d*sh12;
 }

}

void init();
void timer1(float time);

main()

{
 init();
 timer1(1.0000000E-03);
```

```
  for (i=0;i<=2;++i){
    OMG[i] = 0.0000000E+00;
    x[i] = 0.0000000E+00;
    for (j=0;j<=2;++j){
    J[i][j] = 1.0000000E+00;      /*  Jacobian Matrix  */
    IJ[i][j] = 0.0000000E+00;     /*  Inverse Jacobian Matrix  */
    JD[i][j] = 0.0000000E+00;     /*  derivative of Jacobian  */
    M[i][j] = 0.0000000E+00;
    }
  }

  ul_i = ds2002(0x00000020,0x00000011);
  for (;;)
    ;
}
```