# STUDY ON THE DYNAMIC AND FORCE CONTROL OF A ROBOTIC MANIPULATOR

By
**WANG QIAN**

A Thesis Submitted for the Degree of
**Doctor of Philosophy**
in the
**Faculty of Engineering**
UNIVERSITY OF LONDON
November, 1991


*Automatic Control Group*
**Department of Mechanical Engineering**
UNIVERSITY COLLEGE LONDON
Torrington Place
London WC1E 7JE
**ENGLAND**

*Dedicated to my mother*

# ACKNOWLEDGEMENT

I would like to thank everybody in the department, especially in the Automatic Control Group, who offered encouragement, discussion and help during the last three years of my PhD studies. My supervisor, Dr. David R. Broome, provided an ideal working environment, moral encouragement and even material support. Especially, his patience to read everything written by the author during the last three years to 'fix up the English' deserves a special mention. Thanks also to Matt Greenshields, Alistair Greig, Nassos Pittaras, Rowland Travis and Simon Lovell etc. who always offered help and discussion when I needed them. Furthermore, the jocund working company with Alan McBride, Keram Akasary and Martin Sykes etc. will remain in my memory for a long time. Prof. T. Lambert and Dr. N. Thornhill provided advice in this project, and are hereby acknowledged. And finally, my wife Xiaolu Wang, supported me during the hard times, contributed a great deal to the completion of this thesis.

Financial support was jointly provided by the British Council and the Chinese Government, who are greatly acknowledged as otherwise it would not have been possible to carry out this research.

# ABSTRACT

Self-tuning adaptive control for robotic manipulators is the main theme of this thesis and is used for dynamic control and force control of a robotic manipulator both in theoretical simulation and in experimental work.

A simplified dynamics model has been developed for a PUMA type robotic manipulator. Heavy symbolic calculation has been carried out to make full use of the special PUMA geometry so as to further reduce the mathematical burden in controlling the arm dynamics.

Extensive simulation has been carried out using the digital computer. A PASCAL program package with graphics display has been produced for robotic assembly (peg-into-hole) on a VAX workstation. Various dynamic control simulation programs have been written on an IBM PC using MODULA-2. A new self-tuning PID controller, whose gains have an explicit relation with process parameters, has been worked out. A new simulation scheme, which can make direct use of the Newton-Euler equations, has been developed for the robot control.

The self-tuning PID controller is used for the outer-loop force control of the PUMA560 industrial robotic manipulator. A three dimensional compliant device was designed to go between the robotic end-effector and the work environment. A PUMA560 supervisory control program package, incorporating real-time compliant motion control, written in MODULA-2 was developed on an IBM PC, with menu and multi-process support. Experiment has shown that adaptive compliant motion control can largely improve contact quality and tracking ability for robotic inspection in an unknown environment. The author has also succeeded in putting a peg into a hole with a maximum clearance 0.02 mm using the PUMA560.

# TABLE OF CONTENTS

# LIST OF TABLES

Chapter 2



Chapter 6

# TABLE OF FIGURES

Chapter 1



Chapter 2



Chapter 3

Chapter 4

Chapter 5



Chapter 6

Chapter 7

# NOTATION

| Term | Definition |
|---|---|
| $a$ | a direction in tool co-ordinates |
| $\vec{a}_i$ | acceleration of the *i+1*th joint |
| $\vec{a}_3$ | robot wrist acceleration |
| $\vec{b}_i$ | direction of the *i-1*th joint |
| $c_{ij}$ | element of compliance matrix |
| $\vec{d}_i$ | direction of *i*th link |
| $e$ | error between desired and real system response |
| $f$ | friction |
| $f_4$ | force applied to the arm from hand |
| $\vec{f}_{i-1,i}$ | force applied to *i*th link from *i-1*th link |
| $g$ | gravitation constant |
| $\vec{g}$ | vector gravitation form |
| $G_i$ | gravitational term in *i*th link |
| $h$ | parameter in dynamic equation of two DOF manipulator |
| $H_{ij}$ | element of inertia matrix of the dynamic equation |
| $i$ | number of joint |
| $[I_i]$ | *i*th link inertia matrix |
| $J$ | Jacobean matrix |
| $\vec{J}_i$ | Jacobean matrix in vector form |
| $k$ | stiffness of the compliant device, time series |
| $K$ | kinematic energy, stiffness of environment |
| $K_i$ | *i*th link kinematic energy |
| $L_i$ | *i*th link length |
| $m$ | mass |
| $m_i$ | mass of *i*th link |

| | |
|---|---|
| $M$ | mass of the environment |
| $n$ | total number of joints, n direction in tool co-ordinates |
| $N_{i,j}$ | torque to *j*th link from *i*th link |
| $N_4$ | torque applied to the arm from hand |
| $o$ | o direction in tool co-ordinates |
| $P_3$ | robot wrist position |
| $P_i$ | *i+1*th joint position |
| $q_i$ | joint angle |
| $\dot{q}_i$ | joint angular speed |
| $r$ | reference input |
| $\vec{r}_{i,i+1}$ | vector representation of link *i* |
| $\vec{r}_{i-1,3}$ | vector comprised of *i-1*th joint and wrist |
| $\vec{r}_{i,ci}$ | vector form present joint *i* to *i*th link centre |
| $\vec{t}_i$ | direction in the link co-ordinates |
| $T$ | sampling period |
| $\tau_i$ | torque of *i*th joint |
| $\theta_i$ | *i*th joint angular displacement |
| $u$ | control input |
| $U$ | potential energy |
| $v$ | viscous coefficient |
| $\vec{v}_i$ | velocity of the *i*th joint |
| $\vec{v}_3$ | velocity of robot wrist |
| $\vec{v}_{ci}$ | velocity of *i*th link mass centre |
| $V$ | velocity |
| $\vec{\omega}_i$ | angular velocity of the *i*th link |
| $\vec{\dot{\omega}}_i$ | angular acceleration of the *i*th link |
| $y$ | system output |

# CHAPTER 1
# INTRODUCTION

## 1.1 INTRODUCTION

As can see from the title of this thesis, the work is concerned with two aspects of robotics research. The first is dynamic control, other than simple constant PID control, for robotic manipulators. The other one is robotic force control making contact with the environment. The research is framed within *dynamics and control*, i.e. mainly in mechanical aspect of robotics research.

Dynamic control is directly aimed at fast and precise performance. The simple constant gain PID controller provides sufficient performance in many elementary industrial tasks. However, the performance of such controllers decreases rapidly when dynamic effects become significant, for example, the arm may become unstable when moving at high speed or under high load.

Robot control systems have traditionally been associated with the execution of elementary tasks specified in advance by a programming or planning system. Recent developments in planning and control clearly show a trend toward a tighter connection between perception and action. The goal is to develop real-time sensor-based control methodologies for a robot's operations in an evolving world in which there are both tolerance and uncertainties to deal with. Force control has emerged as a basic means of expanding a robot's

capabilities in performing tasks where contact has to be made. Almost all the industrial robot manipulators now commercially available are non-contact type, which greatly restricts their use. Pick and place, spot welding and similar tasks, can be accomplished using only purely position control.

The advent of powerful, cheap and compact computers has had as great an impact on control systems as on other fields of engineering. Modern controller systems, whether used in industrial process control, ship guidance or aircraft 'fly by wire' systems now almost exclusively incorporate microprocessors as the main computational element.

Compared with its analog counterpart, digital controllers offer two major advantages: One is that they decrease size and number of components. The other is its flexibility. Digital controllers win hands down over analogue controllers in this respect, as their control law and general operation may be altered by replacing the control programme, usually just a case of changing a instruction in a program. To alter the control law or operation of a dedicated analogue controller may require a complete redesign of the circuit, obviously something that cannot be carried out quickly and cheaply. Digital controllers with communication links may even be programmed remotely over the communication link, a process known as 'down line loading'. One new micro-controller is designed to plug directly into the phone system to be remotely programmed.

Digital control, especially self-tuning PID, is extensively analysed and simulated and used as a main control scheme in this thesis.

## 1.2 LITERATURE REVIEW

As a highly nonlinear, highly coupled multi-variable system, the robotic manipulator provides one of the most challenging and active fields of research within the control community. Many modern control approaches are applied to the control of the robotic manipulator. However, these modern control

schemes have either little practical use (e.g. lack of persistent excitation problem in the robotic adaptive control) or are too expensive as they have very complicated forms. The following sub-sections will have a review of the current research in two main topics of robotics.

### 1.2.1 Adaptive Control of Robotic Manipulators

Basically there are two kinds of control scheme, one is the so called performance-based control scheme. The other is called model-based control, where the dynamic model of the physical system to be controlled are explicitly appearing in the controller. Fig. 1.1 shows the classification of commonly used robotic control schemes. Although, there exist two control co-ordinates, joint space and Cartesian space, most of the existing control schemes are based on the joint space. Cartesian trajectory control (and force/compliant motion control) are achieved by closing the loop around the inner joint servo loops.

As can be seen in Fig. 1.1, the dynamic model can be used in two ways. It can be placed in the feed-forward loop to form the so called feed-forward compensation (or part of the dynamic model, usually gravitation, to form the so called partially compensation). And the model can also be placed in the feedback loop forming so called feedback linearization, i.e. so called *computed torque control.* Craig [1989] further develop the computed torque control method by using the Lyapunov criteria to identify the model parameters to cover the problem of model uncertainty. The resulting adaptive computed torque control was achieved because of his research. Craig's method is the most robust one but is also the most expensive controller because identification has to be processed together with calculation of robotic dynamics, which is complicated itself.

After the dynamic feed-forward compensation, a set of coupled linear perturbation equations are acquired. Liu [1987] used these coupled linear

equations together with the minimum variance self-tuning adaptive control strategies by Clark and Gawthrop [1975] to form his unique control scheme. But his scheme is complicated and has a central coupled form.

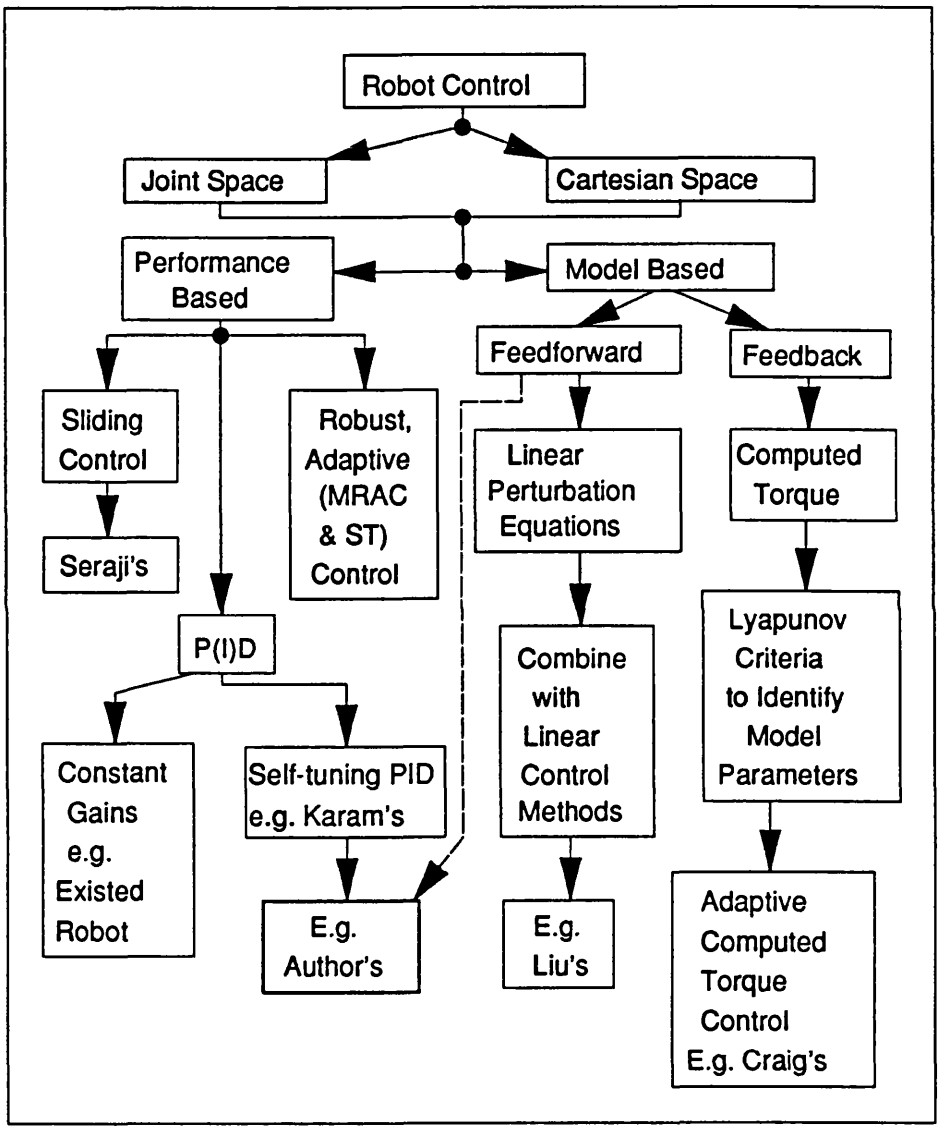



**Fig. 1.1 Control of robotic manipulators**

The author used Wellstead's pole assignment self-tuning adaptive control for designing the control scheme. The model feed-forward compensation is used to decrease the non-linearity and coupled effect. After feed-forward

compensation, the author assumes that the system can be treated as a set of de-coupled sub-systems. So each joint can be treated individually, just as in conventional industrial robotic manipulator control.

Performance-based methods are based on the assumption that an accurate dynamic model is not always available or is difficult to calculate, especially when various unknown payloads are to be manipulated by the arm. In the unknown payload situation, model-based methods have almost no effect. It is mainly for this reason that various performance-based control scheme have been developed during the last decade. Sliding-mode control, robust control and adaptive control are all good examples of these control schemes. Slotine and Li [1991], Asada [1986] and Seraji [1989] are all pioneers in the sliding mode control development for robotic manipulators. Various robust control controls like $H^{\infty}$ are also applied to the manipulator control.

The use of adaptive control for robot manipulators is not new. It is evident that adaptive control will definitely improve the performance of robot manipulators in terms of positional accuracy and response speed. However, industry still uses conventional simple PID controller for each joint. The main reason for this is because of the complexity of adaptive control, which affects reliability, and its high cost to implement.

Dubowsky and DesForges [1979] were the first to use Model Reference Adaptive Control (MRAC), where a continuous time single-input single-output (SISO) adaptive controller was designed for each joint of the manipulator. This method is easy to implement and has a small number of computations. However it ignored the coupling between joints and thus limited the speed of the manipulator. Another main problem is that stability was not guaranteed. Many improvements have been made in this field of research by using Lyapunov's theory and Popov's hyperstability theory.

Koivo and Guo [1983], first used minimum variance Self-tuning control developed by Clark and Gawthrop [1975] for the joint control of a robotic manipulator (Stanford Arm). However, Koivo and Guo's method is limited to systems that are open loop stable and the weighting factors P and Q in the cost function were restricted to be constants, and no rule was given for choosing them. Afterwards, there have being many papers which improve Model Reference Adaptive Control (MRAC) and Self-tuning Adaptive Control (STAC) for robotic manipulators.

Craig [1988], in his book, has a very good review of these papers, which, to summarise, have the following drawbacks:

1). It is still restricted to low speed configuration changing, ie slow speed manipulation.
2). Global stability is not guaranteed.
3). For slow time-varying parameters, Lyapunov stability theory can be used for MRAC and linear theory can be used for self-tuning adaptive control.
4). None of them deals with the problem of lack of persistent excitation.

Craig [1988] did not mention that some adaptive controllers have a centralized structure and some require extensive computation.

Refer to Fig. 1.1, it is worthwhile to single out the following people's work: Seraji [1989], Liu [1987], Craig [1988] and Karam [1989], because they stand for four different methods in the robot manipulator control.

Seraji's sliding mode control is the easiest so far as the author knows, where the controller is totally performance based and with no consideration of the manipulator dynamics. However, in Seraji's scheme, there remains the problem of how to choose the weighted error criterion. And what is more, in simulation the control inputs have shown high chattering phenomena.

Liu [1987] used N-E dynamics equation to make feed-forward compensation, then linearize the dynamic equation to derive a linearized perturbation equation. The parameters of the perturbation equation are identified, which

are then used for updating the controller's gains. His method implies that a rather accurate dynamic model is available and has the centralized form which is very complicated. Liu's method can be viewed as a combination of the model based and adaptive control.

Craig [1988] combine the computed torque and adaptive control for robotic manipulators. The L-E dynamics equations are used for calculating the feedback compensation with a position and velocity feedback loop. What's more, the parameters of the L-E equations are identified using the Lyapunov criterion to cover the problem of model uncertainty. His method is the most robust one with only the dynamic equation structure being needed, but it is also the most complicated.

The self-tuning method that has shown the most promise for manipulator control is an adaptation of Wellstead's [1979] pole-placement self-tuner by Leininger [1983]. Karam [1989] recently showed a very interesting adaptive controller which has the same form as Leininger's for robotic manipulators. Karam's controller is a simple PID structure and the second order dynamic model is assigned to each joint of the robotic manipulator. Because of such a simple choice the explicit relationship between the process parameters and the PID controller's gains can be acquired. His scheme is only an improvement of the traditional PID controller which is widely accepted in industry. The difference is that the dynamics of the robot manipulator are taken into account by resolving the process parameter into the designing of the PID controller using an adaptive strategy. In an early work (Wang [1991a]), this kind of controller has been simulated.

Chapter 3 and Chapter 4 are devoted to the dynamic control of non-linear systems. Simulation and control schemes description will be detailed there.

### 1.2.2 Adaptive Robotic Compliant Motion Control

Manipulation fundamentally requires the manipulator to be mechanically coupled to the object being manipulated, the manipulator may not be treated as an isolated system.

As shown in Fig. 1.2, two main groups can be distinguished: one group (first group) takes the compliant motion as a system, and the dynamics of contact include both manipulator itself and environment. The other group (second group) applies a low stiffness compliant end-effector mounted at the robotic manipulator's end-effector, so as to ignore the manipulator's dynamics.

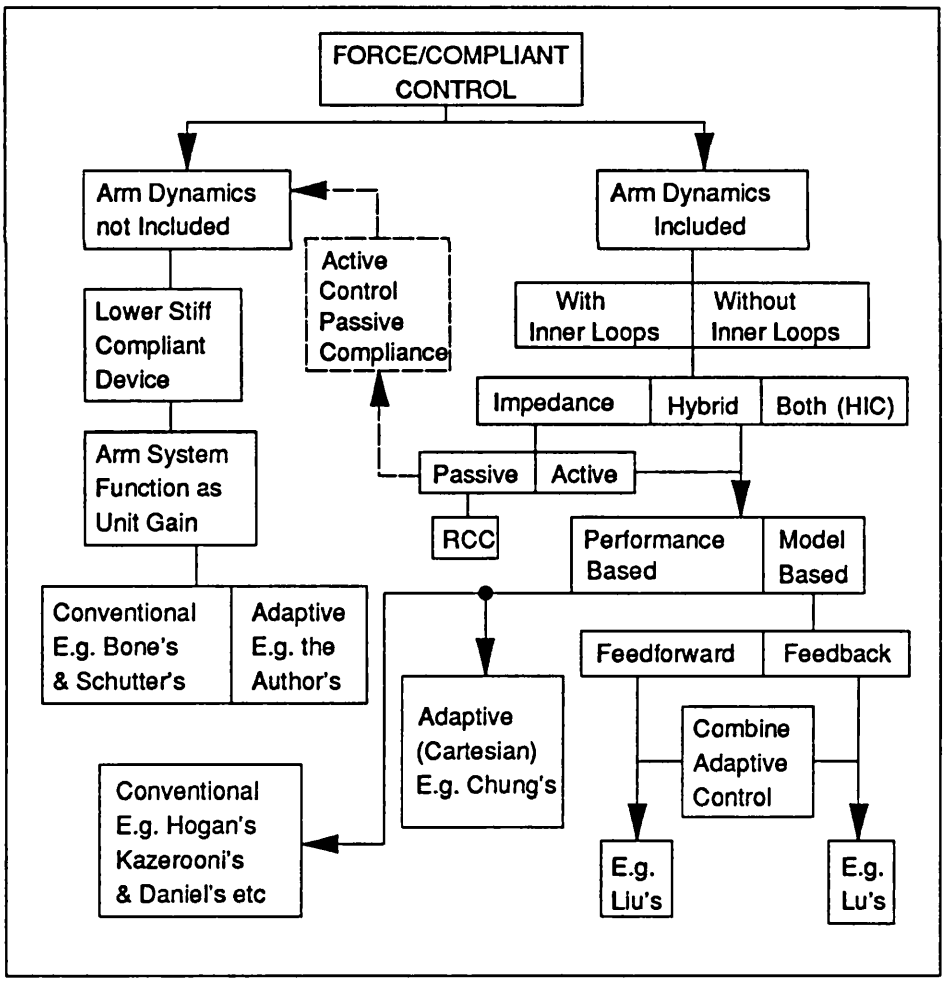



Fig. 1.2 Force control category

In Chapter 5, more detail will be given to the second group. The first group can be further divided according to whether there is an inner high-bandwidth loop (Daniel [1989]), or not, as An [1989], Liu [1988], Epping [1986], Seering [1987] and Youcef-Toumi [1987]. With an inner loop, it is assumed that the force control loop is closed around some higher-bandwidth inner loop, such as a position controller (Kazerooni [1989], Stepien [1987], Whitney [1976], and Paul [1987]). A review of the different control architectures in which force feedback appears is presented by Whitney [1987].

Master and slave manipulators can be dated up to 1940s. Later in 1950s, force-feedback was applied (Whitney [1985]). Time delay of as little as 250 milliseconds will cause these systems to be unstable. At the turning of 1970s, research turned to replacing the human operator with a computer. All the approaches depended on people to formulate the details, and those systems that were tested experimentally encountered the stability problem. The problem has been mitigated by the use of more sophisticated control algorithms, faster computation, and flexible sensors, but as yet no automatic generation of strategies has been achieved.

Force feedback, from its very beginning, is in the operational space, which is especially true when a wrist force sensor is going to be assembled at the manipulator end-effector. With this feature, force control is totally different from joint servo control. When the manipulator dynamics are taken into account, force control is coupled with the joint of the manipulator being used with each joint contributing to the overall contact forces/torques. So co-ordinate transformations between operational space and joint space are required within each sampling period.

The sampling rate will be decreased and so the robot's operational performance and disturbance-rejection ability are reduced. To-date the only experimental work using Cartesian force control in the operational space can be found in An's work [1989]. The manipulator he used is 2 DOF planar with

revolute joints, with no speed reduction gear, ie. the link is driven directly from the actuators. His Cartesian force control scheme is based on the dynamic system of the robotic manipulator making contact with the environment. The whole control scheme is open-loop with no force feedback. The contact force is maintained on the assumption that the joint torque can be controlled precisely, which is made possible on the Direct Drive Arm developed at MIT.

An [1989] succeeded in controlling the contact of his 2 DOF robotic manipulator with the environment. Attention has to be drawn to some parts: 1). The simplicity of the manipulator configuration he chose results in a simple transformation between co-ordinates. 2). There is no wrist-force feedback, so the stability problem becomes less important.

It is for these reasons that many force control schemes in the literature still use the inner higher bandwidth loop. The task space operational path and force control is accomplished by an outer-loop force/position control loop which are closed around the inner-loop. However, in these schemes, under existing computer power, force feedback can not be calculated quickly enough. Taking the PUMA560 industrial robotic manipulator as an example, force feedback can only be computed at the sampling rate of 36 Hz, ie a sampling period of 28 ms. This time period is too long for the real-time force control implementation. The manipulator bounces against the environment uncontrollably. A compliant end-effector was assembled between the robot end-effector and its environment to solve the stability problem.

The effect of using a lower stiffness compliant device is to greatly reduce the whole system stiffness, thus reducing operational speed and cause other side-effects on system performances as will be discussed in Chapter 6. It is highly desirable that the system can be operated in a stable way without a compliant device. Purely Cartesian force control scheme in the operational space without an inner-loop needs a very powerful computer, so as to increase sampling rate.

Daniel's [1989] work suggested that inner-loop force feedback should be added other than position feedback alone. His group has done a lot of work in force control with force (torque) feedback within each joint servo. In their research, they found that velocity feedback measured by a tachogenerator, rather than calculated from the position differences (encoder based, common practice of industrial robots), is crucial for the whole system stability in the extremely low speed manipulation when a robotic manipulator has made contact with its environment.

However, for the PUMA560 robotic manipulator, there is an anti-backlash mechanism which introduces compliances between joint mounted strain gauge (for measuring force/torque) and the motor. This results in a slip/stick phenomena within the drive, not observable from the motion of the robot, which generates high frequency signals from the strain gauges. As soon as the motor moves, stick/slip injects high frequencies into the anti-backlash mechanism which then resonates. The internal resonance is uncontrollable due to the slip/stick non-linearity between the motor and the mechanism and can send a strain gauge based inner-loop controller unstable. This phenomena restricts the bandwidth of the inner-loop controller to below the resonant frequency and thus limits the amount of sustainable outer loop feedback.

Other schemes to improve the inner-loop exist, and are not discussed here in great detail. It is remarked here that no differences are given to Cartesian and operational space (task frame) compliant (force) control schemes, as the main purpose is to define the differences of these control schemes from the common joint servo control systems. The essence of Cartesian and operational space hybrid control schemes is that the co-ordinate transformations are now within the control servo-loop.

Basically robot force control strategies are all the same as those used in robot position control. According to the various methods for force control being interfaced to the arm position servo, different force control methods are

named: Stiffness control; Damping control; Hybrid force-position control; Impedance control, etc. This is still confusing as there are many control schemes for the manipulator itself, like P(I)D, adaptive and sliding mode etc. (refer to the former section). For this reason, it is suggested that when a force control method is referred to, it should also refer to the arm position control method as well, and so suggested names can be Stiffness-P(I)D, Damping-Adaptive and Hybrid-P(I)D force control etc.

According to the way force-feedback is incorporated into the manipulator's position servo, two main force control approaches can be distinguished: (1) Hybrid, and (2) Impedance control. In the later method, the force signal is transferred as position or velocity commands, which are then input to the manipulator position servo and the manipulator remains as a position controlled device.

Impedance Force Control

Impedance force control is a general approach to robot motion and force control that attempts to make a manipulator behave as a mass-spring-dashpot system (Hogan [1985]), whose parameters (inertia-stiffness-damping) can be specified arbitrarily. A simple example would be to give a robot arm spring-like characteristics. A large spring constant results in a stiff arm. Position and force control are considered two forms of impedance control. Position control implies very high impedances, while force control implies the opposite.

Impedance control has not been implemented in its full form, but rather in some simple forms like stiffness or damping control. Impedance control does not specify desired forces or positions, but rather desired dynamic relationship between force and position, in other words mechanical stiffness or "impedances", with fairly slow motions.

Remote Central Compliance (RCC) is a typical passive compliance, capable of quick response, but its applications are necessarily limited to very specified tasks - RCC, for instance, can only achieve peg-into-hole problem.

A programable active control, by contrast, would allow the manipulation of various types of compliant motions by controlling the whole arm impedances. In the other words, we can get the desired end-point stiffness by appropriate controller design of each joint-servo to acquire the desired joint stiffness. In this sense, the manipulator can be seen by the environment as a spring and damping.

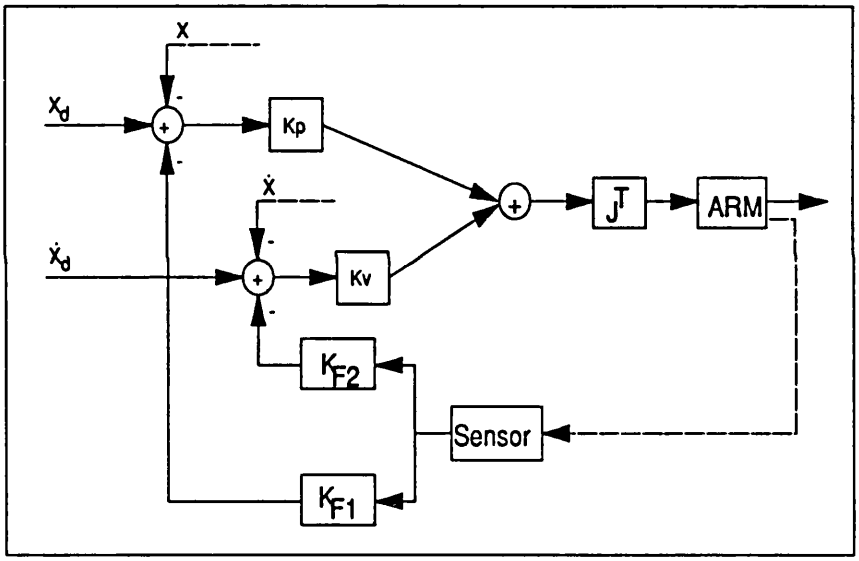



**Fig. 1.3 Impedance force control scheme**

As shown in Fig. 1.3, force is transferred to position and velocity without desired force appearing in the structure. Force control is accomplished by controlling the position. If *Kv* is chosen to be zero, then it has the structure of stiffness control, likewise, if *Kp* equals to zero, the controller is simplified to damping controller.

It is difficult to realize sufficient compliance for good impedance matching with an environment, because the stable region of desired compliance is

limited by the stiffness of the environment. For example, when a manipulator is contouring a stiff object, the environmental stiffness varies between extremely stiff and extremely soft according to whether contact is made or not. In order to reduce the stiffness and to avoid oscillation of motion, heavy damping must be applied, and this damping determines the speed at which the manipulator follows the object when the contact is broken. It has also been realized that stability can be maintained if some compliance, which may be derived from a force sensor or an end-effector, is inserted between the manipulator and an object, since this reduces the environmental stiffness for the joint servo. However, it also reduces the advantage of the stiff servo rigidity of the manipulator and makes it impossible to achieve a combination of rigid motion and soft motion along different axes.

Impedance control has been implemented in many forms. In its simplest form it can be considered as a generalization of damping and stiffness control schemes. In this form, it is essentially a PD position controller, with position and velocity feedback gains adjusted to obtain different impedances.

Hybrid Force Control

Raibert and Craig [1981] first suggested this method for force control, in which force and position are both added to the joint servo of robotic manipulators individually. Forces are not transferred as position commands, but applied directly to the joint servo. That is to say, even if there is no motion, the joint servo may be asked to provide a torque to maintain a pre-specified end-effector contact force.

In Raibert and Craig's [1981] original hybrid force control controller, only proportional gain is present. To improve control property, velocity and integration feedback are added as in Fig. 1.4. Inverse Jacobean matrix is presented in the control structure, which, according to An [1989], is the main reason for the hybrid controller's instability.

The theory backed hybrid force control is that the force-controlled directions and the position controlled direction in the operational space are complementary and orthogonal with each other. So the force-controlled directions can be grouped as a sub-space, as can the position-controlled directions. Both of these sub-spaces contribute to the joint servo, and their demands are transferred to joint space by calculating the inverse Jacobean transformation matrix. The position servo is supposed to be infinitely stiff and rejects all force disturbances acting on the system. Likewise, an ideal force servo exhibits zero stiffness and maintains the desired force application regardless of position disturbances.

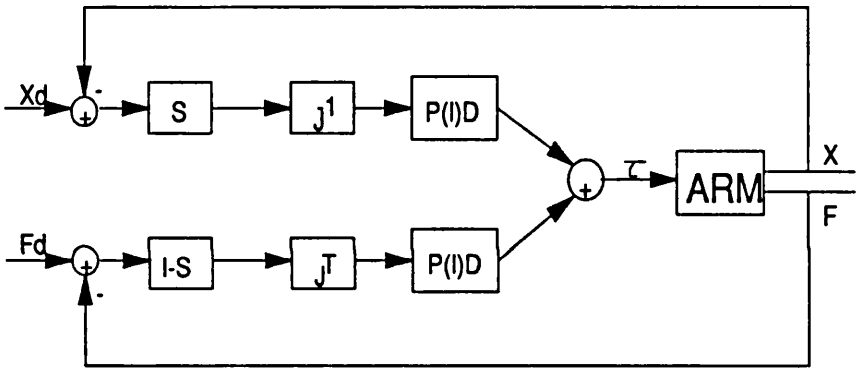



**Fig. 1.4 Hybrid controller**

As described in the former section, it may be useful to be able to control the end-effector to exhibit stiffness other than zero or infinite. In general, we may wish to control the mechanical impedance of the end-effector. In hybrid control schemes, it is assumed that the environment the robotic manipulator is to make contact with is very stiff. In many applications this assumption is not true, so hybrid impedance control is used.

Hybrid Impedance Control Scheme

Hybrid control is a highly intuitive approach to force control, which properly recognizes the distinction between force-controlled and position-controlled sub-spaces. The problem with hybrid control is its failure to recognize the

importance of manipulator impedance. Impedance control considers the effects of manipulator impedance on robot/environment interactions. When performed in task space, a known impedance can be maintained for all configurations. It is considered, however, to be solely a position-controlled scheme, with small adjustments made to react to contact forces. Positions are commanded, and impedances are adjusted to obtain the proper force response. No attempt is made to follow a commanded force trajectory and any distinction between force-controlled sub-spaces and position-controlled sub-spaces is ignored.

However, in hybrid force control, system stability is a problem. It is also worthwhile to point out that hybrid force control is only suitable for a stiff environment (as will be seen in the simulation present in Chapter 5). If the environment is flexible with noticeable displacement when force is applied, then force-controlled and position-controlled direction are difficult to distinguish and are not orthogonal to each other.

Several researchers have tried to combine both impedance and hybrid control schemes to form the so called Hybrid Impedance Control (HIC). The basic idea of HIC is that in the force controlled directions, position feedback is introduced, and in the position controlled directions force feedback is introduced. Fig. 1.5 shows one of the HIC control schemes:

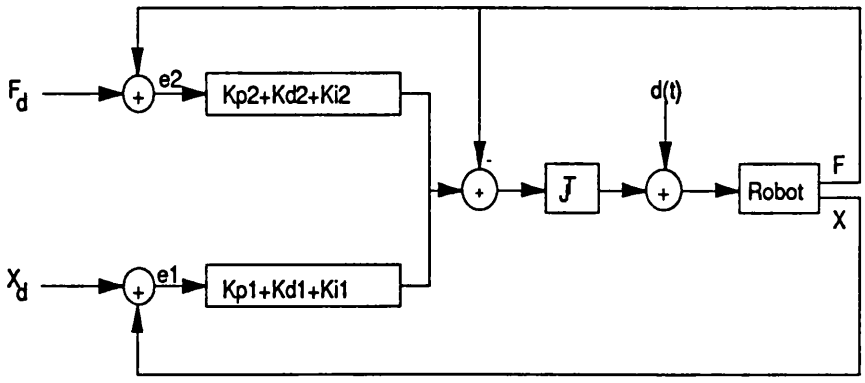



**Fig. 1.5 Hybrid impedance control scheme**

This is the general form of the Hybrid Impedance Control (HIC) scheme. If position gains in the force-controlled direction are zero, and force feedback gains in the position-controlled directions are zero, then the Hybrid control scheme is obtained (different from the original one). If force control servo does not exist, then a kind of impedance control results. As shown in Fig. 1.5, inverse Jacobean matrix does not appear in the control loops (transpose Jacobean matrix does).

Adaptive Cartesian Compliant Control

As stated before, joint space force/torque -feedback is undesirable because of the noise and slip/stick phenomena (for PUMA560, probably not in DDArm). Hybrid force/position control shows coupling between these two loops, which affects the arm performance. Most of the hybrid or hybrid impedance control scheme supposes the control is to be accomplished in the robot operational space, and there is no inner higher bandwidth position/velocity feedback loop.

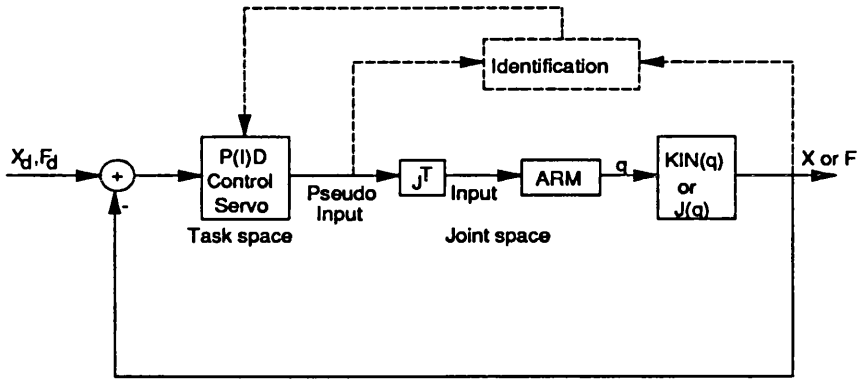



**Fig. 1.6 Adaptive force control scheme**

Adaptive control strategies can also be used in designing control schemes in robotic operation spaces. Liu etc. [1988] proposed a adaptive task space force control scheme, which has the same structure as hybrid impedance. Partial feed-forward using the dynamic model is also discussed in his control scheme. The gains are to be chosen by minimising a general object function. Chung

and Leininger [1990] used Wellstead's pole placement self-tuning adaptive control strategies for task level force control. His controller is of hybrid structure and there is no feed-forward compensation in the control scheme. Both schemes identify the dynamic relationship between the Pseudo-input and the output task-level forces/positions, which are measured at the wrist.

Chung and Leininger [1990] had also implemented their control scheme on a PUMA600 robotic manipulator. The task specified is to move a load of 2.27 kg around a vertical square with round corners. The actual wrist position/orientation of the end-effector with respect to task co-ordinate, which is fixed in Cartesian co-ordinates in this case, was calculated in real-time from the joint encoder. The sampling time required was determined to be 28 ms. Experimental results show further necessary refinement for real-time industrial applications, mainly due to the large sampling period (compared with 0.875 ms in the joint servo feedback). Lu [1991] injected Craig's [1986] model based adaptive control scheme to Hogan [1985]'s impedance control to form his force control scheme.

## 1.3 OBJECTIVE OF THIS RESEARCH

The objective of the research is to obtain a better understanding of the dynamic and force control properties of a robot manipulator. One end goal is to automate a sub-sea inspection task, which is one of the overall aims of the Automatic Control Group, University College London. The present task to move an inspection probe along the inspecting surface is done by human-beings. Robotic compliant motion control is necessary for a manipulator to carry out this task automatically.

With over ten year's experience in automatic inspection, the Automatic Control Group has gained a wide range of expertise in this field, especially in sub-sea or such ill-organised environments. Chapter 7 will have a detailed description on this subject. Usually there are four stages in sub-sea inspection operations.

The actual inspection operation needs the probe to make contact with the working object. This is a very good application for robotic compliant motion control to automatic this process. The work present in this thesis contributes to the last stage of sub-sea inspection.

Another objective of this research is to investigate ways of making the traditional industrial robot manipulators do as many tasks as possible.

## 1.4 SUMMARY OF MAIN CONTRIBUTIONS

As the result of this study, a few contributions are summarised as follows:

1). A new simulation scheme for the control of robotic manipulators was developed for the first time. This scheme can make direct use of Newton-Euler dynamic equations, which have traditionally been thought of as inconvenient for simulation of robotic manipulators.

2). An approximate recursive least square identification was assessed and judged. Its use for robotic manipulator was simulated extensively. This simplified method can greatly reduce the mathematic calculations. Simulation shows that the control effect can be much improved as the chattering control input phenomena will disappear by letting the forgetting factor be slightly great than one.

3). A new self-tuning adaptive PID controller, with new explicit relationship with process parameters, was worked out. Tustin's bi-linear transform was used to acquire the digital PID controller from its analogue counterpart, with adding an extra parameter. A general discrete transfer function for the second order system was used to represent the process.

4). Self-tuning PID for robotic compliant motion control was implemented on PUMA560 in VAL-2. Real-time experiment shows that adaptive compliant motion control can largely improve robotic contact quality and provide much better tracking ability in an unknown environment.

5). Robotic assembly was attempted using the PUMA560 industrial robotic manipulator. It has been succeeded in putting a peg into a hole with maximum clearance 0.02 mm at a reduced speed.

## 1.5 ORGANISATION OF THE THESIS

The whole PhD thesis consists of eight chapters. Chapter 1, as usual, is devoted to introduction, which has a full review of what people have done in this field of research, discussion on objective of this research and main contribution as a result of this research.

Chapter 2 contributes to the kinematics and dynamics of a robotic manipulator — PUMA560. A new solution in the Automatic Control Group, UCL, to the kinematic of PUMA manipulator is described. This solution makes uses of both analysis and geometry together, instead of heavy matrix manipulation, to solve the kinematics problem. A new dynamics model for PUMA560 was worked out and described in Chapter 2.

Chapter 3 deals with self-tuning adaptive control theory. An approximate RLS identification method was introduced and assessed. A new self-tuning PID digital controller based on $\hat{a}$ general discrete transfer function was developed.

Chapter 4 is mainly about the practical application of what described in Chapter 3 for robotic manipulators. It is found that by sightly increasing the forgetting factor to be larger than 1 it is possible to get rid of the control input chattering effect.

Chapter 5 is about the compliant motion control theory. Coupled concept for surface contact was introduced for the first time. Methods to improve contact quality and tracking ability has been investigated.

Chapter 6 is devoted to the application of adaptive compliant motion control for robotic assembly, ie peg-into-hole problem. A compliant device and its feature are described and analysed. A control strategy for assembly has been discussed and the practical experiment was described.

Chapter 7 is for the application of adaptive compliant motion control for robotic sub-sea inspection. It is concluded that adaptive compliant motion can largely improve contact quality and tracking ability.

Chapter 8 is for summary and suggestion for further research.

# CHAPTER 2
# DYNAMICS OF A ROBOTIC MANIPULATOR

## 2.1 INTRODUCTION

Dynamics of a robotic manipulator are introduced in the next few paragraphs to give a clear review of what people have done in this field of research and what has been done in this project.

From the point of view of dynamics, the robotic manipulator is a highly nonlinear, highly coupled and parameter changing multi-variable system. Because of this, the control of robot manipulators is difficult. Nowadays dynamic control methods either ignore or make only limited compensation for the variation of inertia and payload, or coupling between joints, which thus lead to the degradation of response velocity and accuracy. However, nonlinear control methods, such as, torque calculating and nonlinear feed-forward, usually need a more accurate dynamic model and at the same time, a more complicated control structure, thus incurring higher costs in computational burden when put to practical use.

Basically there are two ways to formulate the dynamic equations of a robotic manipulator, one being the Newton-Euler (N-E) method; the other is the Lagrange-Euler (L-E) method. Because of Luh's work [1980], the computation complexity of the N-E method has been reduced dramatically, from $O(n^4)$ to

O(n) (n is the number of joints). A comparison has been made in Table 2.1 (cf. Fu [1987] pp132). This is accomplished by setting up the calculation in recursive from and by expressing the velocities and accelerations of the links in the local link frames of reference. Nevertheless, N-E equations have their own inherent drawbacks in that they are difficult to use in the designing of the control system and in simulation.

The L-E equations, on the other hand, provide explicit state equations for robot manipulator dynamics and can be utilized to analyse and design advanced joint-variable space control strategies. The only defect of L-E equations is their computational inefficiency which arises partly from the 4x4 homogeneous matrices and the inverse Jacobean matrix. Hollerbach [1980] exploited the recursive nature of the L-E formulation. However, these recursive equations destroy the "structure" of the dynamic model which is quite useful in providing insight for designing the controller in state space (Fu [1987] pp84). For state space control analysis, one would like to obtain an explicit set of closed form differential equations (state equations) that describe the dynamic behaviour of a manipulator. In addition, the inertia and coupling forces in the equations should be easily identified so that an appropriate controller can be designed to compensate for their effects.

**Table 2.1 Comparison of dynamics computational complexity**

| Approach | L-E Method | N-E Method |
|---|---|---|
| Multiplications | $128/3\ n^4 + 512/3\ n^3 + 739/3\ n^2 + 160/3\ n$ | 132 n |
| Additions | $98/3\ n^4 + 781/6\ n^3 + 559/3\ n^2 + 245/6\ n$ | 111n - 4 |

In this thesis the author used Horak's [1984] scheme of partitioning the inverse dynamics computation into two parts, an arm and an end-effector (hand), and uses closed form solutions in symbolic form. Horak's idea is only the extension of what people have already done in the kinematics analysis of

robot manipulators. The last three joints of most industrial robot manipulators are coincident at a point, the robot manipulator's kinematics can be analysed separately (Fu [1987]). Horak noticed that the same thing existed in the dynamics analysis of the robotic manipulators, by separating the robot manipulator into two parts. The first part computes the inverse dynamics of the first three links using the L-E equations expressed in symbolic form and the second part computes the inverse dynamics of the hand (i.e. the last three links) using the N-E equations, also in symbolic form.

The symbolic methods fully exploit the particular kinematic and dynamic structure of a given manipulator and eliminate unnecessary arithmetic operations, either inherent in the formulation of kinematic and dynamic equations (e.g., the sparsity of matrices and vectors) or arising from the geometrical and inertial parameters of the manipulator (e.g., parallel/perpendicular joint axes, zero-length links or sparse inertia tensor) (Kircanski [1988]). Also the computations related to the first part of the manipulator are based on classical (non-matrix) L-E equations. There are no rotation matrices, which leads to reduction in the computation burden of the dynamic equations, as will be shown in next sections.

To summarize, the author here included the following strategies to reduce the computation complexity:
1). To divide the robot manipulator into two part, an arm and a hand.
2). To use vector manipulation instead of matrices.
3). To use the symbolic form, and making full use of the PUMA configuration.

## 2.2 KINEMATICS OF THE PUMA560

The position (forward and inverse) of the PUMA can be decided by combining geometry and analysis together (Fu [1987]). An effective program has been written for the PUMA560 robot manipulator kinematics in the Automatic Control Group, with all the calculations being based on vector manipulations and no matrixes being involved. The key points are:

1). To separate the robot manipulator into two parts, an arm and a hand, calculating the kinematics of each individually.

2). To calculate, on-line, the direction of each joint axes $b_{i-1}$ (in this chapter, joints are numbered as 0, 1, ..., 5 while links are numbered as 1, 2, ..., 6) and the direction $d_i$ of each link recursively, using the technique of one vector rotating about the other.

### 2.2.1 Vector Rotation

Mathematics tools used for kinematics analysis is mainly vector rotation, and is outlined here for necessary background.

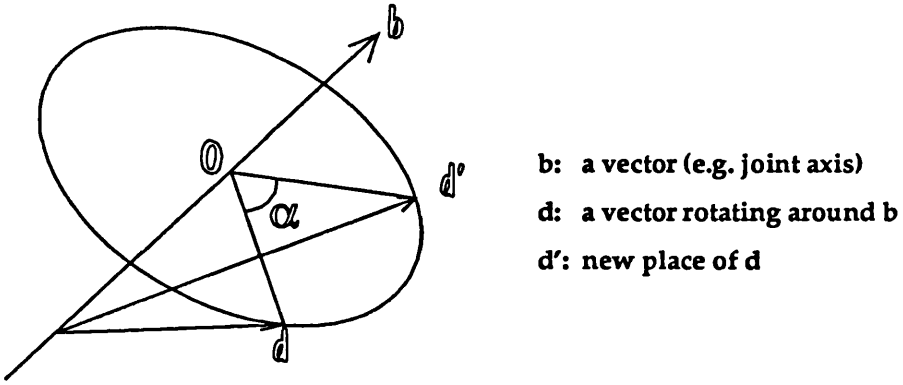



**Fig. 2.1 Vector rotation**

As shown in the figure, a vector, $d$, e.g. link direction, is rotating around a random vector in space $b$, e.g. a joint axis. A program routine is written for vector rotating. Giving the old vector $d$, vector $b$ and rotation angle $\alpha$ the new vector $d'$ can be calculated using the following format: $d'=rot\_vec(b,d,\alpha)$. A MODULA-2 program for doing vector rotation is listed below:

```
PROCEDURE rotVec(VAR pvec, nvec: VECTOR; VAR alpha: REAL) :
VECTOR;
...
nCP:= crossP(nvec, pvec);      (*Cross product of two vectors*)
pDotN:= dotP(pvec, nvec)*(1.0 - cos(alpha));
(*Dot product of two vectors*)
FOR i:= 1 TO 3 DO
  pvec[i]:= pDotN*nvec[i] + cos(alpha)*pvec[i] + sin(alpha)*nCP[i]
END;
RETURN pvec;
...
```

where **crossP** and **dotP** are two separate routines for doing vector cross multiplication and dot multiplication respectively.

### 2.2.2 Position-Orientation Kinematics of a Robotic Manipulator

The first three degrees of freedom decide the position of the PUMA560 robotic manipulator (can be extended as a general rule to other robotic manipulators). The last three degree of freedoms decide the orientation of the PUMA560 robotic manipulator end effector. If the displacement of each joint of the robotic manipulator is decided, the position and orientation of the arm can be uniquely decided (forward kinematics). The other way round is not true, that is, if the position and orientation of the arm is decided, the joint angles can not be uniquely decided (inverse problem).

Forward Kinematics of PUMA560

Forward kinematics means deciding the manipulator position and orientation according to the joint angles. As stated before most industrial manipulators can be divided into an arm and a hand, which can then be treated individually.

(1). Position

Supposing the angular displacements of joint 1, 2 and 3 are known, as shown in Fig. 2.2, then wrist position $P_3$ can be calculated recursively.

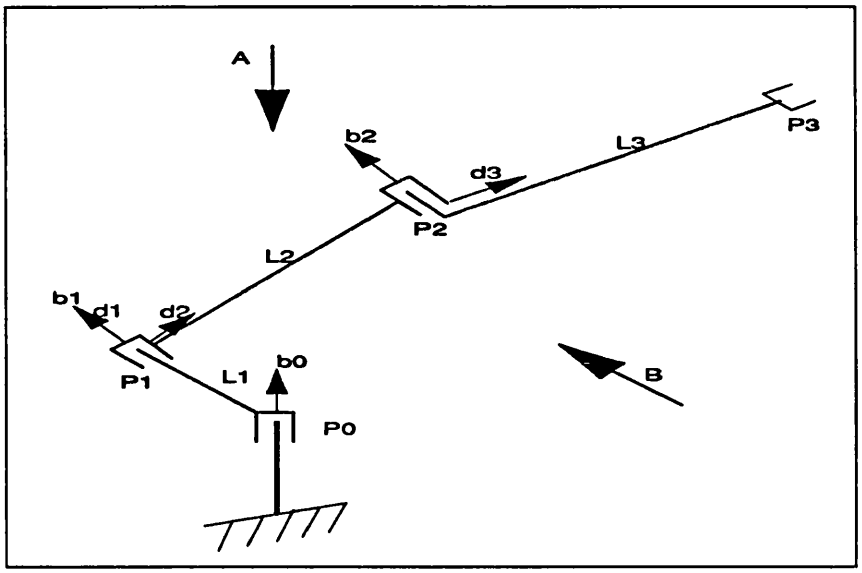



**Fig. 2.2 Calculate wrist position**

In Fig. 2.2, $b_{i-1}$ (bold characters in text stand for vectors too in this thesis) is *i-1*th joint direction, $d_i$ *i*th link direction and $P_i$ is *i*th joint position. $P_3$ is called the wrist position. $b_0$ is coincident to one (usually *z*) axis with the world coordinate and is not changing.

Rotating the first link around $b_0$ in a angle $\theta_1$, the position $P_1$ and orientation $b_1$ of the second joint can be decided using vector rotation described in the former sub-section. At the same time the orientation $b_2$ of joint 3 can be decided as $b_2=b_1$, because of the special PUMA geometry. By rotating $d_2$ around $b_1$, $P_2$ can be calculated, and so is the wrist position $P_3$. This seems straight forward and refer to Lovell [1990] for a detailed description.

(2). Orientation

After the position of the wrist of the manipulator has been calculated, the orientation of the manipulator can be calculated according the joint angles of the last three joints.

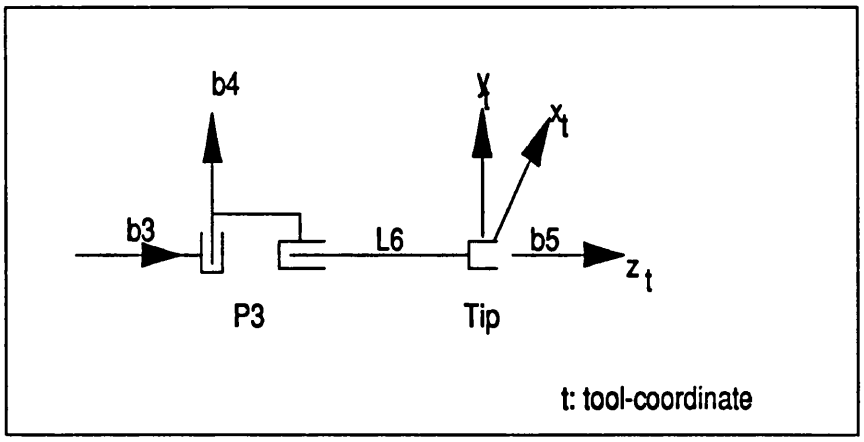



**Fig. 2.3 Calculate orientation**

A similar calculation of the orientations of the last three joints can be made by using vector rotation. Once wrist position $P_3$ and direction of link three $d_3$ have been calculated in the former part, the robotic end-effector, i.e. tip-position and tip-orientation (or $b_5$) can be decided recursively. As shown in Fig. 2.3, direction of joint 5, $b_4$ can be calculated by rotating it, in $\theta_4$, around joint 4 direction $b_3$, which equals to $d_3$ (Fig. 2.2). Direction of joint six $b_5$, which is the same direction as the direction of joint six, can be calculated by rotation it around $b_4$ in the angle $\theta_5$. And lastly, the tip-position and orientation can be calculated.

## Inverse Kinematics of PUMA560

Given a position/orientation in space, the calculation of the corresponding joint angles is called the inverse kinematic problem. The inverse problem is not so straight forward as the forward kinematics problem, with multi-solutions. The Multi-solution problem is solved by choosing a specific configuration convenient for arm manipulation.

As stated before, the manipulator can be divided into two parts, an arm and a hand. Because of this, this research uses geometry together with simple

analysis method to solve the inverse problem. No 4x4 homogeneous matrix transformation is involved. The time for inverse problem is kept to a minimum.

Given a position and orientation to the end-effector of a robotic manipulator, the wrist position $P_3$ can be calculated, which is then used to derive the first three joint angles. The last three joint angles can be derived by the tip-orientation.

(1). Position

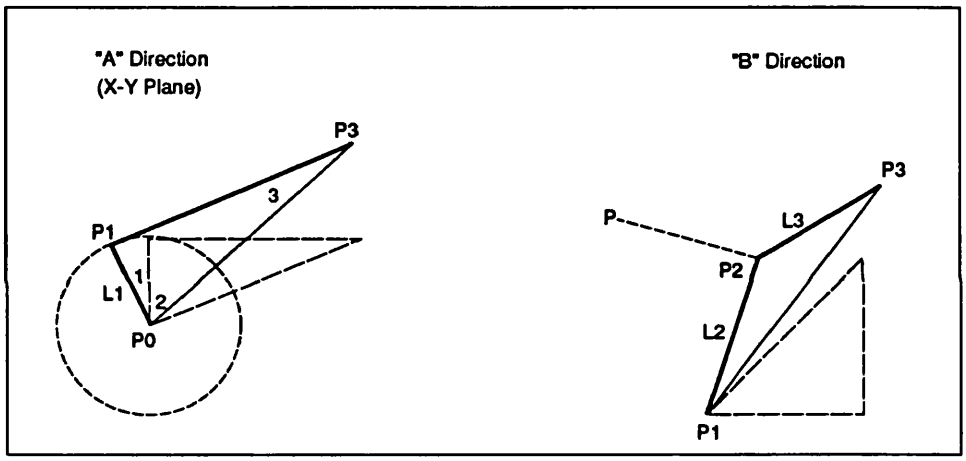



**Fig. 2.4 Calculate first three joint angles**

Suppose the wrist position $P_3$ is specified. Now, it is needed to decide the angles of joint 1, 2 and 3. In Fig. 2.4, the left-hand side is the projection of Fig. 2.2 in the "A" direction and the right-hand side, projection in the "B" direction. And $L_1$ is the first link length, $P_3$ is wrist position projection seeing from "A" or "B" direction, the dashed-line show the initial state and the solid line, the new state of the manipulator.

As shown in the left-hand side Fig. 2.4, once $P_3$ being located, its projection on the $x$-$y$ plane is decided. $P_0$ is the arm shoulder position. Drawing a tangent line to the circle origin at $P_0$, radius, $L_1$, through point $P_3$, we can locate

the position of the second joint $P_1$. The first joint angle, which is the angle between initial link 1 and new link 1, can be easily solved using geometrical analysis.

The right-hand figure in Fig. 2.4 shows the projected image of arm seeing from direction "B". This is a plane which is perpendicular to link 1. Because of the PUMA special configuration, link 2 and 3 will be in this plane --- called the "S" plane. What has to be pointed out is that the PUMA configuration is chosen as concave shape, as shown in initial shape in the right-hand figure. This configuration is suitable for upper operation. However, in this project, the robotic manipulator is mainly used to do lower operations. The configuration is then chosen as a convex shape.

The angular displacements of joint 2 and joint 3 are calculated as following: $P_3$ in the "S" plane is known, and $P_1$ is decided in the first step as described. Taking $P_1$, $L_2$ and $P_3$, $L_3$ as the origins and radii , draw two circles, which will have two cross points. Choose one of the cross points as position of joint 3, $P_2$, according to the configuration required as shown in the figure. The angular displacements of joint 2 and 3 are calculated in the same way as described in the case of joint 1.

(2). Orientation

Given an orientation in space, calculating the joint angles of the last three joints of PUMA560 is called the orientation inverse problem. This problem is more difficult than the former one. Redrawing Fig. 2.3 by rotating the tool-coordinate 90 degrees around its $y_t$ axis, we can get the following Fig. 2.5.

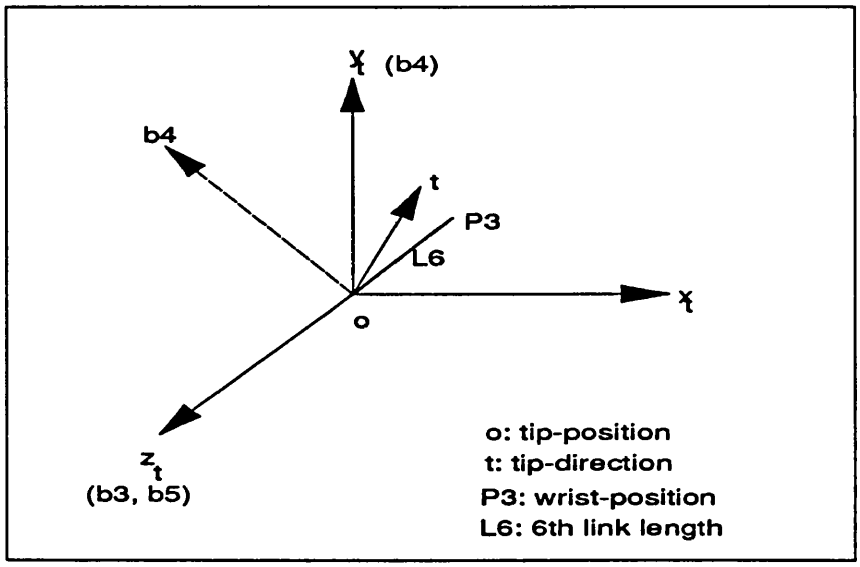



**Fig. 2.5 Calculate last three joint angles**

where $t$ is the tip-direction specified, initial $b_3$, $b_5$ and $z_t$ are coincident, and so are $b_4$ and $y_t$. The angle between vector $t$ and axis $b_3$ decide the required angular displacement of joint 5. The cross product of vector $t$ and axis $b_3$ is the new axis of joint 5, $b_4$. The angle between new and old axis of joint 5 decide the angular displacement of joint 4. A piece of MODULA-2 program for doing this is listed below:

```
...
Angle5:= acos2(dotP(tipDirect, b3), 1.0);
(*Calculate angular displacement of joint 5*)
...
b4:= crossP(tipDirect, b3);
(*Calculate new axis of joint 5*)
...
Angle4:= acos2(dotP(b4, b4Init), 1.0);
(*Calculate angular displacement of joint 4*)
...
```

where **acos2** is a MODULA-2 module to calculate angle between two vectors, and **dotP** and **crossP** are dot product and cross product of two vectors respectively.

### 2.2.3 Motion Kinematics of a Robotic Manipulator

When dealing with arm dynamics, the velocities and accelerations of each joint have to be calculated to judge the dynamic response, from which suitable controllers can be assessed to acquire the best performances.

For simplicity, a line diagram of the PUMA type robot manipulator is drawn as Fig. 2.6. A recursive equation can be obtained as follows (Asada [1986]):

$$\begin{aligned}
\vec{v}_{i+1} &= \vec{v}_i + \vec{\omega}_{i+1} \times \vec{r}_{i,i+1} \\
\vec{\omega}_{i+1} &= \vec{\omega}_i + \dot{q}_{i+1} \cdot \vec{b}_i \\
\vec{a}_{i+1} &= \vec{a}_i + \vec{\omega}_{i+1} \times \vec{r}_{i,i+1} + \vec{\omega}_{i+1} \times (\vec{\omega}_{i+1} \times \vec{r}_{i,i+1}) \\
\vec{\omega}_{i+1} &= \vec{\omega}_i + \ddot{q}_{i+1} \cdot \vec{b}_i + \vec{\omega}_i \times \dot{q}_{i+1} \cdot \vec{b}_i
\end{aligned} \tag{2-1}$$

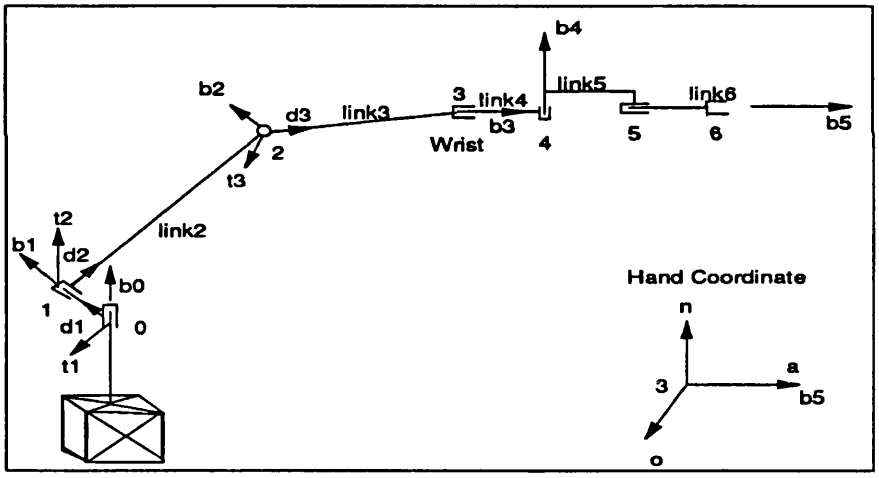



Fig. 2.6 The coordinate systems of PUMA560

For the PUMA, all the joints are revolute types, so we can easily derive from equation (2-1) the following equations:

$$\begin{aligned}
\vec{\omega}_k &= \sum_{i=1}^{k} \dot{q}_i \cdot \vec{b}_{i-1} \\
\vec{v}_k &= \sum_{i=1}^{k} \dot{q}_i \cdot \vec{b}_{i-1} \times \vec{r}_{i-1,k}
\end{aligned} \tag{2-2}$$

The acceleration of each joint and the angular acceleration of each link can be obtained by differentiating equations (2-2). As can be seen in the next section, it is necessary to give the velocity and acceleration of the wrist (i.e. tip of the third link):

$$\vec{v}_3 = \sum_{i=1}^{3} \dot{q}_i \cdot \vec{J}_i$$
$$\vec{a}_3 = \sum_{i=1}^{3} [\ddot{q}_i \cdot \vec{J}_i + \dot{q}_i \cdot \dot{\vec{J}}_i] \qquad (2\text{-}3)$$

where $\vec{J}_i$ is element of Jacobean matrix and,

$$\vec{J}_i = \vec{b}_{i-1} \times \vec{r}_{i-1,3}$$
$$\dot{\vec{J}}_i = \vec{b}_{i-1} \times \vec{r}_{i-1,3} + \vec{b}_{i-1} \times \dot{\vec{r}}_{i-1,3}$$
$$\vec{b}_{i-1} = \vec{\omega}_{i-1} \times \vec{b}_{i-1}$$

$k$ is the $k$th joint, and

$$\vec{r}_{i-1,k} = \vec{r}_{0,k} - \vec{r}_{0,i-1} = \vec{v}_k - \vec{v}_{i-1} \qquad (2\text{-}4)$$

## 2.3 DYNAMICS MODEL OF THE PUMA560

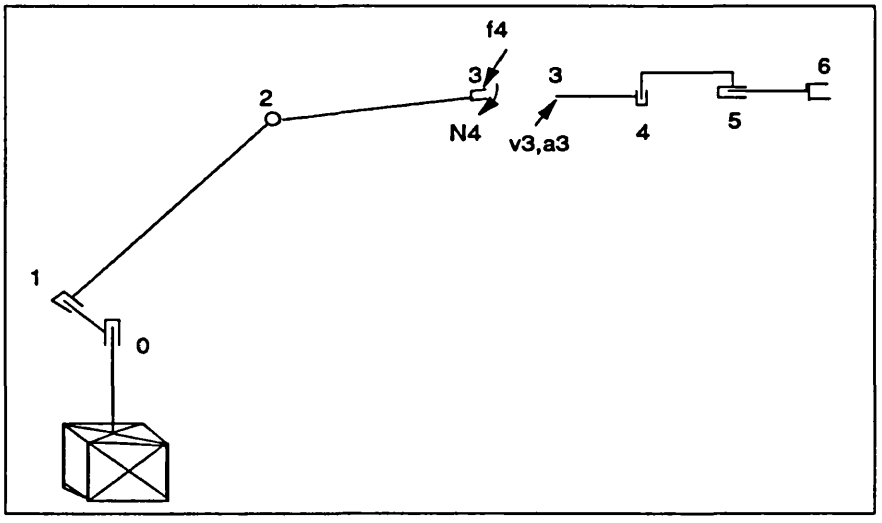



**Fig.2.7 Partitioned manipulator model**

In this section the dynamics equations of the PUMA type robot manipulator will be derived based on Horak [1984]'s scheme of separating the manipulator into two parts, using the non-matrix form of the L-E equations in symbolic forms.

As shown in Fig.2.7, if $v_3$ and $a_3$ (2-3) are evaluated, the dynamics of the second part can be decided by the recursive N-E equations, thus $f_4$ and $N_4$ can be calculated ( Horak [1984]). The dynamics of the first part can be exactly computed if the force ($f_4$) and the moment ($N_4$) applied on it by the second part are known. The idea of Horak's can be expressed by the following steps:

(1). Evaluate the velocity and the acceleration of the wrist.

(2). Compute the dynamics of the second part of the manipulator, i.e., links 4,5 and 6. This computation can be performed exactly once the results of the step one are available. At this point the torques of actuators 4,5 and 6 are available, and so are $f_4$ and $N_4$.

(3). Compute the dynamics of the first part of the manipulator, i.e., links 1,2 and 3, excluding the contribution due to the force and the moment the second part of the manipulator applies on link 3.

(4). Add the dynamic effects of the second part.

### 2.3.1 Dynamics of the Second Part

The recursive Newton-Euler equation is as following:

$$\vec{f}_{i-1,i} - \vec{f}_{i,i+1} + m_i \cdot \vec{g} - m_i \cdot \vec{v}_{ci} = 0$$
$$\vec{N}_{i-1,i} - \vec{N}_{i,i+1} + \vec{r}_{i,ci} \times \vec{f}_{i,i+1} - \vec{r}_{i-1,ci} \times \vec{f}_{i-1,i} - [I_i] \times \vec{\omega}_i - \vec{\omega}_i \times ([I_i] \times \vec{\omega}_i) = 0 \quad (2\text{-}5)$$

where $[I_i]$ is the inertia tensor of link i (Table 2.1) i = 4,5,6. For a revolute joint, the drive torque is balanced with the coupling torque component of $N_{i-1,i}$ which is in the direction of its joint axis:

$$\tau_i = \vec{b}_{i-1} \cdot \vec{N}_{i-1,i} \quad (2\text{-}6)$$

From Table 2.2 it can be seen that the inertia of the last three joints are so small that they can be ignored. It is also noticed that the axes of last three joints coincide at a point, which means that:

$$\vec{r}_{3,c4}=\vec{r}_{4,c4}=\vec{r}_{4,c5}=\vec{r}_{5,c5}=\vec{r}_{5,c6}=0$$

**Table 2.2 Link mass and its centre of PUMA560**

| $Link_i$ | Mass (kg) | in $b_{i-1}$ (m) | in $d_i$ (m) | in $t_i$ (m) |
|---|---|---|---|---|
| 1 | 17.270 | 0.0 | 0.102 | 0.0 |
| 2 | 15.065 | 0.0 | 0.068 | 0.0 |
| 3 | 5.716 | 0.0 | 0.070 | 0.0 |
| 3 and Wrist | 6.830 | 0.0 | 0.143 | 0.0 |
| 4 | 1.482 | -0.019 | 0.0 | 0.0 |
| 5 | 0.372 | 0.0 | 0.0 | 0.0 |
| 6 | 0.260 | 0.032 | 0.0 | 0.0 |

Then equation (2-5) can be rewritten like this:

$$\vec{N}_{i-1,i}=\vec{N}_{i,i+1}$$
$$\vec{f}_{3,4}=(m_4+m_5+m_6)(\vec{v}_3-\vec{g}) \qquad (2\text{-}7)$$

which means that the dynamic effects of the last three joints can just be simplified to one single mass ($m_4 + m_5 + m_6$) seen from the first part. So the force applied to the first part by the second part $f_4 = f_{3,4}$ and $N_4 = 0$ approximately, if no external torque applied by environment.

To derive the torques of the second part is not equally easy. However, some simplification can also be made. The second and the third term of (2-5) can be ignored because of the configuration of the last three joints of the PUMA560 ($\vec{r}_{3,c4}=\vec{r}_{4,c4}=\vec{r}_{4,c5}=\vec{r}_{5,c5}=\vec{r}_{5,c6}=0$). The last term $\vec{\omega}_i\times([I_i]\times\vec{\omega}_i)$ in the second equation of (2-5) can also be ignored compared with the fourth term, because the velocity and acceleration are two small, then:

$$\vec{N}_{5,6} = [I_6] \times \vec{\omega}_6$$
$$\vec{N}_{4,5} = [I_5] \times \vec{\omega}_5 + \vec{N}_{5,6}$$
$$\vec{N}_{3,4} = [I_4] \times \vec{\omega}_4 + \vec{N}_{4,5}$$

Where $[I_i]$ is the diagonal inertia matrics with elements in Table 2.3.

**Table 2.3 Link inertia of PUMA560**

| Link $_{i=}$ | $I_{ibi-1}$ (kg m$^2$) | $I_{idi}$ (kg m$^2$) | $I_{iti}$ (kg m$^2$) |
|---|---|---|---|
| 1 | 0.350 | - | - |
| 2 | 0.539 | 0.130 | 0.524 |
| 3 | 0.086 | 0.0125 | 0.066 |
| 3 and Wrist | 0.19200 | 0.01540 | 0.21200 |
| 4 | 0.00137 | 0.01993 $=I_{4n}$ | 0.01983 $=I_{4t4}$ |
| 5 | 0.00084 | 0.00037 $=I_{5d5}$ | 0.00023 $=I_{5o}$ |
| 6 | 0.00040 | 0.00043 $=I_{6n}$ | 0.00034 $=I_{6o}$ |

Where $t_4 = b_3 \times n$, $d_5 = -a$. Using (2-1) to get the joint angular accelerations of the last links and express them in the each joint coordinate:

$$\vec{\omega}_4 = \dot{\omega}_{4b3}\vec{b}_3 + \dot{\omega}_{4n}\vec{n} + \dot{\omega}_{4t4}\vec{t}_4$$
$$\vec{\omega}_5 = \dot{\omega}_{5b4}\vec{b}_4 + \dot{\omega}_{5d5}\vec{d}_5 + \dot{\omega}_{5o}\vec{o}$$
$$\vec{\omega}_6 = \dot{\omega}_{6b5}\vec{b}_5 + \dot{\omega}_{6n}\vec{n} + \dot{\omega}_{6o}\vec{o} \qquad (2\text{-}8)$$

According to (2-6), only the $b_{i-1}$ division of angular velocity contributes to the torque. And from Table 2.3, the dynamic effect of joint 5 and joint 6 on joint 4 can be ignored, then:

$$\tau_4 = I_{4b3} \cdot \dot{\omega}_{4b3} = 0.0137\dot{\omega}_{4b3}$$
$$\tau_5 = I_{5b4} \cdot \dot{\omega}_{5b4} + I_{6n} \cdot \dot{\omega}_{6n} = 0.0084\dot{\omega}_{5b4} + 0.00043\dot{\omega}_{6n} \doteq 0.0084\dot{\omega}_{5b4}$$
$$\tau_6 = I_{6b5} \cdot \dot{\omega}_{6b5} = 0.0004\dot{\omega}_{6b5} \qquad (2\text{-}9)$$

Where:

$$
\begin{aligned}
\dot{\omega}_{4b3} &= \ddot{q}_1(\vec{b}_0 \cdot \vec{b}_3) + \ddot{q}_4 \\
\dot{\omega}_{4n} &= \ddot{q}_1(\vec{b}_0 \cdot \vec{n}) + (\ddot{q}_2 + \ddot{q}_3)(\vec{b}_2 \cdot \vec{n}) \\
\dot{\omega}_{4i4} &= \ddot{q}_1(\vec{b}_0 \cdot \vec{i}_4) + (\ddot{q}_2 + \ddot{q}_3)(\vec{b}_2 \cdot \vec{i}_4) \\
\dot{\omega}_{5b4} &= \dot{\omega}_{4n} + \ddot{q}_5 \\
\dot{\omega}_{5d5} &= \dot{\omega}_{4b3}(\vec{b}_3 \cdot \vec{d}_5) + \dot{\omega}_{4i4}(\vec{i}_4 \cdot \vec{d}_5) \\
\dot{\omega}_{6b5} &= \dot{\omega}_{5d5} + \ddot{q}_6
\end{aligned}
\qquad (2\text{-}10)
$$

(2-10) are derived according to the last equation of (2-1), the velocity term of which is ignored as in force control the velocity are small compared with acceleration. Table 2.2 and Table 2.3 are acquired according to Armstrong [1986] and Mon [1988]. Some idealizations have been made which would have no effect in the simulation study.

### 2.3.2 Dynamics of the First Part

Once the force applied by the second part has been calculated the dynamics of the first part can be decided exactly. Suppose K is the total kinetic energy of the first part and U, the total potential energy of the first part, the L-E equation is as following, excluding the effects of the second part:

$$
\begin{aligned}
\tau_j &= d(\partial K/\partial \dot{q}_j)/dt - (\partial K/\partial q_j) + (\partial U/\partial q_j) \\
K &= \sum_{i=1}^{3} K_i \\
U &= \sum_{i=1}^{3} (m_i \cdot \vec{g} \cdot \vec{r}_{0,ci})
\end{aligned}
\qquad (2\text{-}11)
$$

It is rather tedious to rewrite all the equations here. A report written by the author in Wang [1989] detailed the procedures to get all the dynamics equation of the first part of the robot manipulators. Here, only the strategies used in the report are rewritten as follows.

The principle axes of each link have the same direction as the joint coordinate, and are originated at the centre of mass. It is convenient to express the angular velocity of each joint at its joint coordinate for convenience. From equation (2-2):

$$\begin{aligned}
&\vec{\omega}_1 = \dot{q}_1 \cdot \vec{b}_0 \\
&\vec{\omega}_2 = \dot{q}_1 \cdot \vec{b}_0 + \dot{q}_2 \cdot \vec{b}_1 = \dot{q}_1 \cdot D_{02} \cdot \vec{d}_2 + \dot{q}_1 \cdot T_{02} \cdot \vec{t}_2 + \dot{q}_2 \cdot \vec{b}_1 \\
&\vec{\omega}_3 = \dot{q}_1 \cdot \vec{b}_0 + \dot{q}_2 \cdot \vec{b}_1 + \dot{q}_3 \cdot \vec{b}_2 \\
&\quad = \dot{q}_1 \cdot D_{03} \cdot \vec{d}_3 + \dot{q}_1 \cdot T_{03} \cdot \vec{t}_3 + (\dot{q}_2 + \dot{q}_3) \cdot \vec{b}_2
\end{aligned} \tag{2-12}$$

It can be seen from the equations above that no division of $\dot{q}_1$ on the direction of $b_1$ and $b_2$ because $b_1$ and $b_2$ is perpendicular to $b_0$. where

$$\begin{aligned}
&D_{0i} = \vec{b}_0 \bullet \vec{d}_i \\
&T_{0i} = \vec{b}_0 \bullet \vec{t}_i
\end{aligned} \tag{2-13}$$

The kinetic energy of the first part can be written as follows:

$$\begin{aligned}
&K_1 = (1/2) \cdot m_1 \cdot \vec{v}_{c1}^2 + (1/2) \cdot I_{1b0} \cdot \dot{q}_1^2 \\
&K_2 = (1/2) \cdot m_2 \cdot \vec{v}_{c2}^2 + (1/2) \cdot I_{2b1} \cdot \dot{q}_2^2 + (1/2) \cdot \dot{q}_1^2 \cdot (I_{2d2} \cdot D_{02}^2 + I_{2i2} \cdot T_{02}^2) \\
&K_3 = (1/2) \cdot m_3 \cdot \vec{v}_{c3}^2 + (1/2) \cdot I_{3b2} \cdot (\dot{q}_2 + \dot{q}_3)^2 \\
&\quad + (1/2) \cdot \dot{q}_1^2 \cdot (I_{3d3} \cdot D_{03}^2 + I_{3i3} \cdot T_{03}^2)
\end{aligned} \tag{2-14}$$

where $I_{idj}$, $I_{itj}$ and $I_{ibj-1}$ (see Table 2.3) are the inertia of the $i$th link rotating around direction $d_j$, $t_j$ and $b_{j-1}$ respectively. The potential energy is:

$$U = \sum_{i=1}^{k} m_i \cdot \vec{g} \bullet \vec{r}_{0,ci}$$

$$G_j = \partial U / \partial q_j$$

Therefore, it is easy to get:

$$
\begin{aligned}
G_1 &= 0 \\
G_2 &= m_2 \cdot g \cdot T_{02} \cdot r_{1,c2} + m_3 \cdot g \cdot (T_{02} \cdot r_{1,2} + T_{03} \cdot r_{2,c3}) \\
G_3 &= m_3 \cdot g \cdot T_{03} \cdot r_{2,c3}
\end{aligned} \tag{2-15}
$$

The torques of the first joints can be written as below (Wang [1989]):

$$
\begin{aligned}
\tau_1 &= \sum_{i=1}^{3} (m_i \cdot \vec{a}_{ci} \bullet \vec{\vec{v}}_{ci,1}) \\
&\quad + \ddot{q}_1 \cdot (I_{1b0} + D_{02}^2 \cdot I_{2d2} + T_{02}^2 \cdot I_{2t2} + D_{03}^2 \cdot I_{3d3} + T_{03}^2 \cdot I_{3t3}) \\
&\quad + 2 \cdot \dot{q}_1 \cdot \dot{q}_2 \cdot [T_{02} \cdot D_{02} \cdot (I_{2t2} - I_{2d2}) + T_{03} \cdot D_{03} \cdot (I_{3t3} - I_{3d3})] \\
&\quad + 2 \cdot \dot{q}_1 \cdot \dot{q}_3 \cdot D_{03} \cdot T_{03} \cdot (I_{3t3} - I_{3d3}) \\
\tau_2 &= \sum_{i=2}^{3} (m_i \cdot \vec{a}_{ci} \bullet \vec{\vec{v}}_{ci,2}) \\
&\quad + \ddot{q}_2 \cdot (I_{2b1} + I_{3b2}) + \ddot{q}_3 \cdot I_{3b2} \\
&\quad + \dot{q}_1^2 \cdot [D_{02} \cdot T_{02} \cdot (I_{2d2} - I_{2t2}) + D_{03} \cdot T_{03} \cdot (I_{3d3} - I_{3t3})] \\
&\quad + G_2 \\
\tau_3 &= m_3 \cdot \vec{a}_{c3} \bullet \vec{\vec{v}}_{c3,3} \\
&\quad + \ddot{q}_2 \cdot I_{3b2} + \ddot{q}_3 \cdot I_{3b2} \\
&\quad + \dot{q}_1^2 \cdot D_{03} \cdot T_{03} \cdot (I_{3d3} - I_{3t3}) \\
&\quad + G_3
\end{aligned} \tag{2-16}
$$

Where:

$$
\begin{aligned}
\vec{\vec{v}}_{ci,j} &= \partial \vec{v}_{ci} / \partial \dot{q}_j = \partial \vec{r}_{0,ci} / \partial q_j = \vec{b}_{j-1} \times \vec{r}_{j-1,ci} \\
\vec{v}_{ci} &= \sum_{j=1}^{i} (\partial \vec{r}_{0,ci} / \partial q_j) \cdot \dot{q}_j = \sum_{j=1}^{i} \vec{\vec{v}}_{ci,j} \cdot \dot{q}_j
\end{aligned} \tag{2-17}
$$

and $I_{idj}$, $I_{itj}$ and $I_{ibj-1}$ are the inertia of the $i$th link (see Table 2.3). And let:

$$
\begin{aligned}
\tau_{jL} &= \sum_{i=1}^{3} m_i (\vec{a}_{ci} \times \vec{\vec{v}}_{ci,j}) \\
w_{jk}^{(i)} &= \vec{\vec{v}}_{ci,j} \bullet \vec{\vec{v}}_{ci,k} \\
\vec{w}_{jk}^{(i)} &= \vec{\vec{v}}_{ci,j} \times \vec{\vec{v}}_{ci,k}
\end{aligned} \tag{2-18}
$$

And it is easy to notice that:

$$
\begin{aligned}
&w_{jk}^{(i)} = w_{kj}^{(i)} \\
&\overrightarrow{w}_{jk}^{(i)} = -\overrightarrow{w}_{kj}^{(i)} \\
&(\vec{A} \times \vec{B}) \cdot \vec{C} = \vec{A} \cdot (\vec{B} \times \vec{C}) \\
&(\vec{A} \times \vec{B}) \cdot (\vec{C} \times \vec{D}) = (\vec{A} \cdot \vec{C}) \cdot (\vec{B} \cdot \vec{D}) - (\vec{A} \cdot \vec{D}) \cdot (\vec{B} \cdot \vec{C})
\end{aligned}
\tag{2-19}
$$

then

$$
\begin{aligned}
\tau_{1L} &= \ddot{q}_1 (m_1 \cdot w_{11}^{(1)} + m_2 \cdot w_{11}^{(2)} + m_3 \cdot w_{11}^{(3)}) + \ddot{q}_2 \cdot (m_2 \cdot w_{21}^{(2)} + m_3 \cdot w_{21}^{(3)}) + \ddot{q}_3 \cdot m_3 \cdot w_{31}^{(3)} \\
&\quad + \dot{q}_2^2 \cdot m_2 (\vec{b}_1 \cdot \overrightarrow{w}_{21}^{(2)}) + \dot{q}_3^2 \cdot m_3 \cdot (\vec{b}_2 \cdot \overrightarrow{w}_{21}^{(3)}) \\
&\quad + 2 \cdot m_3 [\dot{q}_1 \cdot \dot{q}_2 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{21}^{(3)}) + \dot{q}_1 \cdot \dot{q}_3 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{21}^{(3)}) + \dot{q}_2 \cdot \dot{q}_3 \cdot (\vec{b}_1 \cdot \overrightarrow{w}_{31}^{(3)})] \\
\tau_{2L} &= \ddot{q}_1 \cdot (m_2 \cdot w_{12}^{(2)} + m_3 \cdot w_{12}^{(3)}) + \ddot{q}_2 \cdot (m_2 \cdot w_{22}^{(2)} + m_3 \cdot w_{22}^{(3)}) + \ddot{q}_3 \cdot m_3 \cdot w_{32}^{(3)} \\
&\quad + \dot{q}_1^2 \cdot [m_2 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{12}^{(2)}) + m_3 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{12}^{(3)})] + \dot{q}_3^2 \cdot m_3 (\vec{b}_2 \cdot \overrightarrow{w}_{32}^{(3)}) \\
&\quad + 2 \cdot m_3 \cdot [\dot{q}_1 \cdot \dot{q}_3 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{32}^{(3)}) + \dot{q}_2 \cdot \dot{q}_3 \cdot (\vec{b}_1 \cdot \overrightarrow{w}_{32}^{(3)})] \\
\tau_{3L} &= m_3 \cdot [\ddot{q}_1 \cdot w_{13}^{(3)} + \ddot{q}_2 \cdot w_{23}^{(3)} + \ddot{q}_3 \cdot w_{33}^{(3)} \\
&\quad + \dot{q}_1^2 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{13}^{(3)}) + \dot{q}_2^2 \cdot (\vec{b}_1 \cdot \overrightarrow{w}_{23}^{(3)}) + 2 \cdot \dot{q}_1 \cdot \dot{q}_2 \cdot (\vec{b}_0 \cdot \overrightarrow{w}_{23}^{(3)})]
\end{aligned}
\tag{2-20}
$$

As has been stated in the last section, the dynamic effect of the last three joints behaves just like a mass (m4+m5+m6) seen from the first part approximately. So, if we change m3 in equation (2-20) to 6.830 kg (Table 2.2), $I_3$ in equation (2-16) to the fourth row of Table 2.3, the dynamics effects of the first part can be reasonably thought to have been included.

## 2.4 SUMMARY

Equation (2-9), (2-10), (2-18) and (2-20) are the dynamics equations of the PUMA560, which for convenience can be rewritten as below:

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\\ \tau_5\\ \tau_6\end{bmatrix} = \begin{bmatrix} m11 & m21 & m31 & 0 & 0 & 0\\ m12 & m22 & m32 & 0 & 0 & 0\\ m13 & m23 & m33 & 0 & 0 & 0\\ m14 & 0 & 0 & m44 & 0 & 0\\ m15 & m25 & m35 & 0 & m55 & 0\\ m16 & m26 & m36 & m46 & 0 & m66 \end{bmatrix} \cdot \begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\\ \ddot{q}_4\\ \ddot{q}_5\\ \ddot{q}_6\end{bmatrix} + \begin{bmatrix}0\\ G_1\\ G_2\\ 0\\ 0\\ 0\end{bmatrix} \tag{2-21}$$

$$+[\dot{q}_1,\dot{q}_2,\dot{q}_3,\dot{q}_4,\dot{q}_5,\dot{q}_6] \cdot \begin{bmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & 0 & 0 & 0\\ h_{21}^{(2)} & h_{22}^{(2)} & h_{23}^{(2)} & 0 & 0 & 0\\ h_{31}^{(3)} & h_{32}^{(3)} & h_{33}^{(3)} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\\ \dot{q}_4\\ \dot{q}_5\\ \dot{q}_6\end{bmatrix}$$

The formulations and the calculations of the equation (2-21) are shown in the appendix table A2.1. The velocity term of the above equation can be ignored in force control as the velocity will be small compared with the acceleration, as will be shown in Chapter 5.

This is the explicit dynamic equation of the PUMA type robot manipulator. In the programming of these equations, further reduction in calculations can be accomplished using some programming tricks.

A total of 246 multiplications and 172 additions are needed. The computation efficiency is accomplished mainly by:

1). To partition the robot manipulator into two parts, an arm and a hand, with some simplification being made for the interaction between these two parts.
2). As shown in (2-8) and (2-12), to express the link angular accelerations and velocities in the joint coordinate rather than in the Cartesian coordinate.
3). To make full use of the special geometrical configuration of the PUMA type robot manipulator, like $b_1$ equals to $b_2$ etc.
4). To express the equations symbolically.

Armstrong [1986] obtained a set of dynamic equations for the PUMA560 robot manipulator with total calculation being 305 by abbreviating the original L-E equations. However, his method for abbreviation is complicated to understand and his results are difficult to verify. The method presented here is easy to understand, easy to verify and can be extended for other robot manipulator geometries with little change.

## Appendix 2.1

**Table A2.1 Formulations and calculations of (2-21)**

| Terms | Formulations | Calculations* |
|---|---|---|
| m11 | $m_1 w_{11}^{(1)} + m_2 w_{11}^{(2)} + m_3 w_{11}^{(3)}$ $+I_{1b0} + D_{02}^2 I_{2d2} + T_{02}^2 I_{2i2} + D_{03}^2 I_{3i3} + T_{03}^2 I_{3i3}$ | 15 & 7 |
| m21 | $m_2 w_{21}^{(2)} + m_3 w_{21}^{(3)}$ | 2 & 1 |
| m31 | $m_3 w_{31}^{(3)}$ | 1 & 0 |
| m12 | = m21 | 0 & 0 |
| m22 | $m_2 w_{22}^{(2)} + m_3 w_{22}^{(3)} + I_{2b1} + I_{3b2}$ | 2 & 3 |
| m32 | $m_3 w_{32}^{(3)} + I_{3b2}$ | 1 & 1 |
| m13 | = m31 | 0 & 0 |
| m23 | = m32 | 0 & 0 |
| m33 | $m_3 w_{33}^{(3)} + I_{3b2}$ | 1 & 1 |
| m14 | $I_{4b3}(\vec{b}_0 \cdot \vec{b}_3)$ | 4 & 3 |

| | | |
|---|---|---|
| m44 | $I_{4b3} = 0.0137$ | 0 & 0 |
| m15 | $I_{5b4}(\vec{b}_0 \cdot \vec{b}_4)$ | 4 & 3 |
| m25 | $I_{5b4}(\vec{b}_2 \cdot \vec{b}_4)$ | 4 & 3 |
| m35= | = m25 | 0 & 0 |
| m55 | $I_{5b4} = 0.0084$ | 0 & 0 |
| m16 | $I_{6b5}[-(\vec{b}_0 \cdot \vec{b}_3)(\vec{b}_3 \cdot \vec{b}_5) - (\vec{b}_0 \cdot \vec{t}_4)(\vec{t}_4 \cdot \vec{b}_5)]$ | 15 & 13 |
| m26 | $I_{6b5}[(\vec{b}_2 \cdot \vec{t}_4)(-\vec{t}_4 \cdot \vec{b}_5)]$ | 7 & 6 |
| m36= | = m26 | 0 & 0 |
| m46 | $I_{6b5}(-\vec{b}_3 \cdot \vec{b}_5)$ | 4 & 3 |
| m66 | $I_{6b5} = 0.0004$ | 0 & 0 |
| Rest $m_{ij}$ = | 0 | |

| | | |
|---|---|---|
| $h_{22}^{(1)}$ = | $m_2(\vec{b}_1 \cdot \vec{w}_{21}^{(2)})$ | 4 & 3 |
| $h_{33}^{(1)}$ = | $m_3(\vec{b}_2 \cdot \vec{w}_{21}^{(3)})$ | 4 & 3 |
| $h_{12}^{(1)}$ = | $2 \cdot m_3(\vec{b}_0 \cdot \vec{w}_{21}^{(3)})$ $+2 \cdot T_{02}D_{02}(I_{2i2} - I_{2d2}) + 2 \cdot T_{03}D_{03}(I_{3i3} - I_{3d3})$ | 9 & 7 |
| $h_{13}^{(1)}$ = | $2 \cdot m_3(\vec{b}_0 \cdot \vec{w}_{21}^{(3)} + 2 \cdot D_{03}T_{03}(I_{3i3} - I_{3d3})$ | 7 & 5 |
| $h_{23}^{(1)}$ = | $2 \cdot m_3(\vec{b}_1 \cdot \vec{w}_{31}^{(3)})$ | 5 & 3 |
| $h_{11}^{(2)}$ = | $m_2(\vec{b}_0 \cdot \vec{w}_{12}^{(2)}) + m_3(\vec{b}_0 \cdot \vec{w}_{12}^{(3)}$ $+D_{02}T_{02}(I_{2d2} - I_{2i2}) + D_{03}T_{03}(I_{3d3} - I_{3i3})$ | 12 & 11 |
| $h_{33}^{(2)}$ = | $m_3(\vec{b}_2 \cdot \vec{w}_{32}^{(3)}$ | 4 & 3 |

$h_{13}^{(2)} =$ $2 \cdot m_3(\vec{b}_0 \bullet \vec{w}_{32}^{(3)})$ 5 & 3

$h_{23}^{(2)} =$ $2 \cdot m_3(\vec{b}_1 \bullet \vec{w}_{32}^{(3)})$ 5 & 3

$h_{11}^{(3)} =$ $m_3(\vec{b}_0 \bullet \vec{w}_{13}^{(3)} + D_{03}T_{03}(I_{3d3} - I_{3i3})$ 5 & 5

$h_{22}^{(3)} =$ $m_3(\vec{b}_1 \bullet \vec{w}_{23}^{(3)})$ 4 & 3

$h_{12}^{(3)} =$ $2 \cdot m_3(\vec{b}_0 \bullet \vec{w}_{23}^{(3)})$ 5 & 3

Rest $h_{jk} =$ 0

$G_2 =$ $m_2 \cdot g \cdot T_{02} \cdot r_{1,c2} + m_3 \cdot g \cdot (T_{02} \cdot r_{1,2} + T_{03} \cdot r_{2,c3})$ 7 & 2

$G_3 =$ $m_3 \cdot g \cdot T_{03} \cdot r_{2,c3}$ 4 & 0

Rest $G_i =$ 0

Where:

$m_i$ the mass of the *i*th link (Table 2.1). Subtotal 140 &

$I_{ibj-1}$, $I_{iti}$, $I_{idi}$ are the inertia of *i*th link in the joint coordinate (Table 2.2). 98

And from (2-18) and (2-19):

$w_{11}^{(1)} =$ $\vec{v}_{c1,1} \bullet \vec{v}_{c1,1}$ 3 & 3

$w_{11}^{(2)} =$ $\vec{v}_{c2,1} \bullet \vec{v}_{c2,1}$ 3 & 3

$w_{11}^{(3)} =$ $\vec{v}_{c3,1} \bullet \vec{v}_{c3,1}$ 3 & 3

$w_{21}^{(2)} =$ $\vec{v}_{c2,2} \bullet \vec{v}_{c2,1}$ 3 & 3

$w_{21}^{(3)} =$ $\vec{v}_{c3,2} \bullet \vec{v}_{c3,1}$ 3 & 3

$w_{31}^{(3)} =$ $\vec{v}_{c3,3} \bullet \vec{v}_{c3,1}$ 3 & 3

$w_{12}^{(2)} =$ $w_{21}^{(2)}$ 0 & 0

$w_{12}^{(3)} =$ $w_{21}^{(3)}$ 0 & 0

| | | |
|---|---|---|
| $w_{22}^{(2)} =$ | $\vec{v}_{c2,2} \cdot \vec{v}_{c2,2}$ | 3 & 3 |
| $w_{22}^{(3)} =$ | $\vec{v}_{c3,2} \cdot \vec{v}_{c3,2}$ | 3 & 3 |
| $w_{32}^{(3)} =$ | $\vec{v}_{c3,3} \cdot \vec{v}_{c3,2}$ | 3 & 3 |
| $w_{13}^{(3)} =$ | $w_{31}^{(3)}$ | 0 & 0 |
| $w_{23}^{(3)} =$ | $w_{32}^{(3)}$ | 0 & 0 |
| $w_{33}^{(3)} =$ | $\vec{v}_{c3,3} \cdot \vec{v}_{c3,3}$ | 3 & 3 |
| $\vec{w}_{21}^{(2)} =$ | $\vec{v}_{c2,2} \times \vec{v}_{c2,1}$ | 6 & 3 |
| $\vec{w}_{21}^{(3)} =$ | $\vec{v}_{c3,2} \times \vec{v}_{c3,1}$ | 6 & 3 |
| $\vec{w}_{31}^{(3)} =$ | $\vec{v}_{c3,3} \times \vec{v}_{c3,1}$ | 6 & 3 |
| $\vec{w}_{32}^{(3)} =$ | $\vec{v}_{c3,3} \times \vec{v}_{c3,2}$ | 6 & 3 |
| $\vec{w}_{13}^{(3)} =$ | $-\vec{w}_{31}^{(3)}$ | 0 & 0 |
| $\vec{w}_{23}^{(3)} =$ | $-\vec{w}_{32}^{(3)}$ | 0 & 0 |
| | | Sub-total 54 & 42 |

Where: cf. (2-17)

| | | |
|---|---|---|
| $\vec{v}_{c1,1} =$ | $\vec{b}_0 \times \vec{r}_{0,c1}$ | 6 & 3 |
| $\vec{v}_{c2,1} =$ | $\vec{b}_0 \times \vec{r}_{0,c2}$ | 6 & 3 |
| $\vec{v}_{c3,1} =$ | $\vec{b}_0 \times \vec{r}_{0,c3}$ | 6 & 3 |
| $\vec{v}_{c2,2} =$ | $\vec{b}_1 \times \vec{r}_{1,c2}$ | 6 & 3 |
| $\vec{v}_{c1,1} =$ | $\vec{b}_1 \times \vec{r}_{1,c3}$ | 6 & 3 |
| $\vec{v}_{c3,3} =$ | $\vec{b}_2 \times \vec{r}_{2,c3}$ | 6 & 3 |
| | | Sub-total 36 & 18 |

| | | |
|---|---|---|
| Eq. (2-13) | $D_{02} = \vec{b}_0 \cdot \vec{d}_2$ | 3 & 3 |
| | $D_{03} = \vec{b}_0 \cdot \vec{d}_3$ | 3 & 3 |
| | $T_{02} = \vec{b}_0 \cdot \vec{t}_2$ | 3 & 3 |
| | $T_{03} = \vec{b}_0 \cdot \vec{t}_3$ | 3 & 3 |
| | Where b t d are provided by the kinematics analysis. | Sub-total 12 & 12 |

---

Total 242 multiplications and 170 additions.

* The first digit means number of multiplications, and the second one, additions.

# CHAPTER 3
# SELF-TUNING ADAPTIVE CONTROL

## 3.1 INTRODUCTION

In this chapter, self-tuning adaptive control design is detailed. Simulation work is based on a single input single output system (SISO) for simplicity. MODULA-2 and MATLAB are both used for the simulation task. A real robotic manipulator with 2 DOF doing a real task will be detailed in the next chapter. The main aim of this chapter is to have a discussion of the self-tuning adaptive control design, and its main use for non-linear SISO systems.

Advances in microprocessor technology have made digital control far more attractive than before (Greenshields [1989]). Digital controllers offer many advantages compared with their analogue counterparts. They allow complicated control laws, such as adaptive control, optimum control etc. to be possible and the resulting system performance to be much more accurate.

Self-tuning adaptive control is closely based on on-line recursive identification of the dynamic system, to estimate the dynamic behaviour of the plant. An approximate least square identification algorithm is introduced. Comparison

has been made with the normal least square identification. Some issues on identification have been discussed, especially initial conditions and persistent excitation and their effects on identification.

A general discrete transfer function (z-transfer) form for a second order system was established. A revised bi-linear transfer was used to acquire the digital counter part of PID controller. A subsequent new formulation of a self-tuning PID controller was acquired. Adaptation lies in the fact that the PID controller's gains can be adjusted according to changing process dynamics. Symbolic calculations have been carried out to acquire a new explicit relation between controller gains and process parameters.

With an explicit relation between the process parameters and the PID controller gains, much computation time can be saved. Most of the calculation can be done off-line before the explicit relation is acquired. The physical meaning of each term in the controller can be easily identified, as it has the same form as the analogue PID controller.

Three kind of digital PID controllers are given in this chapter. Three different version of self-tuning PID controllers are then inferred respectively according the different digital forms of the PID controllers. Their features are discussed and compared.

In section 1.4, non-linear identification and control is simply introduced for non-linear single input and single output systems. Its extension to multi-input multi-output systems is out the reach of this project and needs further research.

## 3.2 RECURSIVE LEAST SQUARE IDENTIFICATION

It is a key element in adaptive control to identify the process parameters on-line. The powerfulness of self-tuning adaptive control lies in its ability to adapt to

unknown systems. However, when using it in non-linear systems, as will be seen in the following, identification can cause a parasitic oscillation or chattering phenomena of control inputs.

In this section, the principle of identification of the parameters of the regression model will be discussed. Least square is the simplest and most widely used method, which was invented in the eighteenth century by Gauss to determine the orbits of planets.

### 3.2.1 Recursive Identification Method

The Recursive Least Squares (RLS) method (Ljung [1983]) is used to estimate the process parameters on-line, and is summarised by the following steps:

$$\begin{aligned} \theta(k) &= \theta(k-1) + L(k)[y(k) - \theta^T(k-1)\phi(k)] \\ L(k) &= \frac{P(k-1)\phi(k)}{\lambda + \phi^T(k)P(k-1)\phi(k)} \\ P(k) &= \lambda^{-1}(I - L(k)\phi(k)\phi^T(k))P(k-1) \end{aligned} \qquad (3\text{-}1)$$

where $\theta^T(k) = [a_1, a_2, b_1, b_2]$ is called the process parameter vector (as will be pointed out there are four parameters to be identified for a general second order system). $\phi^T(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)]$ is called the lagged inputs and outputs, and $\lambda$ is the forgetting factor. P, the co-variance matrix, is originally set to $\alpha$I, $\alpha$ is a large constant and I is the identity matrix.

As pointed out in the introduction chapter, the main purpose of this research is to use self-tuning adaptive control for a robotic manipulator, which is a typical highly non-linear, coupled system. As it has been understood, the underlying assumption of self-tuning adaptive control is that the system to be controlled is of linear nature with unknown constant parameters. This assumption restricts the application of self-tuning control to wider range and is unacceptable.

In the case of non-linear applications, the system parameters would keep changing. There are two situations (Astrom [1989]), one being the parameters are changing slowly. The other is that the system parameters are changing rapidly. The former difficult can be solved by adding a forgetting factor as already seen in equation (3-1). In the case of rapidly parameter changing Astrom [1989] suggested resetting of the matrix P by substitute it with a diagonal matrix $\alpha I$. Karam [1989] further extended this idea as to maintain matrix P to be diagonal in every sampling time by neglecting the computation of off-diagonal elements.

In this simplified recursive identification, the matrix P actually becomes a vector. Much computation burden can be avoided. However, this approximate approach causes identification degrading.

### 3.2.2 Effect of Initial Conditions

The above has shown two different approaches to identify process parameters. To evaluate these methods a program written in MATLAB is listed in Appendix 3.1.

As can see from the program, there are a lot of matrix calculations in the normal RLS identification, which are time consuming. If the approximate method suggested by Karam [1989] is used, where the calculation of the off-diagonal elements in matrix P is ignored, a lot of computation can been avoided and real-time implementation based on using micro-processor is then possible (Karam [1989]).

Fig. 3.1, 3.2, 3.3 and 3.4 show the results of different initial conditions using the two method respectively. The process is described by $y(k) = 2y(k-1) - y(k-2) + 0.00045u(k-1) + 0.00045u(k-2)$, with four parameters to be identified. The input signal is a random with magnitude $\pm 10$.

Rewriting the process in the standard form of ARMA model, the $z$ transfer function of the process would have the following form:

$$G_p(z^{-1}) = \frac{1 - 2z^{-1} + z^{-2}}{0.00045z^{-1}(1 + z^{-1})}$$

Comparing with the general transfer function form for second order system

$$G_p(z^{-1}) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{b_1 z^{-1} + b_2 z^{-2}}$$

It can be easily identify that: $a_1 = -2$; $a_2 = 1$; $b_1 = b_2 = 0.00045$. There are four plots in each of the figures of this chapter. Plot (a) is for parameter $a_1$, (b) for $a_2$, (c) for $b_1$ and (d) for $b_2$.

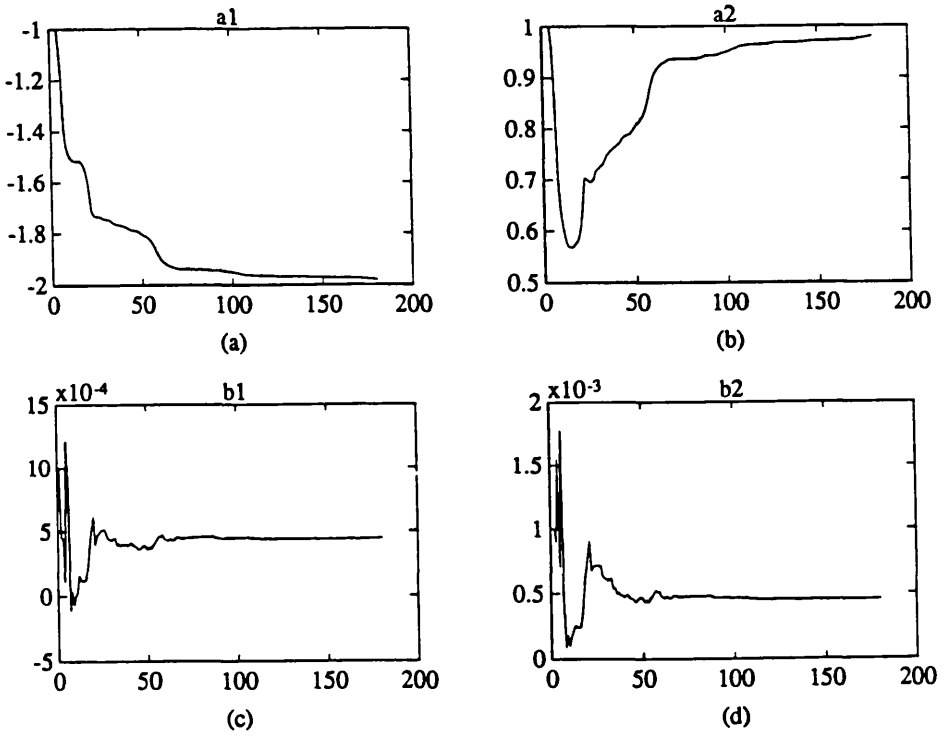



**Fig. 3.1 Normal RLS results**

The identification algorithm should be able to identify these parameter values. In Fig. 3.1 and 3.2, the initial values are chosen as: $a_1 = -1$; $a_2 = 1$; $b_1 = b_2 = 0.001$. From Fig. 3.1, it can be seen that if there is sufficient excitation, large initial deviations can even lead to the same true values of the system parameters. Even though well excitated, the approximate RLS algorithm (Fig. 3.2) can not find the true values if provided with wrong initial parameters.

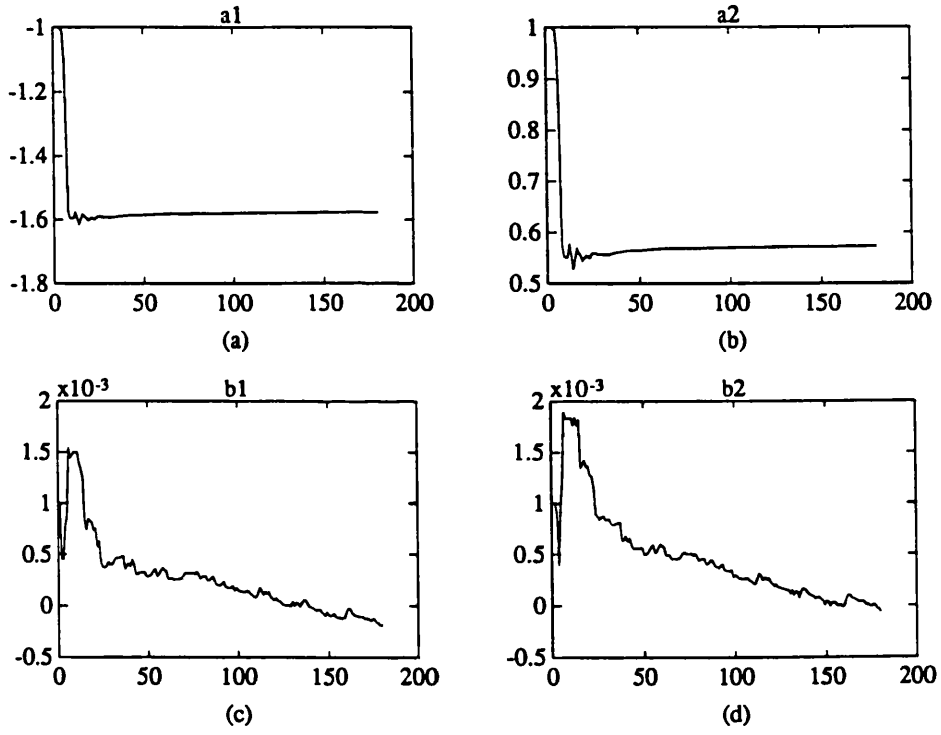



**Fig. 3.2 Approximate RLS results with large initial deviation**

However, as can be seen from Fig. 3.3, if there are no initial deviation in parameter $a_1$ and $a_2$, the approximate method can very well identify the true values of $b_1$ and $b_2$ (0.00045) even if they are provided with wrong initial values (0.001).

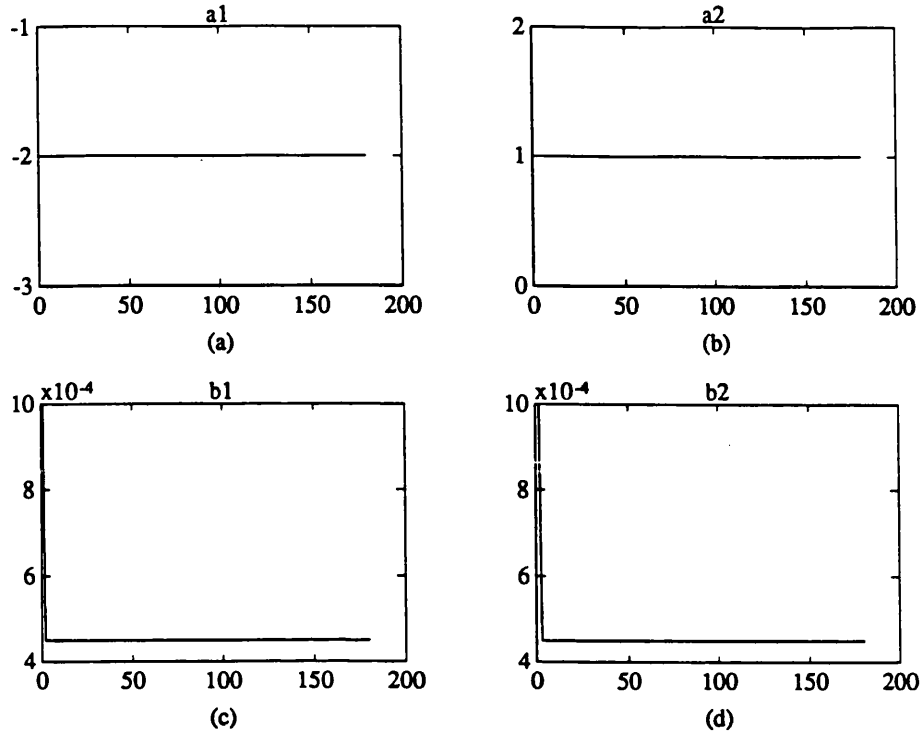



**Fig. 3.3 Approximate RLS results with no initial deviation**

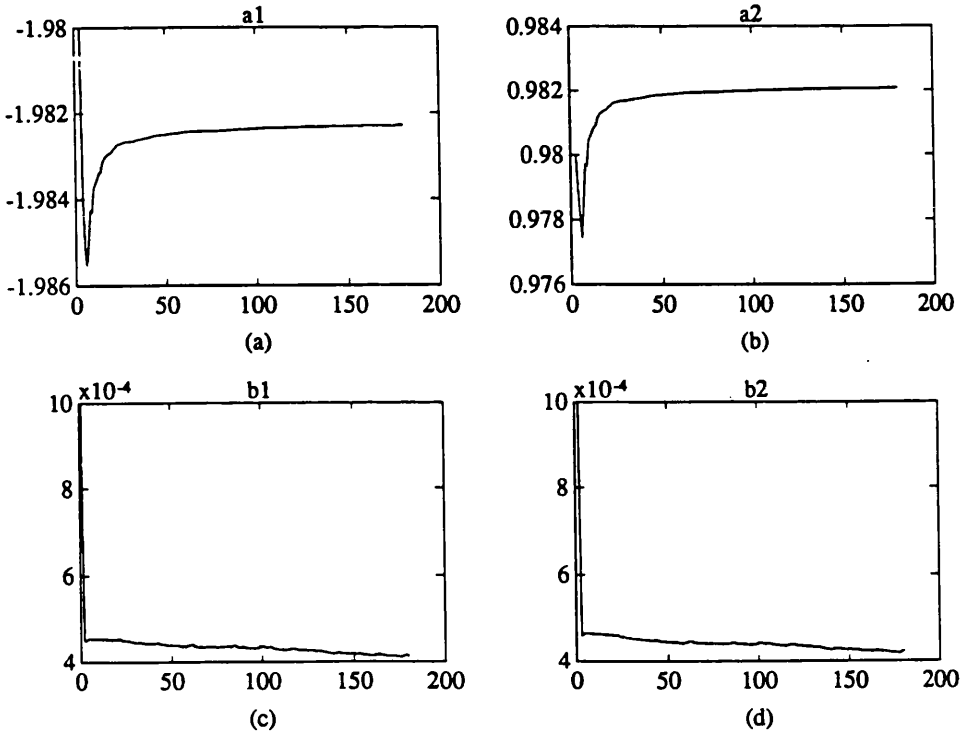



**Fig. 3.4 Approximate RLS results**

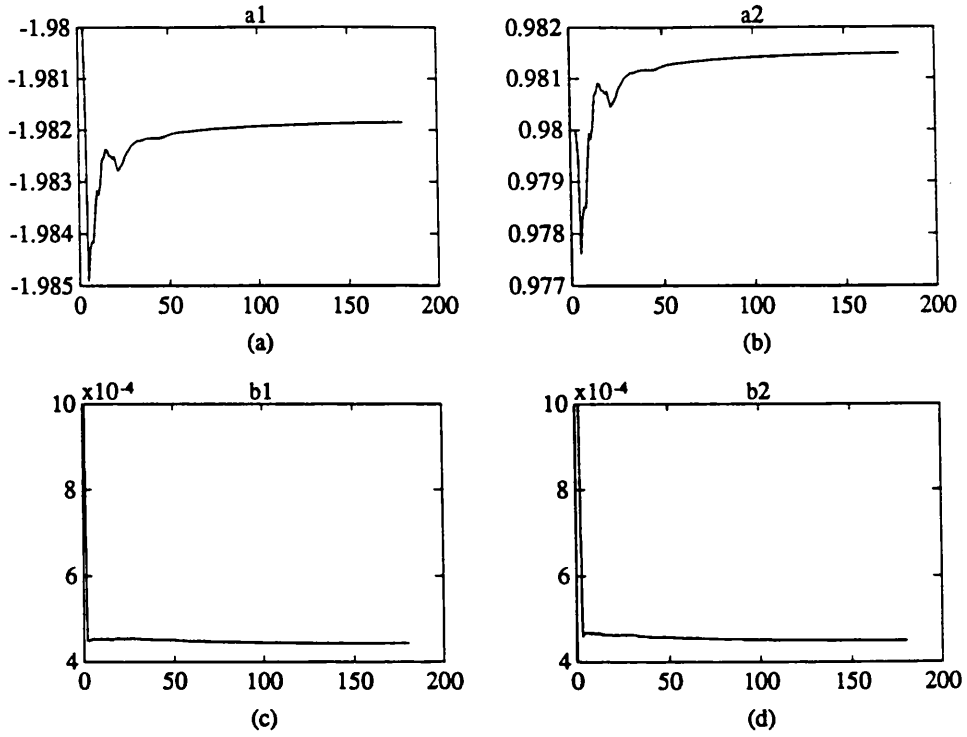



**Fig. 3.5 Approximate RLS results with forgetting factor $\gamma = 1.05$**

In practice, exact parameter values of $a_1$ and $a_2$ are very difficult to be knew a priori. Fig. 3.4 and 3.5 assume that there is small deviation in the parameters $a_1$ and $a_2$. The approximate RLS can not find the true parameter values. As can be seen from Fig. 3.5, the forgetting factor slightly larger than 1 will give better identification results.

From this analysis it can be concluded that the approximate identification approach can only identify the system parameters providing rather correct initial parameter values are given. Needless to say, if there is no sufficient excitation, both schemes can not identify the true values of the system parameters. The persistent excitation problem will be discussed in the next section.

### 3.2.3 Persistent Excitations

In on-line identification, there is the problem of persistent excitation, and in the case of robotic manipulators, this condition can rarely be satisfied. However, this has been shown not to be an obstacle in using self-tuning adaptive control for robotic manipulators. In this sub-section, a less persistent excitation signal is used, that is, $\sin(4t)$. It can be seen, that even if plant is not well excitated, rather correct parameter values can be identified, providing the true initial parameters $a_1$ and $a_2$ are used. For a second order system, $a_1$ and $a_2$ equal approximately to -2 and 1 respectively as will be pointed out in sub-section 3.3.5.

If only one parameter is to be identified, there is no problem of lack of persistent excitation. If there are many parameters to be identified, as is usually the case, persistent excitation is then necessary. Sometimes identifying the system parameters precisely is not necessary. Especially in the application of robotic manipulator control, persistent excitation can not be obtained for a dedicated task with pre-specified trajectory.

Astrom [1989] has fully discussed various input signals. It is understood that random signals and pulse signals are the most persistent ones. The second one is periodic signal (square wave, for example) containing many frequencies other than a specific one as in sinusoid signal, which is less persistent. The last one is step signal. Pulse signal may cause damage to the plant, random signal is frequently used in the system identification.

In the following figures, the initial values are chosen as: $a_1 = -1.98$; $a_2 = 0.98$; $b_1 = b_2 = 0.001$. There are small deviations in parameter $a_1$ and $a_2$, but larger ones in $b_1$ and $b_2$, as true $a_1$=-2, $a_2$=1, and $b_1$=$b_2$=0.00045.

Fig. 3.6 shows the identification results of normal RLS method under non-persistent excitation signal. As shown in above figure, after 100 steps (about 50 steps in Fig. 3.1), the four parameters approach their true values.

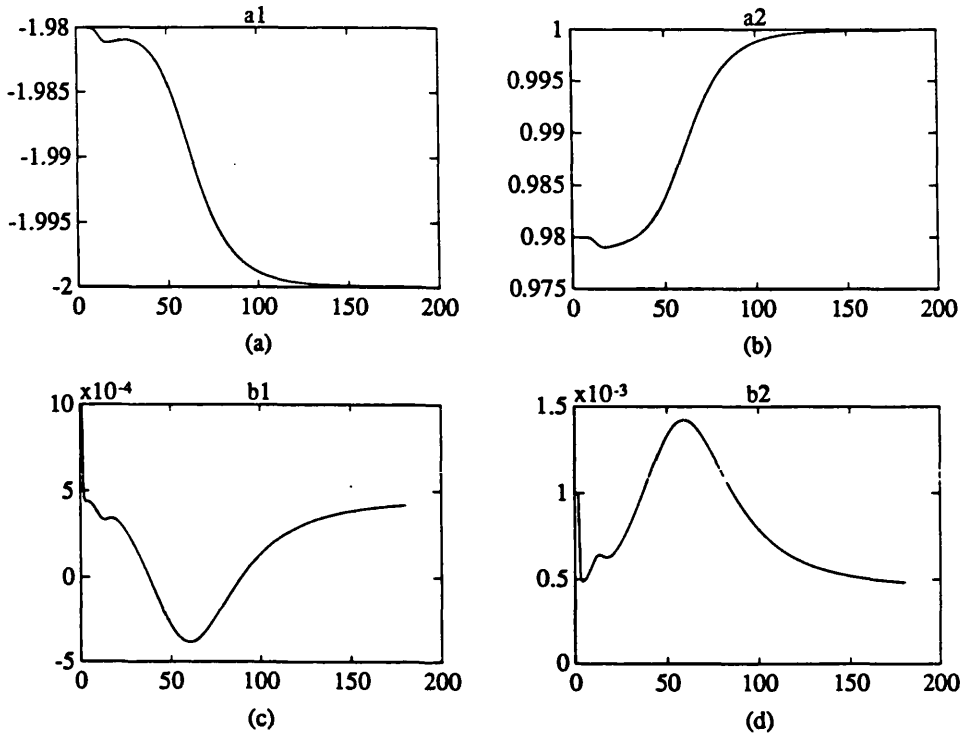



**Fig. 3.6 Normal RLS results**

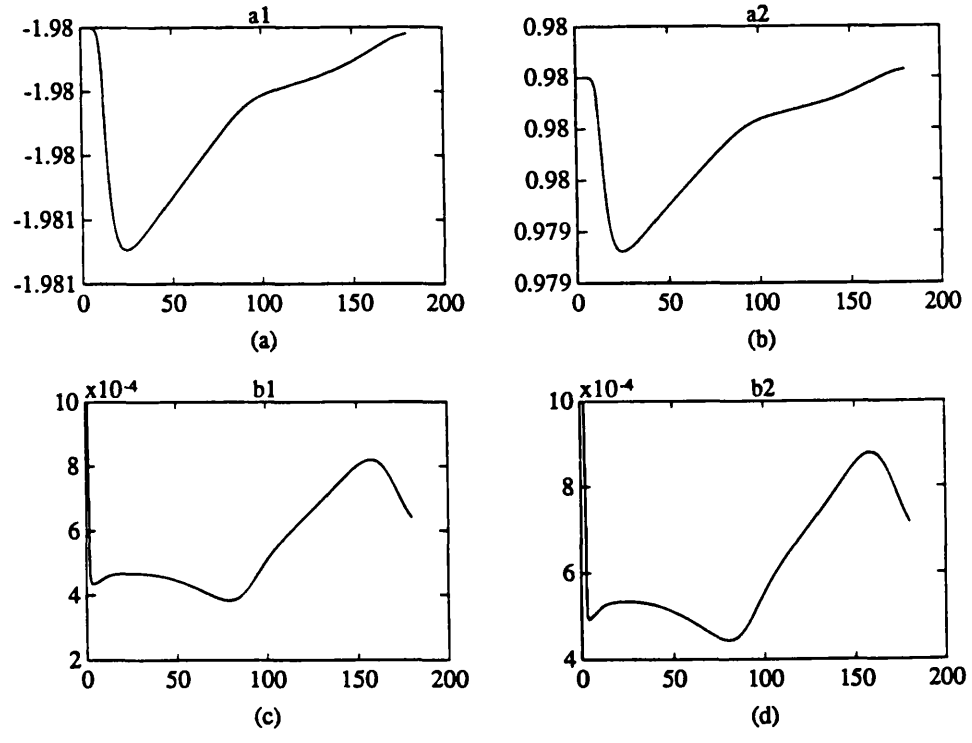



**Fig. 3.7 Approximate RLS results**

However, the approximate method does not show this results, as can be seen from Fig. 3.7. What is more, the four parameters keep changing continually, without tendency to approach constant values.

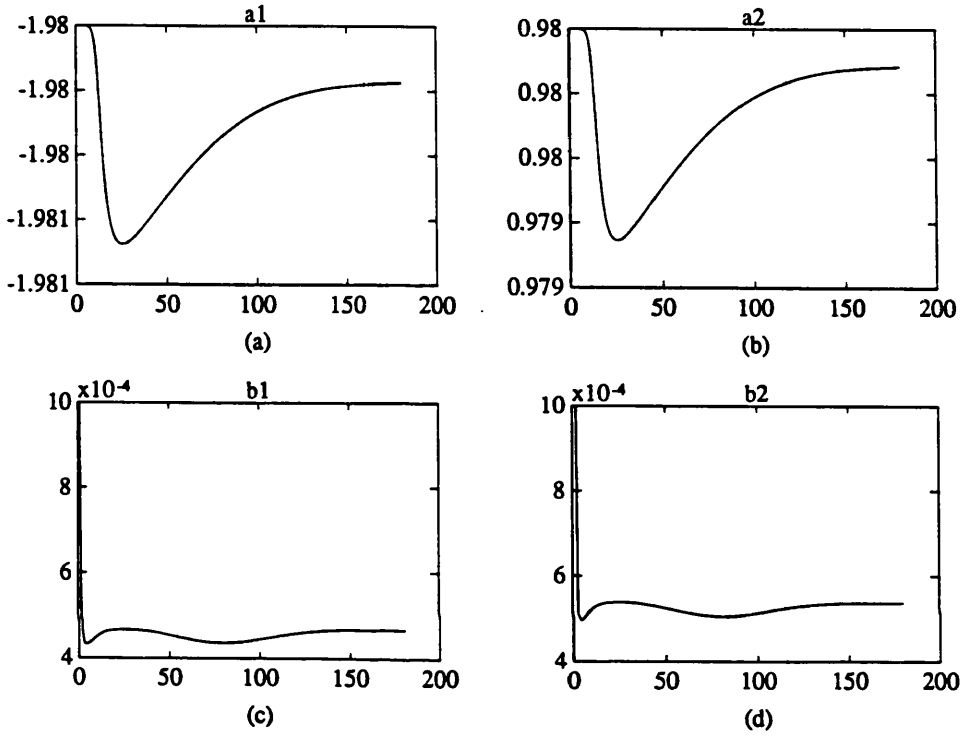



**Fig. 3.8 Approximate RLS results with forgetting factor $\gamma = 1.05$**

By letting the forgetting factor slightly larger than 1, as can see from Fig. 3.8, the four parameters approach constant values in the approximate method.

## 3.3 VARIOUS DIGITAL SELF-TUNING PID

The PID controller was originally developed in the analogue form of continuous time. P stands for proportional control effort with gain $K_p$, I for integral with gain $K_i$ and D for derivative with gain $K_d$. High proportional gain is good for fast response. The integral gain is used to reduce stead-state error and the derivative gain is used to improve whole system stability. The

PID regulator covers the basic requirements of a control system and forms the basis of classical control theory. Because of its simplicity, robustness and cheapness, the PID controller has being widely accepted by industry.

As micro-processors are becoming cheaper and cheaper, digital control using a computer is becoming more and more popular. As a result, much more sophistic controller can be achieved at the micro-processor. Optimum control and adaptive control can only become fact in the last decade although the basic idea was realised as early as 1950s. Digital counterparts for various controller has been imposed.

There are many different forms of digital PID controllers according to the ways of how it is transferred from the analogue form.

### 3.3.1 Simulation Method

Here, some simple concepts about simulation of SISO systems are to be addressed. In the real situation, the response of a dynamic system underwent control input is to be measured. In the case of simulation this requirement can not be met. The dynamic model of real physical plant has to be used to calculate the response.

Runge-Kutta numerical integration is usually used in the simulation to acquire the system response. The Runge-Kutta method is little bit complicated as not to be suitable for simple simulation. Here some simple alternatives are presented and comparison are made. In the approximate method, forward difference is used to represent differential equation, i.e. $\dot{q} = \frac{q(k)-q(k-1)}{T}$, resulting in a difference equation. The response can thus be calculated by solving algebraically the difference equation.

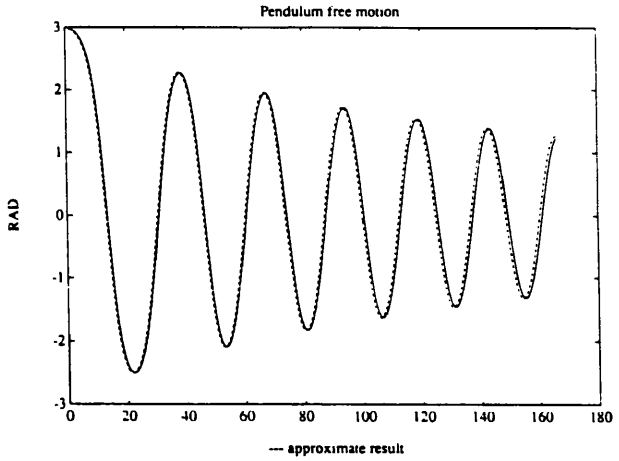



**Fig. 3.9 Comparison between Runge-Kutta and an approximate method**

In Fig. 3.9, the solid line in the left plot shows the Runge-Kutta results, and the dotted line, the approximate result. As can see, two methods give close results. The figures present free movement of a pendulum, shown in Appendix 3.2. The dynamic model is presented by: $\tau = \ddot{q} + 0.25\dot{q} + 100\sin(q)$. The pendulum stays at initial position 3 (RAD). The sample period is chosen as T=0.03, and the smaller T will give more accurate response both for Runge-Kutta and the approximate method. It is suggested to use Runge-Kutta numerical integration for simulation. Only if a control scheme works with this Runge-Kutta integration, approximate integration methods can then be used. Appendix 3.3 lists MODULA-2 program of numerical integration for single input and single output (SISO) systems. Most of the simulation work carried out in this project used Runge-Kutta method.

### 3.3.2 Digital PID Controller

To form a digital PID controller, first it is necessary to analyse the analog PID controller, which has the following format:

$$u = K_p \cdot e + K_i \int_0^t e \cdot dt + K_d \cdot \dot{e} \qquad (3\text{-}2)$$

where $e$ is the error signal input to the PID controller. The transfer function is of the following form:

$$\frac{u}{e} = G_c(s) = K_p + \frac{K_i}{s} + K_d s \tag{3-3}$$

Although there are several methods (Ogada K. [1987], pp330) to transfer the analogue PID controller to digital counterpart, only two are commonly used. One is the approximate transfer:

$$s = \frac{1 - z^{-1}}{T}$$

and the other is the so called bi-linear transfer:

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

According to the above discussion of transferring the analogue PID controller to its digital counterpart, many different digital PID controllers are available because of the different transfer methods being used. Two different digital PID controllers are given here as the result of forward difference and bi-linear transfer:

$$\frac{u(k)}{e(k)} = G_c(z^{-1}) = \frac{s_0 + s_1 z^{-1} + s_2 z^{-2}}{1 - z^{-1}} \tag{3-4}$$

$$\frac{u(k)}{e(k)} = G_c(z^{-1}) = \frac{s_0 + s_1 z^{-1} + s_2 z^{-2}}{1 - z^{-2}} \tag{3-5}$$

where $s_0$, $s_1$ and $s_2$ are digital gains comprised of the analogue gains $K_p$, $K_i$, $K_d$ and sampling period T.

A revised digital PID controller, first appeared in Wang [1991], is acquired based on equation (3-5) by adding an extra parameter $s_3$ in the controller, equation (3-6). The reason for this will be explained in sub-section 3.3.4.

$$\frac{u(k)}{e(k)} = G_c(z^{-1}) = \frac{s_0 + s_1 z^{-1} + s_2 z^{-2}}{1 + s_3 z^{-2}} \tag{3-6}$$

These digital PID controller gains $s_0$, $s_1$, $s_2$ and $s_3$ are chosen using the Z-plane root locus method in this thesis.

Simulation for constant PID controller is carried out and some results and conclusions are as follows (the dynamic model used for the design of PID controller is chosen as $\tau = \ddot{q}$):

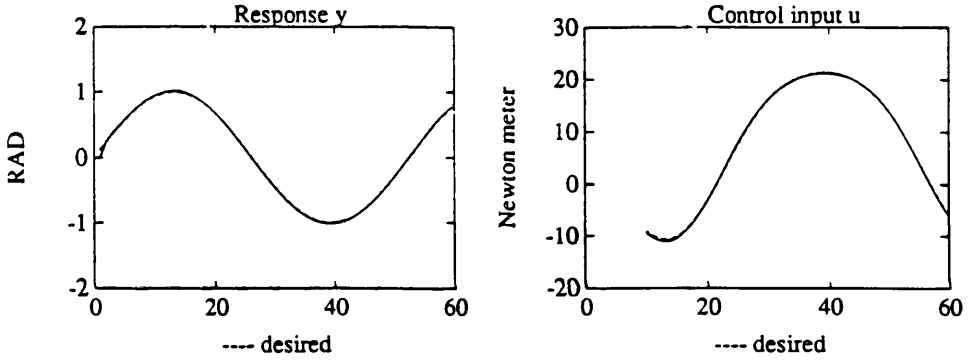



**Fig. 3.10 Constant PID controller without model mismatch**

In Fig. 3.10, there is no model mismatch between the real plant and the model used to design PID controller. As can be seen, the control effort is perfect.

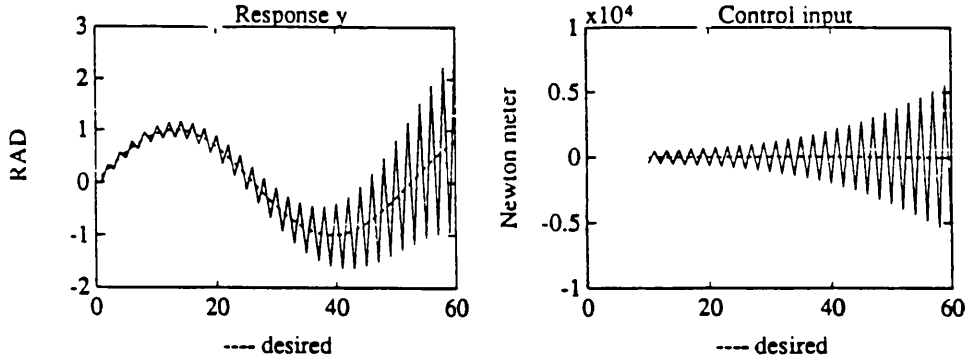



**Fig. 3.11 Constant PID controller with 25% model mismatch**

However, supposing the above model is not accurate by assuming that after some time the dynamic model becomes $\tau = 0.76\ddot{q}$, the results are disastrous, as shown in Fig. 3.11, if the controller gains do not know this change and adapt themselves accordingly. Both control input and response diverge to infinity.

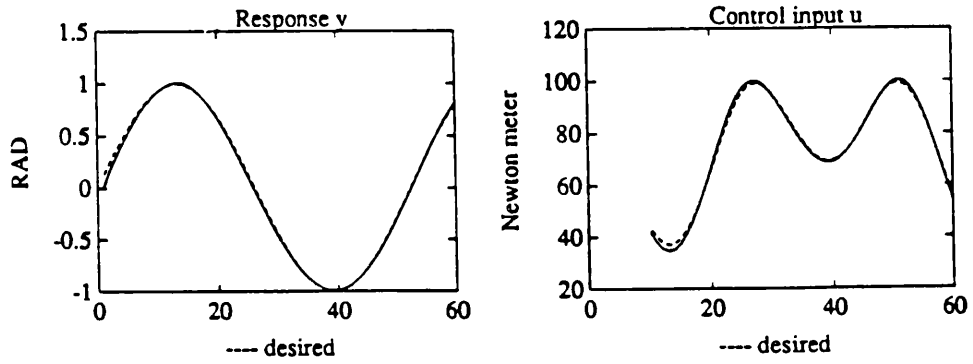



**Fig. 3.12 Constant PID controller without model mismatch but highly non-linear**

In Fig. 3.12, the simulation results are acquired with the real plant having the dynamic behaviour: $\tau = \ddot{q} + 100\cos(q)$ with high non-linearity term $100\cos(q)$, which the controller does not know. As can be seen, the control effect is still very good.

In the figures of this sub-section, the desired response is chosen as $r = sin(4t)$ and sampling period, T=0.03 s.

### 3.3.3 Self-tuning Adaptive Control Strategy

Self-tuning adaptive control is developed as the development of the digital computer, as the controller itself is inherently nonlinear and also on-line identification has to be incorporated in the process. As shown in Figure 3.13, there are two loops, one being the fast feedback loop and another, slow adapt loop.

A reference model is assigned to the system and it is assumed that the system will behave as a linear time-invariant model during the sampling interval T. Within the sampling time period linear control theory can be used to design

the feedback law, which is by pole assignment in this case. The reference model for each arm joint is chosen as a second order difference model which is simple to implement and identify. Expressing the system in discrete form is outlined in the following.

Transfer function for the zero holder (ZoH) is (Chapter 5 of Franklin [1990]):

$$\frac{1-z^{-1}}{s}$$

Then the discrete transfer function of plant is:

$$G_p(z^{-1}) = (1-z^{-1})Z\left\{\frac{G_p(s)}{s}\right\}$$

This will result in a general form (Wang [1990]):

$$\frac{y(k)}{u(k)} = G_p(z^{-1}) = \frac{1+a_1z^{-1}+a_2z^{-2}}{b_1z^{-1}+b_2z^{-2}} \qquad (3\text{-}7)$$

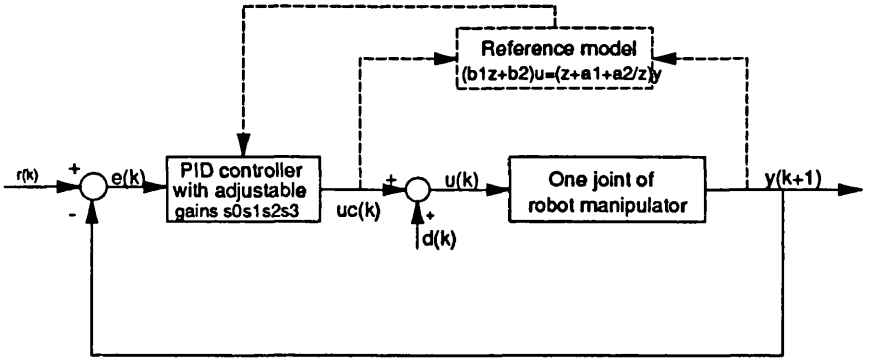



**Fig. 3.13 Self-tuning adaptive control scheme**

This is a general discrete-time transfer function for the second order system, in ARMA form. Where y(k) and u(k) represent the measured (sampled) position (angle) variable and the driving signal (i.e. motor voltage) of each joint respectively. The robotic manipulator dynamics are highly coupled and nonlinear, and so this linear, single input, single output equation will

not be accurate. So the process parameters $a_1$, $a_2$, and $b_1$, $b_2$ will change with time. Thus at each time step the process parameters are continually estimated in order to adapt the PID gains.

Having obtained an estimate of the local joint dynamics it is possible to calculate PID gains, which will affect response. The equation for a particular form of PID control is (refer to equation (3-6)):

$$\begin{aligned} u(k) = &-s_3 u(k-2) + s_0[r(k) - y(k)] \\ &+ s_1[r(k-1) - y(k-1)] + s_2[r(k-2) - y(k-2)] \end{aligned} \tag{3-8}$$

where r(k) is the reference value, e.g., desired joint angle of a pendulum shown in Appendix 3.2.

The control parameters are calculated using the most recently estimated process parameters in such a way as to set the position of the closed loop poles to user definable values within the unit circle, thus largely defining the dynamics of the closed loop response (Wellstead [1979]).

By combining (3-7) and (3-6), the closed-loop transfer function of the whole system will have the denominator of fourth order:

$$T(z^{-1}) = 1 + t_1 \cdot z^{-1} + t_2 \cdot z^{-2} + t_3 \cdot z^{-3} + t_4 z^{-4}$$

Then, an explicit relationship between the process parameters and the controller's parameters can be acquired for the second order system (Broome and Wang [1991]).

$$\begin{aligned} s_0 &= (t_1 - a_1)/b_1 \\ s_3 &= \frac{b_1 b_2 t_3 - b_2^2 t_2 - b_1^2 t_4 + b_2^2 a_2 + b_2^3 s_0}{b_1 b_2 a_1 - b_2^2 - a_2 b_1^2} \end{aligned} \tag{3-9}$$

$$s_2 = (t_4 - a_2 s_3)/b_2$$

$$s_1 = (t_2 - a_2 - b_2 s_0 - s_3)/b_1$$

By properly choosing the parameters $t_1$, $t_2$, $t_3$ and $t_4$ the system will be stable when $T(z^{-1})$ has zeros within the unit circle. One example is that $t_1$= -0.9, $t_2$=$t_3$=$t_4$=0, which gives three poles at zero and one pole at 0.9.

### 3.3.4 An Approximate Self-tuning PID Controllers

Equation (3-7) shows a general discrete transfer function for the second order system. However, the plant can also be presented by a approximate ARMA model using the forward difference equation (Karam [1989]):

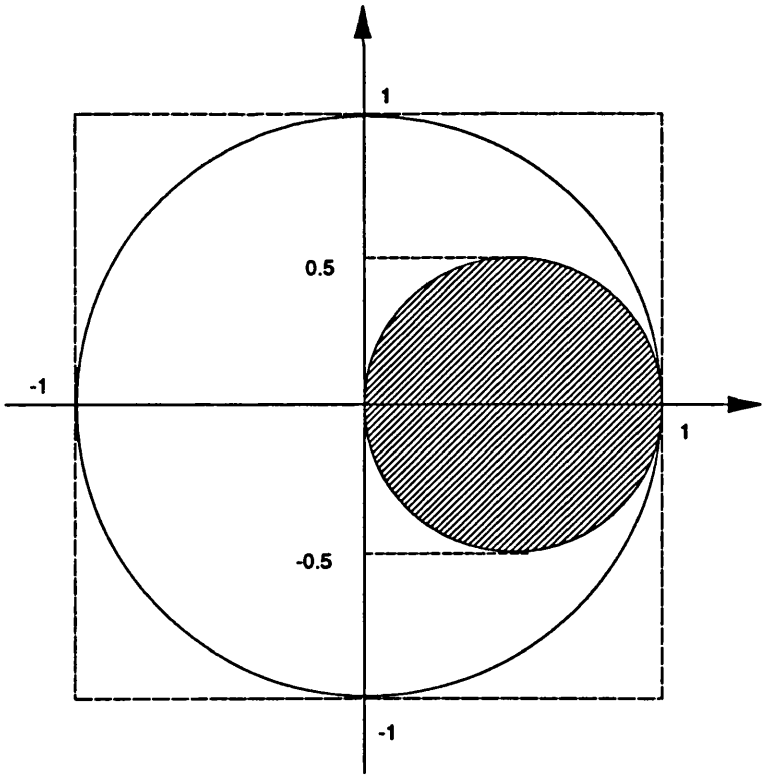



**Fig. 3.14 Circle for poles to lie in**

$$G_p(z^{-1}) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{b z^{-1}} \tag{3-10}$$

Karam [1989] uses this ARMA model and approximate PID controller shown in equation (3-4), written in another form:

$$\begin{aligned} u(k) &= u(k-1) + s_0[r(k) - y(k)] \\ &\quad + s_1[r(k) - y(k-1)] + s_2[r(k) - y(k-2)] \end{aligned} \tag{3-11}$$

Form equation (3-10) and (3-11), the following explicit self-tuning law will result applying the adaptive strategy described in the former sub-section:

$$\begin{aligned} s_0 &= \frac{t_1 - a_1 + 1}{b} \\ s_1 &= \frac{t_2 + a_1 - a_2}{b} \\ s_2 &= \frac{t_3 + a_2}{b} \end{aligned} \tag{3-12}$$

As the forward difference method is used to acquire the PID controller, the poles have to lie in the shaded circle shown in Fig. 3.14 according to the mapping between continuous time and discrete time (Franklin [1990]). Poles 0, 0 and 0.9, as they were chosen in the former sub-section, will give a bad response in the adaptive control law of (3-12). However, if choosing three poles near the right part of the smaller circle in Fig. 3.14, e.g. $z_1$=0.7943, $z_2$=0.8865 and $z_3$=0.93303, the approximate PID controller presented in this sub-section will give good results. The following sub-section presents some simulation results of both self-tuning adaptive PID controller described in this sub-section and the former one.

### 3.3.5 Simulation of Self-tuning PID Control

The dynamic model for simulated self-tuning PID controller is described in Appendix 3.2. The initial PID controller is designed according to the model $\tau = \ddot{q}$. Simulation results and analysis are presented as follows.

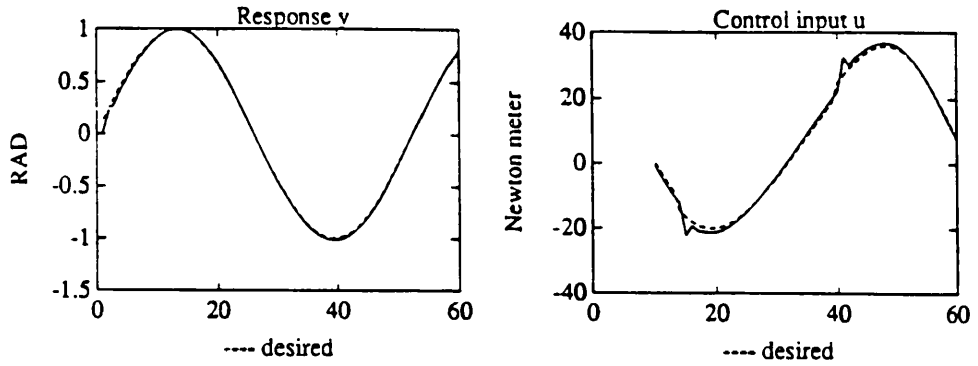



**Fig. 3.15 Self-tuning PID controller without model mismatch**

In Fig. 3.15, the parameters in equation (A3-1) are chosen as: ME=0, C=2, v=5 and g=9.8. There is no model mismatch between the real plant and the model used to design initial PID controller. As can be seen, the control effort is perfect and the control input can follow the desired one very well.

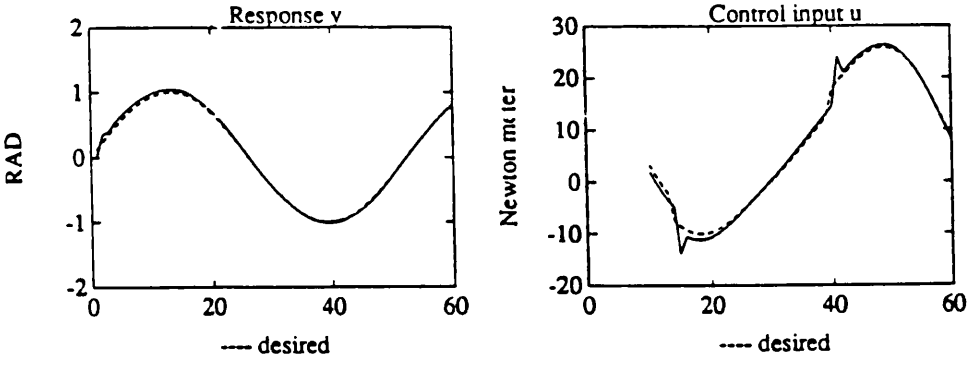



**Fig. 3.16 Self-tuning PID controller with 25% model mismatch**

In Fig. 3.16, the parameters are chosen as: ME=0.25, C=2, v=5 and g=9.8. There is a model mismatch of 25% between the real plant and the model used to

design initial PID controller. As can be seen, the control effort is as perfect as shown in Fig. 3.15 and the control input can follow the desired one very well (compared with Fig. 3.11).

However, in Fig. 3.17, supposing the experimental is carried out in another planet which has much larger gravitation than the Earth, e.g. g=98. Other parameters are: ME=C=v=0. Simulation shows bad response and control input (Fig. 3.17, compare with Fig. 3.12).

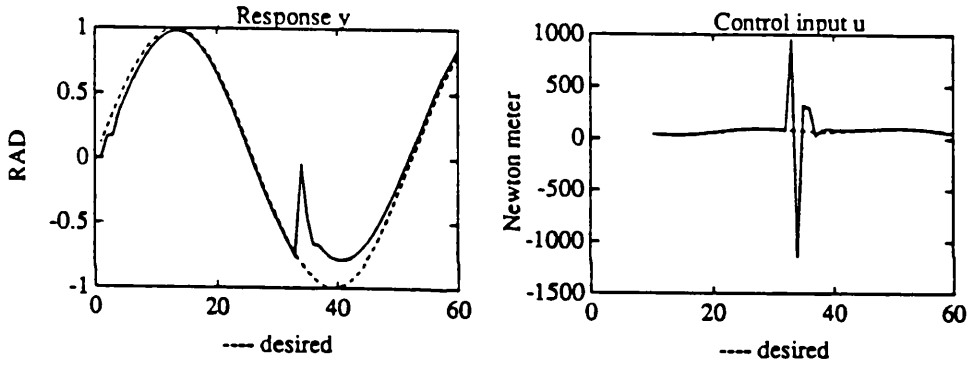



**Fig. 3.17 Self-tuning PID controller without model mismatch but highly non-linear**

In the figures of this sub-section, the desired response is chosen as $r = sin(4t)$ and sampling period, T=0.03 s. The forgetting factor is chosen as: $\lambda = 0.98$.

As can see from the simulation results, self-tuning PID can well solve the problem of model mismatch as the controller itself identifies the process dynamics and adapts to changing situations. However, self-tuning PID controllers behave badly when non-linearities are very high. This is mainly because of the identification process of identification and will be discussed in the next section.

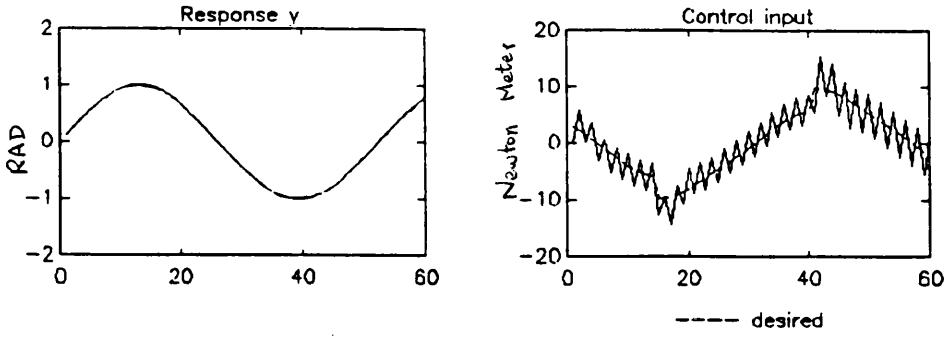



**Fig. 3.18 The approximate ARMA model results**

In Fig. 3.18, Karam and Warwick [1989]'s adaptive control scheme, described in sub-section 3.3.4, is used. However, the three poles are chosen as: 0, 0 and 0.9. The simulation shows that control input chatters (refer to last sub-section).

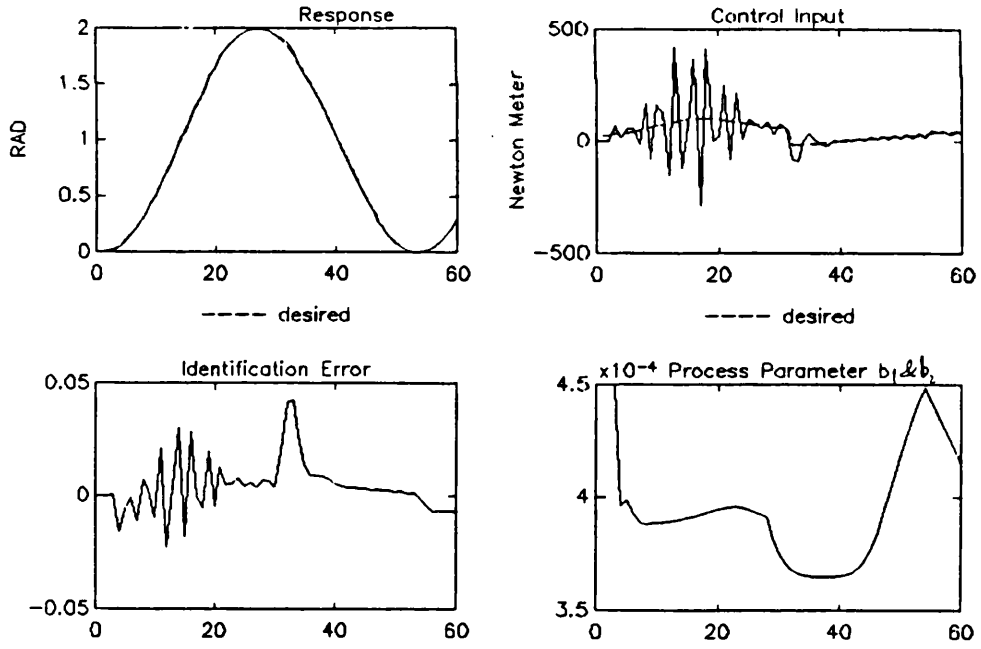



**Fig. 3.19 The new self-tuning PID controller**

Fig. 3.19 shows another simulation result using the new self-tuning PID controller. The desired reference input in chosen as: *1-cos(4t)*.

### 3.3.6 Linear Approaching to Non-linear Plant

The main purpose of this research is to use self-tuning adaptive control for a robot manipulator, which is highly non-linear. The basic idea is to use a linear model to stand for the non-linear model, resulting in fast changing system parameters. In the linear situation, $g$=C=*0* as shown in equation (A3-1), two constant parameters ($m$, $v$) are to be identified, which are contained in the discrete parameters $a_1$, $a_2$ and b in ARMA form as equation (3-10).

Now, consider the non-linear situation, $g \neq 0$ as shown in equation (A3-1). The non-linearity comes from the gravitational term characterised as non-linear term *cos(q)*. The underline assumption in equation (3-10) is to use a linear term $g' \cdot q$ to approach the non-linear term $g \cdot \cos(q)$ resulting in a changing parameter:

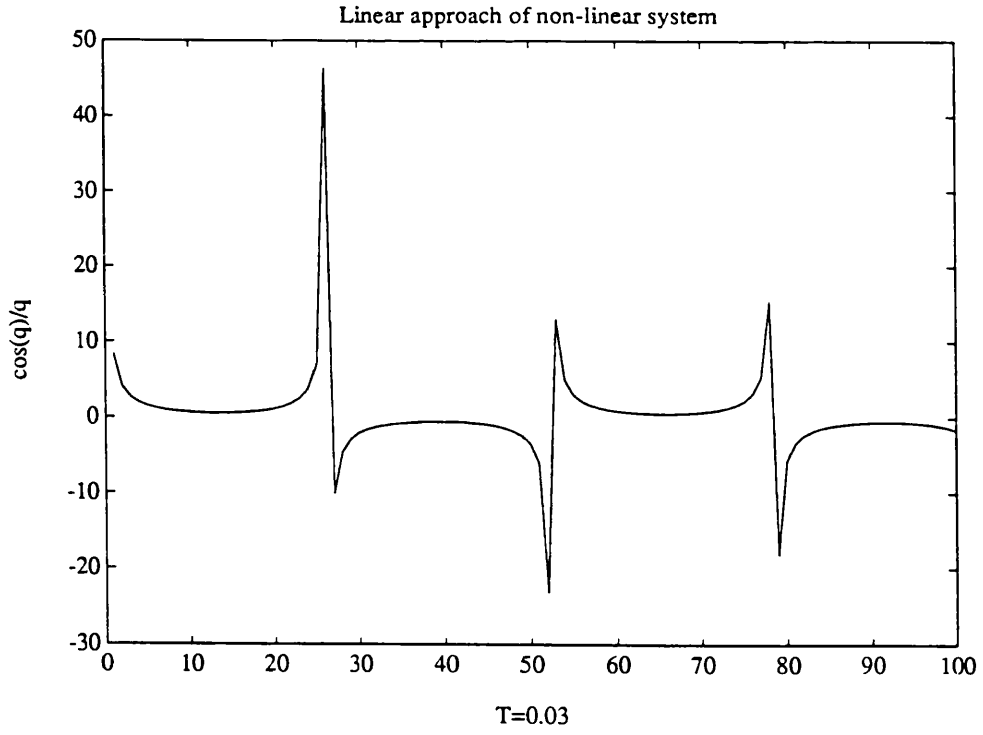



**Fig. 3.20 Parameter in linear approach**

$$g' = \frac{g\cos(q)}{q}$$

$g'$ is drawn in Fig. 3.20 to show its variations in time.

As shown in Fig. 3.20, parameter $g'$ undergos periodic changes. And at some points, when $q$ equals zero, $g'$ goes to infinity. This will cause control problems as has already been seen in the former section (Fig. 3.17). As will be seen in the next chapter, this problem is overcame by letting the forgetting factor $\lambda$ slightly greater than one. What this does is loosely identify parameter $g'$ (the average value), as the accurate value goes to infinity.

## 3.4 NON-LINEAR SYSTEM IDENTIFICATION FOR ROBOT CONTROL

The MATLAB program for non-linear identification and control is listed at Appendix 3.5. Instead of linear approximation as in sub-sections 3.3.3 and 3.3.4, non-linear identifying the non-linear parameter m3 as in Fig. A3.1 of Appendix 3.2, which stands for the payload, is carried out in this section.

### 3.4.1 Description of the Scheme

Fig. 3.21 shows the structure of the control scheme. There are two loops. One is the feedback loop and the other is the feed-forward loop. Both of the feedback controller and the feed-forward loop have changeable parameters, which are updated according to the non-linear system identification.

A Non-linear Auto-Regression Moving Average (NARMA) model is used to stand for the non-linear dynamics of a plant. The parameters associated with the linear terms in the NARMA model are used in the feedback control design. And the parameters associated with the non-linear terms in the NARMA model are used in the feed-forward control design.

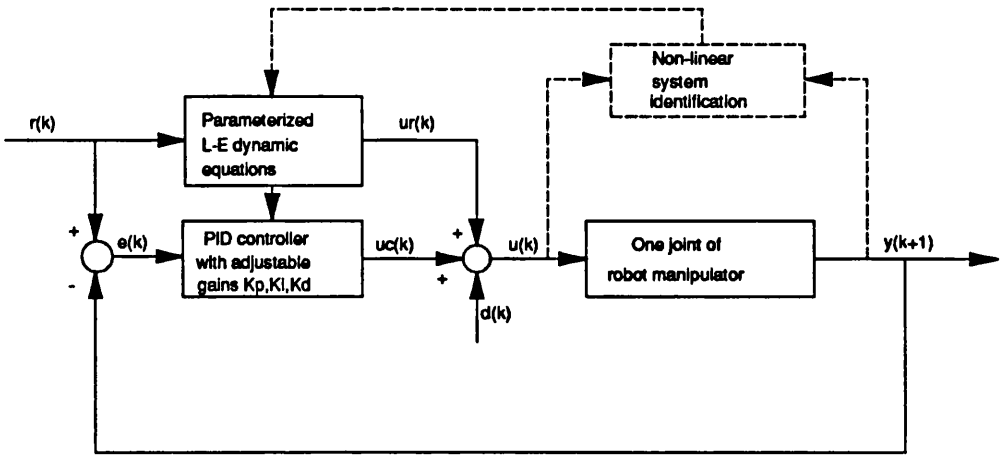



**Fig. 3.21 Non-linear identification and control scheme**

To make things easy, the SISO system shown in Appendix 3.2 is used as an example, with only one non-linear term. The NARMA model will be of the following form, slightly different from equation (3-7) or (3-10):

$$bu(k-1) = y(k) + a_1 y(k-1) + a_2 y(k-2) + a_3 F(k) + d(k) \qquad (3\text{-}13)$$

Where F(k) is the non-linear term, which is the function of the response y(k), i.e. $F(k) = f[y(k)]$. This non-linear term is used for the feed-forward compensation in the feed-forward loop as shown in Fig. 3.21. Parameters $a_1$, $a_2$ and $b$ are used in designing the self-tuning feedback control law. And parameters $b$ and $a_3$ are used in deciding the amount of feed-forward compensation. The following sub-section will present some simulation results of this control scheme.

### 3.4.2 Simulation Results

The non-linear model presented in equation (A3-1) of Appendix 3.2 is further simplified, in simulation, as:

$$\tau = \ddot{q} + 98\dot{q}\cos(q)$$

It can be easily found out, comparing with the NARMA model of equation (3-13), that:

$$a_1 = -2\,,\ a_2 = 1\,,\ b = T*T$$

$$a_3 = 98\,,\ F(k) = \dot{q}\cos(q)$$

This situation can easily be realised as the parameters are linear (Hsia [1977]). The least square identification method can be used. The model can be divided into two part, linear term and non-linear term. The linear part is controlled by a self-tuning adaptive controller and the non-linear term is compensated by the non-linear feed-forward loop.

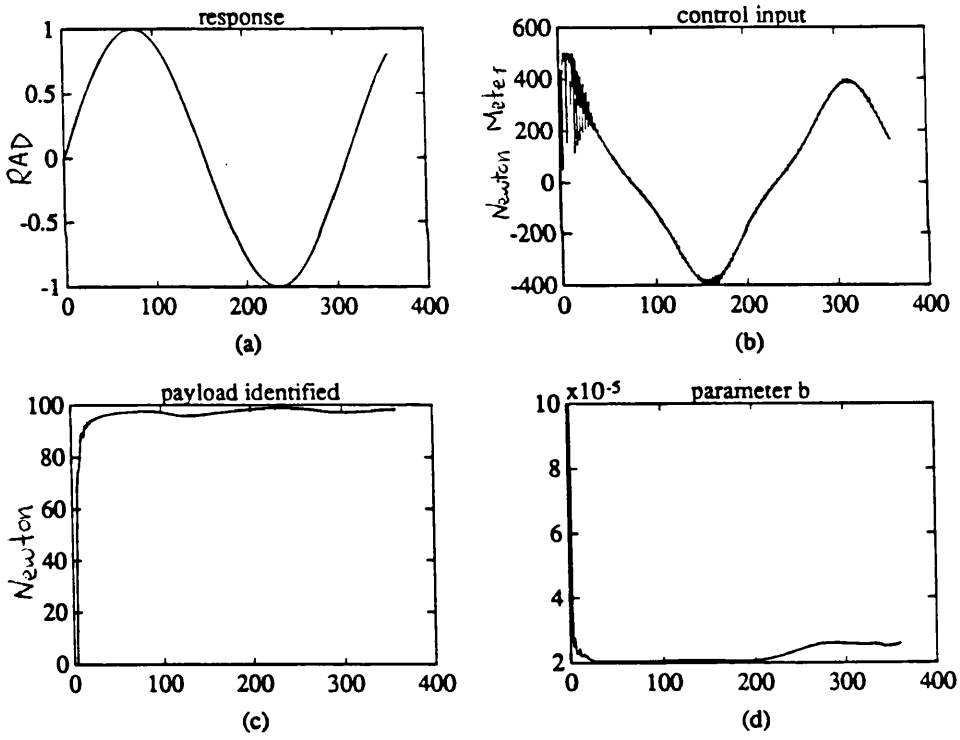



**Fig. 3.22 Result of non-linear identification and control**

Fig. 3.22 shows the simulation results. The desired reference input is chosen as sin(4$t$), sampling period T=0.005 (so, $b$=2.5x$10^{-5}$) and the forgetting factor is chosen as $\lambda = 0.98$. The first small plot in Fig. 3.22 shows output, whose differences from desired one are not distinguishable with naked eyes. Plot

(b) is the control input compared with the desired one. Plot (c) shows the non-linear parameter identified (approaches 98) and plot (d), parameter $b$ identified (approaches $2.5\text{x}10^{-5}$).

## 3.5 DISCUSSION AND CONCLUSIONS

In this chapter, ideal actuators were considered, where the computer DA convertor output can be transformed as torque without any side effects on the dynamic behaviour of the system. If the dynamics of the actuator are to be taken into consideration, the system would be of third order. However, the electrical time constant is usually much smaller than the mechanical time constant and can be ignored (Fu [1987]). This allows us to simplify the third order system to be second order system as used above.

The above discussion is suitable for electrical actuators. As most sub-sea operation manipulators are driven by hydraulic actuators, the whole system including actuators will be of third order (Broome [1984]). Here, an explicit relationship between process parameters and controller gains is given for the third order system as below.

The general discrete transfer function for the third order system would be (Broome [1992]):

$$G_p(z^{-1}) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}$$

and with the general form for the digital PID controller:

$$G_c(z^{-1}) = \frac{s_0 + s_1 z^{-1} + s_2 z^{-2}}{1 + s_3 z^{-1} + s_4 z^{-2}}$$

Five process parameters now need to be identified, and the explicit relation between the process parameters and the controller's gains is very difficult to

acquire, if not impossible. However, ignoring $b_3$ and $s_4$ in the above formulations by letting them be zeros, a recursive explicit relationship can be obtained. The final form acquired is (compare with equation (3-9)):

$$s_3 = \frac{(t_3 - a_3)b_1^2 b_2 - b_2^2(t_2 b_1 - a_2 b_1 - b_2 t_1 + a_1 b_2) - b_1^3 t_4}{a_2 b_1^2 b_2 + b_2^2(b_2 - a_1 b_1) - b_1^3 a_3}$$

$$s_2 = \frac{t_4 - a_3 s_3}{b_2}$$

$$s_1 = \frac{t_2 b_1 - a_2 b_1 - a_1 b_1 s_3 - b_2 t_1 + a_1 b_2 + b_2 s_3}{b_1^2}$$

$$s_0 = \frac{t_1 - a_1 - s_3}{b_1} \qquad (3\text{-}14)$$

Section 3.4 shows a new control scheme for the non-linear system. This scheme can identify the typical parameter of highly non-linear term. Craig [1986] uses a Lyapunov strategy for non-linear parameter identification. The recursive non-linear identification presented in section 3.4 is based on the conventional least square method, which offers a simple solution.

1). This work has shown that selection of the PID controller's gains can be automated according to different payload and manipulation speed. This is more advanced than commercial robots which have no adaptation in the control gains to different working conditions.
2). By neglecting the computation of off-diagonal elements in P(k), of equation (3-1), much time can be saved in the identification without causing much degradation in performance if nearly correct initial parameter values are given.
3). By adding an extra parameter $s_3$, a general explicit relationship between PID controller gains and process parameters has been acquired (3-9).
4). A general second order discrete transfer function is necessary to provide local joint dynamics estimation, especially when friction is high.

The adaptive control strategies have been derived under the assumption that the process parameters are constant but unknown. The practical real process is not always the case. More discussion on this will be presented in the next chapter.

## Appendix 3.1 MATLAB Program for RLS Identification

```
%% MATLAB program for RLS identification
%RLS identification. Created by Q. Wang, April, 1991.;
beta=input('beta=');          %input forgetting factor;
P= [200 0 0 0; 0 200 0 0;  0 0 200 0; 0 0 0 200];
%input initial parameters below;
a1=input('a1='); a2= input('a2=');
b1=input('b1='); b2= input('b2=');
y=0; y1=0; y2=0; u1= 0; u= 0;          %initial values;
I=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
theta=[a1 a2 b1 b2]';              %parameter vector;
rand('normal');
for j=1:180,
 phi=[-y1 -y2 u u1]';
 e=y-phi'*theta;
 beta1=beta+phi'*P*phi;
 gama=P*phi/beta1;
 P=(I-gama*phi')*P/beta;
% Degrading to ARLS if let off-diagonal to be zero
% P(2)= 0; P(3)= 0; P(4)= 0; P(5)= 0;
% P(7)= 0; P(8)= 0; P(9)= 0; P(10)=0;
% P(12)=0; P(13)=0; P(14)=0; P(15)=0;
 theta=theta+gama*e;
 A1(j)=theta(1);  A2(j)=theta(2);
 B1(j)=theta(3);  B2(j)=theta(4);
 u2=u1;  u1= u;
 u=10*sin(j*0.01*4.0);
% u= 20*(rand(1)-0.5);
 U(j)=u;
 y2= y1; y1= y;
 y=2.0*y1-y2+0.00045*(u+u1);
 Y(j)=y;
end;
subplot(221),plot(Y),plot(U),plot(A1),plot(A2);
plot(e),plot(B1); plot(B2)              %present results
%% end
```

## Appendix 3.2 Model of a SISO Non-linear Plant for Simulation

A pendulum, shown in Fig. A.1, is used to test the simulation scheme with a non-linearity. The dynamic equation is simplified to:

$$\tau = (1 - ME) \cdot \ddot{q} + v \cdot \dot{q} + C \cdot sign(\dot{q}) + g \cdot \cos(q) \qquad \text{(A3-1)}$$

where ME --- Model mismatch
(value from 0 to 0.5);
τ --- torque;
C --- Coulomb friction;
v --- viscous friction;
g --- gravitation constant.

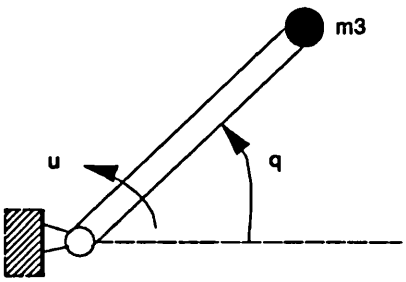



**Fig. A3.1 One DOF Pendulum**

## Appendix 3.3 Program for Runge-Kutta Numerical Integration

```
(*Runge-Kutta numerical integration for SISO system*)
... ...
PROCEDURE Rung_Kut(Old_Vect : TWOREAL; u : REAL; i : INTEGER) : TWOREAL;
VAR
 Y, Yv, An, Bn, Cn, Dn, BTn, DTn : REAL;
 j : INTEGER;
 RESPN : TWOREAL;
BEGIN
 Y:= Old_Vect[1]; Yv:= Old_Vect[2];
 FOR j:= 0 TO INTEGER(T/0.003)-1 DO
  An:= 0.0015*DynaModel(u, Y, Yv, i);
  BTn:= 0.0015*(Yv + 0.5*An);
  Bn:= 0.0015*DynaModel(u, Y+BTn, Yv+An, i);
  Cn:= 0.0015*DynaModel(u, Y+BTn, Yv+Bn, i);
  DTn:= 0.003*(Yv + 0.5*Cn);
  Dn:= 0.0015*DynaModel(u, Y+DTn, Yv+2.0*Cn, i);
  Y:= Y + 0.003*(Yv + 1.0/3.0*(An + Bn + Cn));
  Yv:= Yv+ 1.0/3.0*(An + 2.0*Bn + 2.0*Cn + Dn);
 END;
 RESPN[1]:= Y; RESPN[2]:= Yv;
 RETURN RESPN;
END Rung_Kut;
(*-------------------------------------------------------------*)
```

## Appendix 3.4 MODULA-2 Program for Adaptive SISO Non-linear System Control

```
MODULE Pendu6;
IMPORT IO, FIO, Lib, Respons1, MathLib0, VecLab;
(*-------------------------------------------------------------*)
BEGIN
 Respons1.InputConstants;
 SetInitCondition;
 FOR i:= 0 TO 600 DO
  t:= REAL(i)*T;
  r2:= r1; r1:= r;
  r:= 1.0-MathLib0.cos(4.0*t);
(**)
```

```
    beta1:= beta + P11*Y1*Y1+P22*Y2*Y2+P33*(u)*(u)+P44*(u1)*(u1);
    gama[1]:= -P11*Y1/beta1;
    gama[2]:= -P22*Y2/beta1;
    gama[3]:= P33*(u)/beta1;
    gama[4]:= P44*(u1)/beta1;
    e:= Y + a1*Y1 + a2*Y2 - b1*(u) - b2*(u1);
(**) WriteData;
    a1:= a1 + gama[1]*e;
    a2:= a2 + gama[2]*e;
    b1:= b1 + gama[3]*e;
    b2:= b2 + gama[4]*e;
    P11:= (1.0 + gama[1]*Y1) * P11 / beta;
    P22:= (1.0 + gama[2]*Y2) * P22 / beta;
    P33:= (1.0 - gama[3]*(u)) * P33 / beta;
    P44:= (1.0 - gama[4]*(u1)) * P44 / beta;
(**)
   s0:= ( t1 - a1 ) / b1;
   s3:= (b1*b2*t3-b2*b2*t2-b1*b1*t4+b2*b2*a2+b2*b2*s0*b2);
   s3:= s3/(b1*b2*a1-b1*b1*a2-b2*b2);
   IO.WrReal(s3, 5, 9);  IO.WrLn();
   s1:= ( t2-a2-s0*b2-s3) / b1;
   s2:= ( t4-a2*s3) / b2;
   u2:= u1;          u1:= u;
   u:= -s3*u2 + s0*(r-Y) + s1*(r1-Y1) + s2*(r2-Y2);
   Y3:= Y2; Y2:= Y1;  Y1:= Y;
(*Runge-Kutta method*)
(*  RESP:= Respons1.Rung_Kut(RESP, u, i); Y:= RESP[1]; *)
(*Approximate method*)
   Y:= 2.0*Y1 - Y2 + (T*T)/(1.0+ME)*(u - v/T*2.0*(Y1-Y3)
     - C*VecLab.sign((Y1-Y2)) - g*(MathLib0.sin(Y1) );
(*Desired Torqe*)
   uu:= (1.0+ME)*16.0*(1.0-r) + v*4.0*MathLib0.sin(4.0*t)
     + C*VecLab.sign(MathLib0.sin(4.0*t)) + g*MathLib0.sin(r);
(**)
 END;
END Pendu6.
```

## Appendix 3.5 MATLAB Program for Non-linear Identification

```
%This is to test Non-linear identification
%and control for robotic manipulator 6/8/91
beta= input('forgeting factor='); T= input('sample T=');
P= [200 0, 0 200];
a3= input('payload m3='); b= input('b=');
y= 0; y1= 0; y2= 0; r= 0; r1= 0; r2= 0;
u= 0; uu= u + a3/b*4;
I= [1 0, 0 1]; theta= [a3 b]'; rand('normal');
for j= 1.8/T;
 r2= r1; r1= r; r= sin(4*j*T); vr= 4*cos(4*j*T);
 vel= (y1-y2)/T; y3= vel*cos(y1);
 a3= theta(1); b= theta(2);
 phi= [-y3 uu]';
 e= y-2*y1+y2 - phi'*theta;
 gama= P*phi/(beta+phi'*P*phi);
 theta= theta + gama*e;
 P= (I-gama*phi')*P/beta;
 s0= 2.1/b; s1= -3/b; s2= 1/b;
 u= u + s0*(r-y)+s1(r1-y1)+s2*(r2-y2) + 2*rnad(1)-1;
```

```
 uu= u + a3/b*vr*cos(r);
 if (uu>500), uu= 500;
 elseif (uu<-400), uu= -400; end;
 y2= y1; y1= y;
 y= 2*y1-y2 +T*T*(uu - 98*vel*cos(y1));
 m3(j)= a3/b; B(j)= b; U(j)= uu;
 ref(j)= -16*r + 98*vr*cos(r);
 Y(j)= y;
end;
subplot(221); j= 1:1.8/T;
plot(m3(j)),title('payload identified'),pause;
plot(B(j)), title('parameter b'), pause;
plot(Y(j)), title('response') , pause;
plot(U(j)), title('control input')
```

# CHAPTER 4
# CONTROL AND SIMULATION OF ROBOTIC MANIPULATORS

## 4.1 INTRODUCTION

The robotic manipulator is a highly non-linear, highly coupled multi-variable system, and provides one of the most challenging and active fields of research within the control community. In the former chapter, the design of self-tuning adaptive control, especially self-tuning PID, was discussed in detail. In this chapter, the use of adaptive control for robotic manipulators is discussed. And a new simulation scheme for adaptive control of robotic manipulators is also developed and presented.

Of course there always exists a trade-off between cost and robot performance. The existing robotic manipulators using fixed gains P(I)D controller, with the gains being adjusted to give critical damping with maximum payload and maximum moving speed. This kind of control scheme has the advantage of simplicity and low cost. However, it has no adaptation to changing working conditions (e.g. changing payload and speed). It only works at the reduced speed (Fu [1987]).

Current work at UCL to improve sub-sea inspection operation has involved extensive research using computer simulation to implement adaptive control strategy in place of the constant P(I)D control scheme (Broome [1991]). Obviously, there are many very sophisticated control schemes in the literature. However, they are either too expensive in contrast with the existing computer power, or too complicated.

It is not obvious that robotic manipulators, which have been designed, assembled and tested for surface operations, will have satisfactory performance when used in operating media other than air. This is especially true for underwater operations where added mass and hydrodynamic drags will affect the dynamic response of the arm. Good arm dynamic response and stability will in general require auto-tuning of the control system gains, and so this is an ideal application for using an adaptive control strategy.

Computer simulation has long been recognized as a powerful way of designing adaptive controllers for the robot manipulator. What is the difference between simulation and real-time control? To be simple, in simulation, a model of the system, which is usually a mathematical equation (s), must be used to calculate the response of the system. While in real-time control this work can be accomplished by measuring the real system response.

As will be indicated in this chapter there are two ways of doing simulation, one is the forward method where the standard way is to use the Runge-Kutta numerical integration method to calculate the system response. Another one is called backward simulation scheme, which was developed during the research of this project.

Backward simulation can mainly find use in nonlinear control (e.g. robotic manipulator) and it is first concluded here that the Newton-Euler dynamics equations are equally well qualified as Lagrange-Euler robotic equations in simulating dynamics control of the robotic manipulators (Wang [1991]). In the linear case, both forward (normal) and backward simulation are identical.

In the forward method, the dynamic equations are used in the numerical integration, but they are used in the numerical differentiation in the backward method. As will be indicated in the later of this chapter that backward simulation is even safer than the forward (normal) method. There are two reasons: one is that numerical integration is essentially a smoothing process but numerical differentiation is more affected by the inaccuracies of the function values (Kreyszig [1979]). Another reason is that in backward simulation it is necessary to reduce the sampling period T to accurately identify process parameters so as to calculate the right response, especially in the situation of highly nonlinear and highly coupled dynamics system like a robotic manipulator. However, there are many cases where accurate identification of the system is not essential, as has already discussed in sub-section 3.3.6 of Chapter 3.

## 4.2 CONVENTIONAL CONTROL SCHEME

Most of the industrial robotic manipulators are controlled by simple P(I)D controller with constant gains. All the existing control schemes are based on the joint actuation, with each joint being controlled separately. The most obvious advantage is its simplicity and cheapness to implement. The following is an analysis of the control scheme of the common robotic manipulator, PUMA560, as an example.

As shown in Fig. 4.1, the arm controller consists of a DEC LSI-11/73 computer and six 6503 microprocessors in a hierarchical configuration, each with a joint position (angular) encoder, a digital-to-analog converter (DAC), and a current amplifier. The LSI-11/73, which is at the top of the system hierarchy, functions as a supervisory computer, which has two main functions: (1) on-line user interaction and sub-task scheduling from the user's VAL-2 commands, and (2) sub-task co-ordination with the six 6503 microprocessors to carry out the command. Various functions, which reside in the EPROM memory of the LSI-11/73 computer, include:

1. Coordinate systems transformations (e.g., from world to joint coordinates or vice versa).
2. Joint-interpolated trajectory planning; this involves sending increment location updates corresponding to each set point to each joint every 28 ms.
3. Acknowledging from the 6503 microprocessors that each axis of motion has completed its required increment motion.
4. Looking ahead two instructions to perform continuous path interpolation if the robot is in a continues path mode.

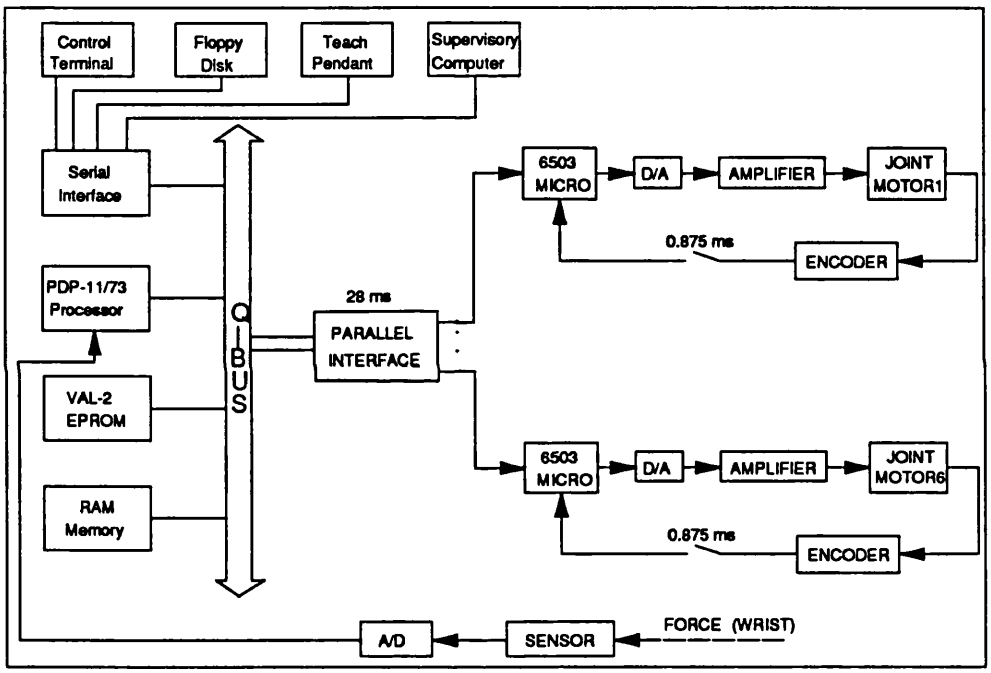



**Fig. 4.1 PUMA560 robotic arm servo control architecture**

Each joint is treated as a simple servomechanism with a 6503 microprocessor which is an integral part of the joint controller. It communicates with the LSI-11/73 computer through an interface board which functions as a demultiplexer that routes trajectory set points information to each joint controller. The parallel interface board transmits the data to and from the Q-bus of the LSI-11/73 (see Fig. 4.1). The microprocessor computes the joint error signal and sends it to the analog servo board which has a current feedback designed for each joint motor.

As shown in the figure, there are two servo loops for each joint control. The outer-loop provides position error information and is updated by the 6503 microprocessor about every 0.875 ms. The inner loop consists of analog devices and a compensator with derivative feedback to dampen the velocity variable. Both servo loops gains are constant and tuned to perform as a "critically damped joint system" at a speed determined by the VAL-2 program.

Each of the six 6503 micro-processors associated with each joint has the following functions:

1. Every 28 ms, receive and acknowledge trajectory set points from the LSI-11/73 computer and perform interpolation between the current joint value and the desired joint value.
2. Every 0.875 ms, read the register value which stores the increment values from the encoder mounted at each axis of rotation.
3. Output the signal to drive the joint via a D/A converter.
4. Form the output signal from the positional error and its derivative.

It can be seen that the PUMA560 robot control scheme is basically a proportional plus integral plus derivative control method (PID controller). One of the main disadvantages of this control scheme is that the feedback gains are constants pre-specified. It does not have the capability of updating the feedback gains under varying payloads. Since an industrial robot is a highly nonlinear system, the inertial loading, the coupling between joints and the gravity effects are all either position-dependant or position- and velocity-dependant terms. Furthermore, at high speeds the inertial loading term can change dramatically. Thus, the above control scheme using constant feedback gains to control a nonlinear system does not perform well under varying speeds and payloads. As a result, the PUMA arm moves with noticeable vibration at reduced speeds.

## 4.3 DYNAMICS AND TRAJECTORY OF A MANIPULATOR

Chapter 2 gives an approximate dynamic model for the PUMA type robotic manipulators. Here a two degree of freedom robotic manipulator, which represents the second and third joint of the PUMA560, is used in the simulation for simplicity. As discussed in Mon [1988], the first joint has less coupling effect to the rest of the joints and the arm, which is comprised of the first three joints of the PUMA560 manipulator and represents the most dynamics property of the whole manipulator. So, the second and third link are chosen for the dynamic analysis. The choice is not less general but the coupled and highly non-linear dynamics has already been taken into account. This kind of simulation can be found largely in the literature. A real task is used requiring the arm to move in a vertical line.

### 4.3.1 Dynamic Model of 2 DOF Manipulator

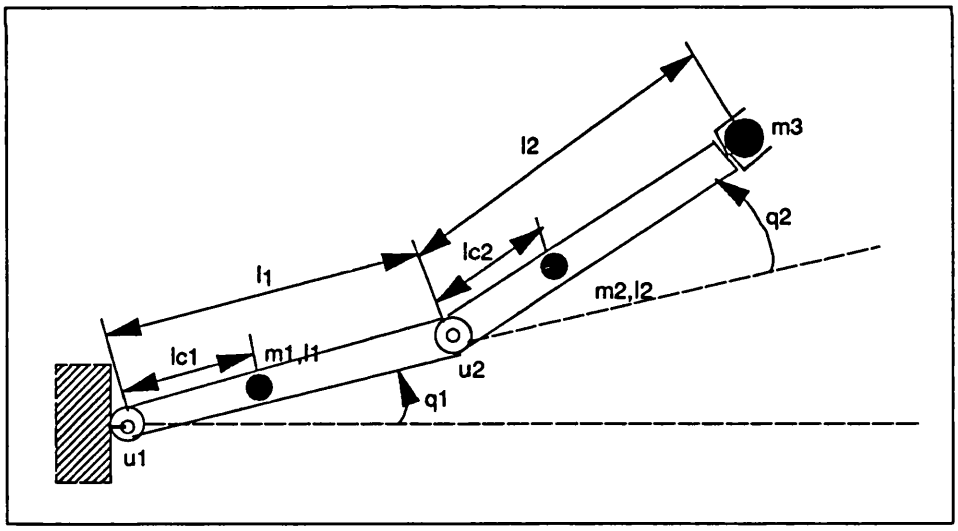



**Fig. 4.2 A 2 DOF robot manipulator with payload**

In Fig. 4.2, $l_{c1}$ and $l_{c2}$ are the mass centre distances, $I_1$ and $I_2$ are the inertias of link 1 and link 2 around their mass centres respectively and $m_3$ is the payload. The Newton-Euler equation for this 2 DOF robot manipulator is (Asada [1986]):

$$\begin{aligned}\tau_1 &= \tau_2 + \vec{r}_{0,c1} \times m_1 \vec{v}_{c1} + \vec{r}_{0,1} \times m_2 \vec{v}_{c2} \\ &\quad -\vec{r}_{0,c1} \times m_1 \vec{g} - \vec{r}_{0,1} \times m_2 \vec{g} + I_1 \dot{\omega}_1 + F_1 \\ \tau_2 &= \vec{r}_{1,c2} \times m_2 \vec{v}_{c2} - \vec{r}_{1,c2} \times m_2 \vec{g} + I_2 \dot{\omega}_2 + F_2\end{aligned} \tag{4-1}$$

The above equations can be written in Lagrange-Euler form:

$$\begin{aligned}\tau_1 &= H_{11}\ddot{q}_1 + H_{12}\ddot{q}_2 - h\dot{q}_2^2 - 2h\dot{q}_1\dot{q}_2 + G_1 + F_1 \\ \tau_2 &= H_{22}\ddot{q}_2 + H_{12}\ddot{q}_1 + h\dot{q}_1^2 + G_2 + F_2\end{aligned} \tag{4-2}$$

$F_1$ and $F_2$ include both Coulomb and viscous friction for joint 1 and joint 2 respectively and they are stabilizing factors by dissipating energy. Where:

$$\begin{aligned}F_1 &= 5.0\dot{q}_1 + 2.0 sign(\dot{q}_1) \\ F_2 &= 3.5\dot{q}_2 + 1.2 sign(\dot{q}_2)\end{aligned} \tag{4-3}$$

$$\begin{aligned}H_{11} &= m_1 l_{c1}^2 + I_1 + m_2(l_1^2 + l_{c2}^2 + 2l_{c1}l_{c2}\cos(q_2)) + I_2 \\ H_{12} &= m_2 l_{\ 1} l_{c2}\cos(q_2) + m_2 l_{c2}^2 + I_2 \\ H_{22} &= m_2 l_{c2}^2 + I_2 \\ h &= m_2 l_{\ 1} l_{c2}\sin(q_2) \\ G_2 &= m_2 l_{c2}\vec{g}\cos(q_1 + q_2) \\ G_1 &= m_1 l_{c1}\vec{g}\cos(q_1) + m_2\vec{g}(l_{c2}\cos(q_1 + q_2) + l_1\cos(q_1)) \\ F_2 &= 2.5\dot{q}_2 + sign(\dot{q}_2) \\ F_1 &= 5\dot{q}_1 + 2sign(\dot{q}_1)\end{aligned} \tag{4-4}$$

Suppose $m_1$=15.91 (kg), $m_2$=6.83 (kg), $l_1$=$l_2$=0.432 (m), $l_{c1}$0.068 (m), $l_{c2}$=0.143 (m), $I_1$=0.539 and $I_2$=0.192 (kgm$^2$), these represent the second and the third joint of the PUMA560 industrial robot manipulator (Seriji [1989]), then, according to equation (4-4):

$H_{11}$= 2.219+0.844cos($q_2$)
$H_{12}$= 0.332+0.422cos($q_2$)
$H_{22}$= 0.332

$$h = 0.422\sin(q_2) \tag{4-5}$$
$$G_1 = 39.518\cos(q_1)+9.572\cos(q_1+q_2)$$
$$G_2 = 9.572\cos(q_1+q_2)$$

Supposing a payload m3=3 kg (it is unknown to the controller) is applied at the end-effector, then the above equations become:

$$H_{11}= 3.308+1.962\cos(q_2)$$
$$H_{12}= 0.861+0.981\cos(q_2)$$
$$H_{22}= 0.860$$
$$h = 0.981\sin(q_2) \tag{4-6}$$
$$G_1 = 52.218\cos(q_1)+22.253\cos(q_1+q_2)$$
$$G_2 = 22.253\cos(q_1+q_2)$$

Forth order Runge-Kutta (Kreyszis [1979], pp.800-803) numerical integration was used to calculated the responses, whose MODULA-2 program is listed in Appendix 4.4. The real trajectory was calculated according to the joint responses supposing there are joint position sensors (encoders) presented at each joint.

### 4.3.2 Real Trajectory Description

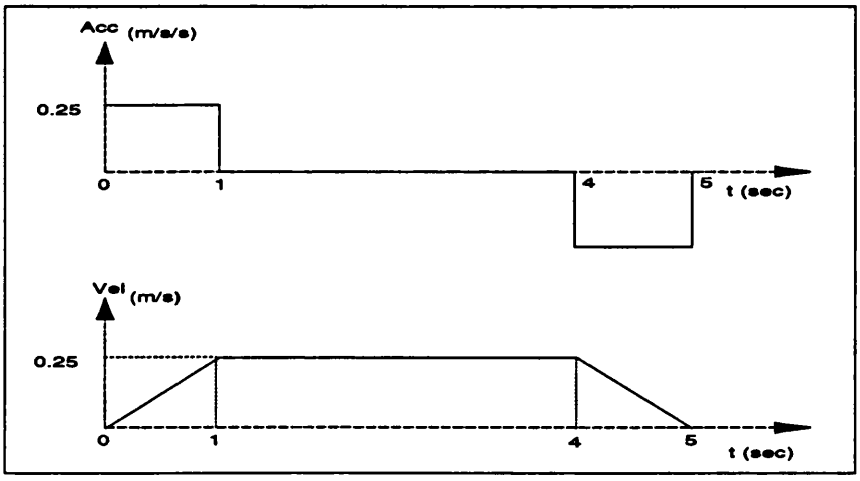



**Fig. 4.3 Velocity/acceleration profile**

The velocity and acceleration profile in Y direction is as shown in Fig. 4.3. These profiles will result in a displacement profile as in Fig. 4.4.

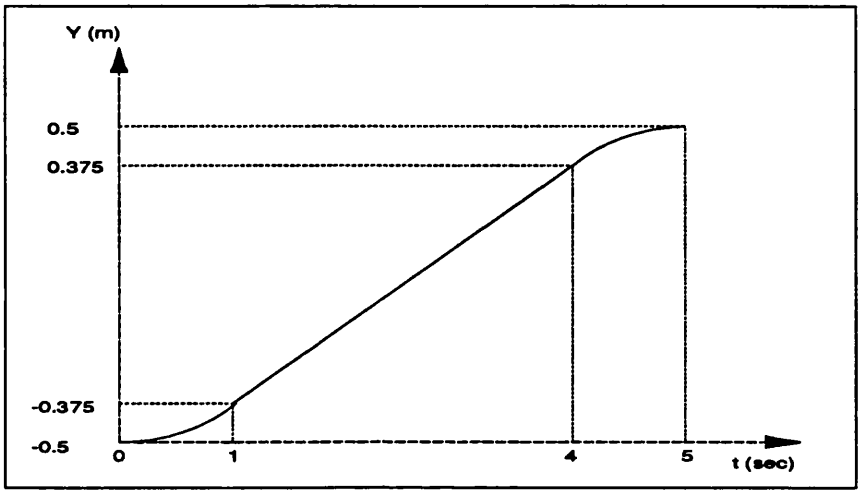



**Fig. 4.4 Displacement in Y direction profile**

The real task is for the manipulator to move along a vertical line x=0.5, from point $P_1(0.5, -0.5)$ to point $P_2(0.5, 0.5)$, in 5 seconds. The arm configuration is chosen as a convex as shown in Fig. 4.5.

The forward kinematics is straight forward:

$$Px = 0.432\cos(q_1) + 0.432\cos(q_1+q_2)$$
$$Py = 0.432\sin(q_1) + 0.432\sin(q_1+q_2) \qquad (4\text{-}8)$$

Where Px and Py are the position in X and Y directions of the end-effector of the manipulator. The inverse kinematics can be solved combining geometry and analysis:

$$r_2 = -\pi + a$$
$$r_1 = b - r_2/2 \qquad (4\text{-}9)$$

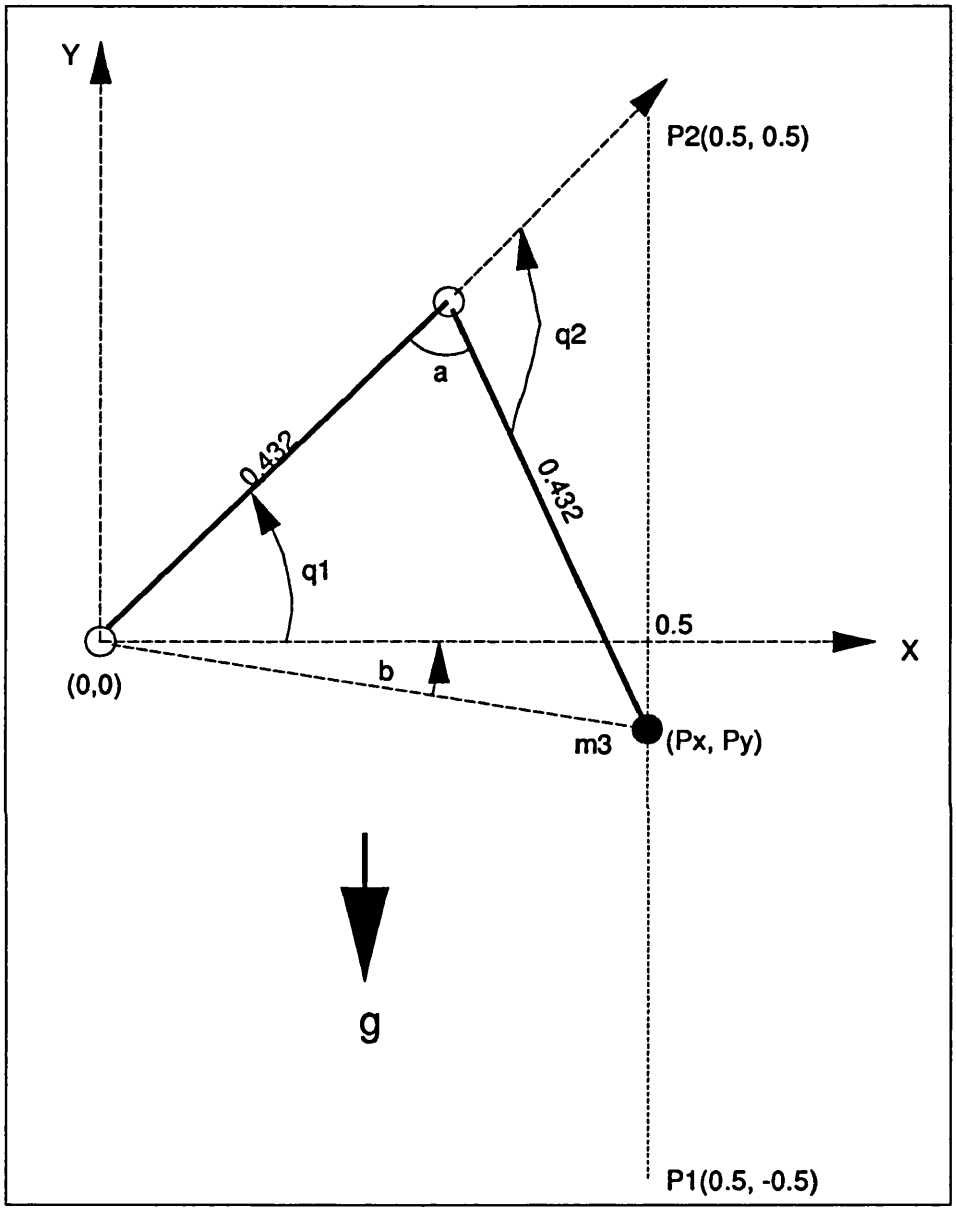



**Fig. 4.5 Manipulator trajectory**

Where $r_1$ and $r_2$ can be thought as the desired reference joint angles for joint 1 and 2 respectively. And *a* and *b* are two angles shown in Fig. 4.5, and their values are:

$$a = \cos^{-1}\left(\frac{2\times 0.432^2 - Px^2 - Py^2}{2\times 0.432^2}\right)$$

$$b = \mathrm{tg}^{-1}\left(\frac{Py}{Px}\right)$$

The desired joint displacements can be calculated either on-line within each sampling period or off-line using the above inverse kinematics equations.

Payload $m_3$ is applied to the arm near the middle of the manipulation, i.e. when the arm moves through the point (0.5,-0.1) it picks up a weight of 3 kilograms.

## 4.4 FORWARD SIMULATION SCHEME

In this section, the self-tuning PID controller described in Chapter 3 is used for a two DOF robotic manipulator described in section 4.3. Each joint is controlled individually.

### 4.4.1 General Description

Fig. 4.6 is the simplified structure of Fig. 3.13 in Chapter 3 to show the simulation structure. In simulation, two models are required. One is the control model, the other is the real dynamic model for calculating the output y(k+1), as shown in Fig. 4.6, where C stands for the controller. This real model is usually the rewritten Lagrange-Euler equations --- the forward dynamics. Some kind of numerical integrations, usually the Runge-Kutta method, are then used to calculate the system response.

Parameters are chosen as: sampling period T=0.005 (s), $t_1$=-0.9, $t_2$=$t_3$=$t_4$=0.0. Initial $u_1$ and $u_2$ are set to zero, and initial $a_1$=-2.0, $a_2$=1.0 and $b_1$=$b_2$=0.0000125. The manipulator is requested to move from point P1(0.5, -0.5) to point P2(0.5, 0.5) in 5 seconds and initially rests at point P1.

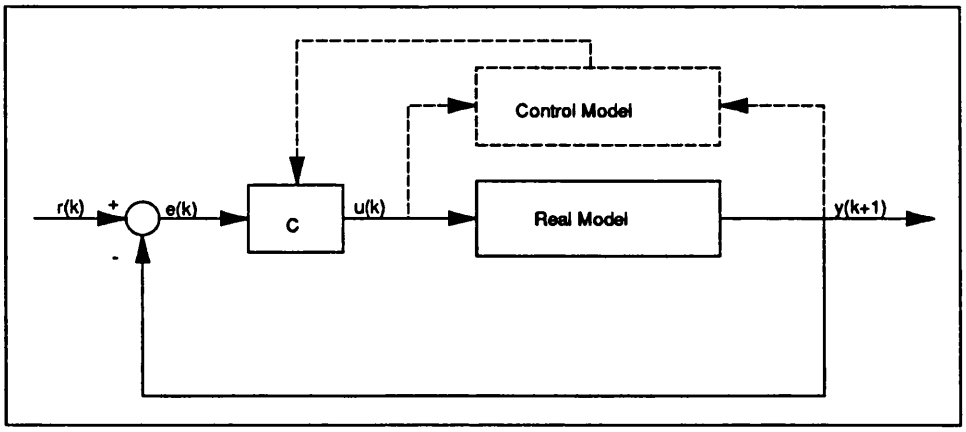



**Fig. 4.6 Forward simulation illustration**

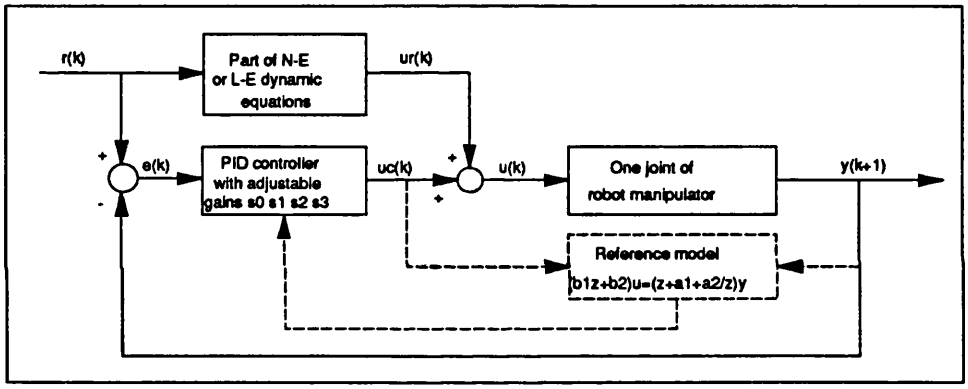



**Fig. 4.7 Compensated forward simulation illustration**

### 4.4.2 Simulation Results and Analysis

In Fig. 4.8, the forgetting factor is chosen as $\lambda$=0.98. As can be seen from the position trajectory in the above figure, the tracking is very good. However, this tracking ability is obtained at the price of high chattering control inputs. The same situation is shared by the sliding mode control. Slotine [1991] gives an example of how the sliding mode control is used to get rid of the chattering control input problem. However, the system given is a single input and single output (SISO) system. As shown in Fig. 4.8, as the robotic manipulator is highly nonlinear and highly coupled, the control inputs show large variance (chattering), especially at the initial stage.

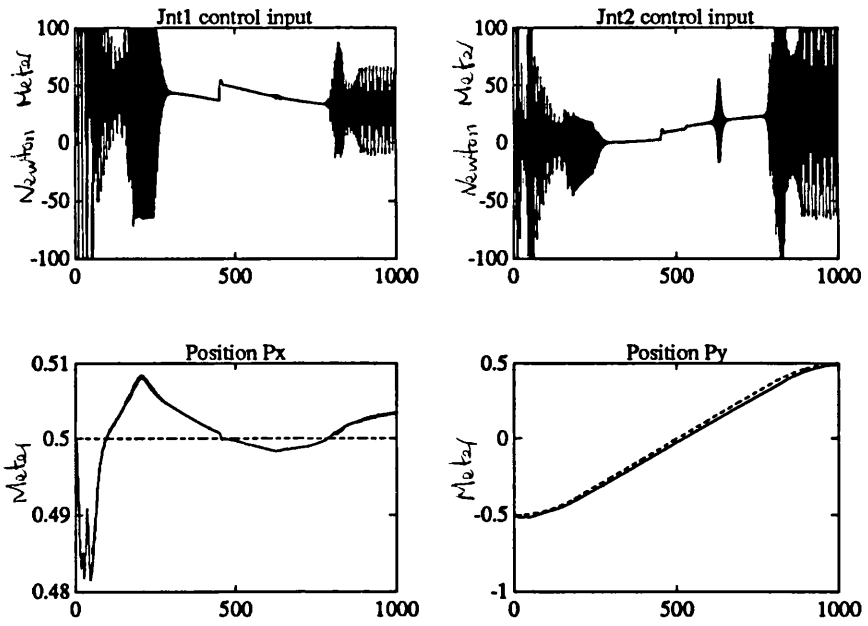



**Fig. 4.8 Control input for joint 1, joint 2 and position trajectory in X, Y directions compared with desired ones (dashed lines)**

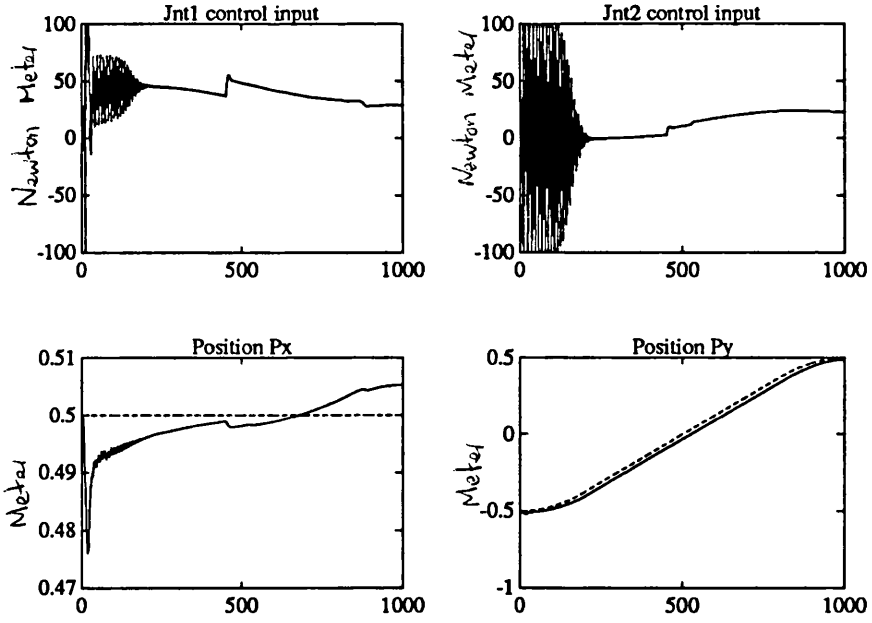



**Fig. 4.9 Control input for joint 1, joint 2 and position trajectory in X, Y directions compared with desired ones (dashed lines)**

In Fig. 4.9, the forgetting factor is chosen as $\lambda$=1.02. The control inputs chattering effect has been much reduced without much degrading of the tracking ability as can see from the third and fourth figure of Fig. 4.9 (compare with Fig. 4.8). It is worthwhile to mention that the payload, which is suddenly added near the middle of operation, does not cause any vibration as the initial effect does. What is more, the controller gains $s_0$, $s_1$, $s_2$, and $s_3$ in equation (3-9) of Chapter 3 approach constant values (for joint 1, $s_0$ --> 32645, $s_1$ --> -45749, $s_2$ --> 14754 and $s_3$ --> -0.4677; for joint 2, $s_0$ --> 26400, $s_1$ --> -37531, $s_2$ --> 12354 and $s_3$ --> -0.4958). This indicates that for a dedicated payload and manipulation speed a set of constant control gains can be acquired and stored using an adaptive law. And when the manipulator is required to repeated the same task, these gains can be used again to avoid the initial chattering phenomena.

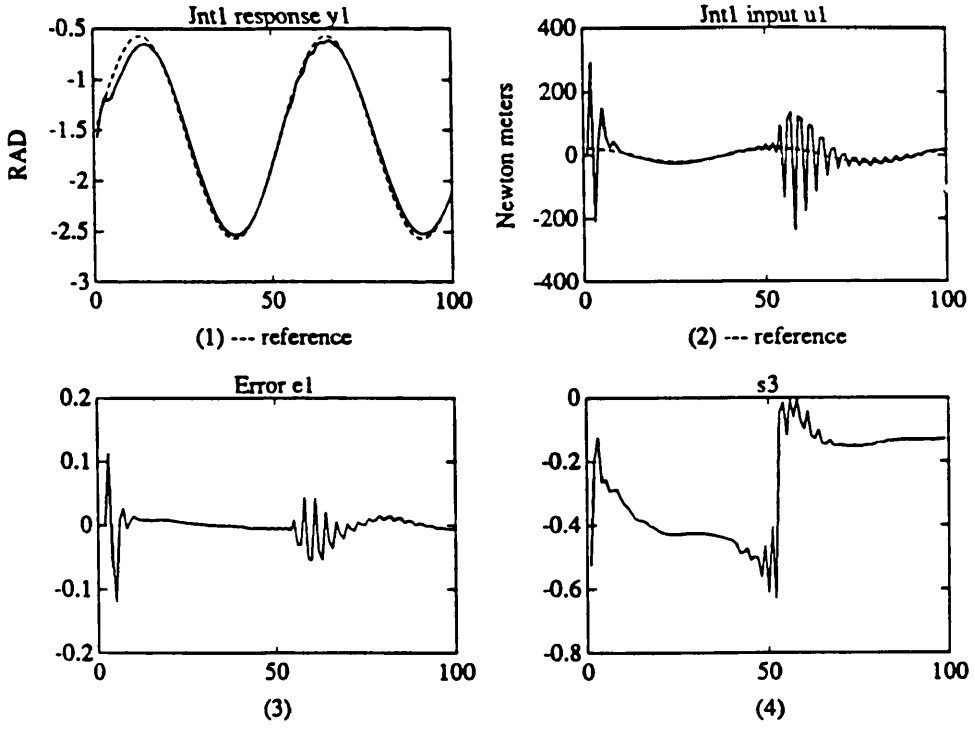



**Fig. 4.10 First joint simulation results**

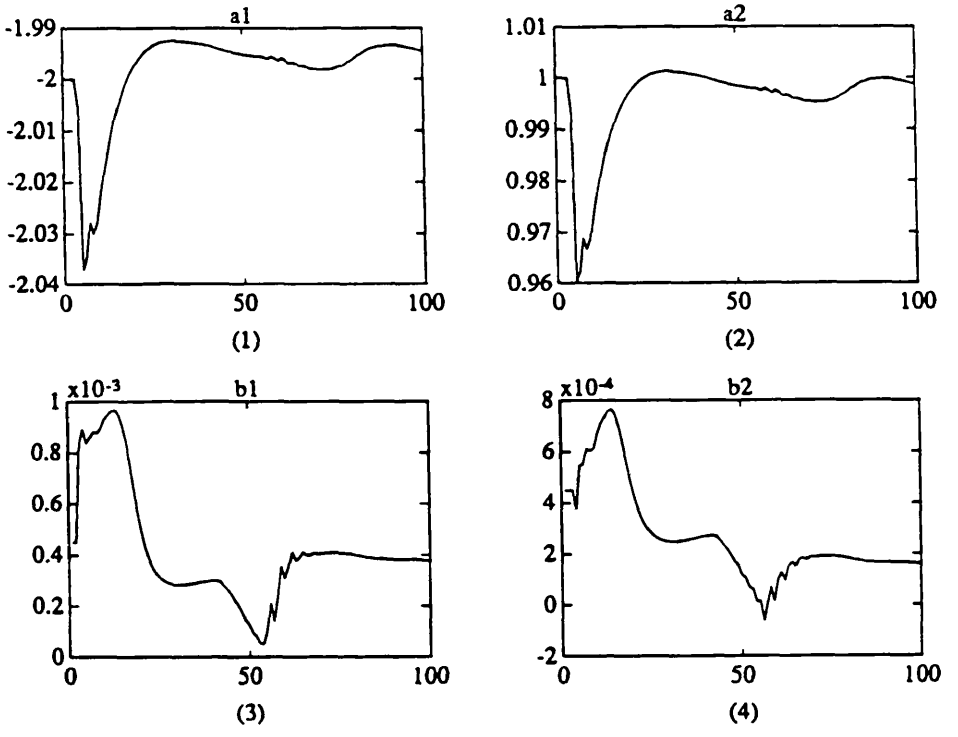



**Fig. 4.11 First joint parameters identified**

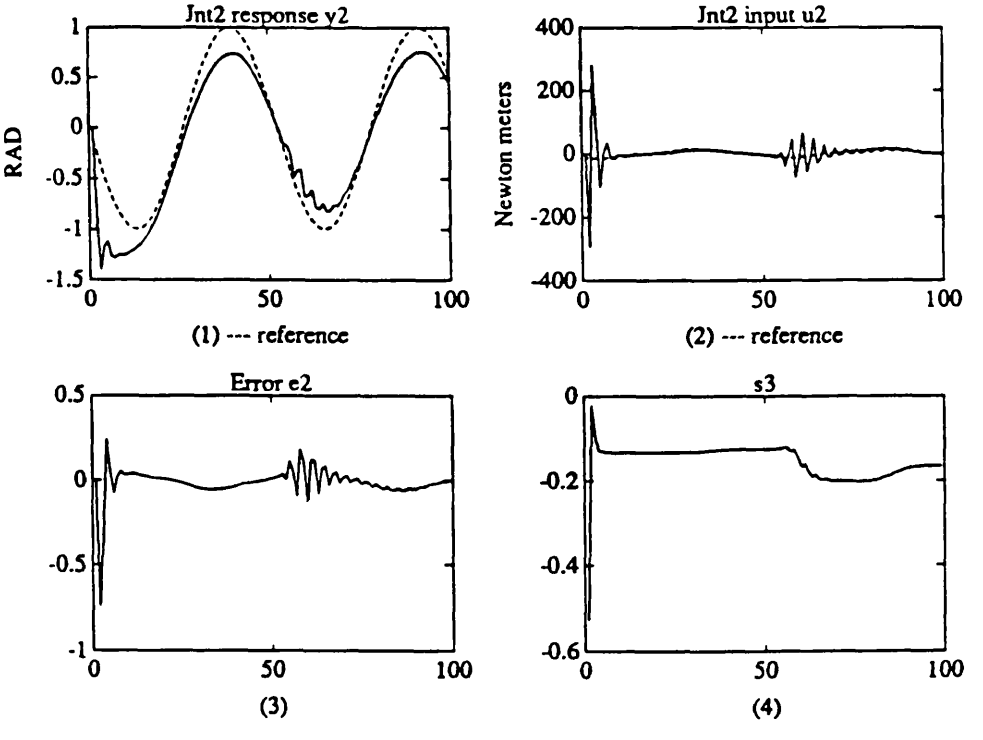



**Fig. 4.12 Second joint simulation results**

The conditions for the calculating in Fig. 4.10 to Fig. 4.13 are T=0.03, λ=0.96 and other parameters are the same as in Fig. 4.8 and 4.9. Fig. 4.10 and Fig. 4.12 show the first and second joint response (1), control input (2), identification error (3) and controller parameter s3 (4) respectively. Fig. 4.11 and Fig. 4.13 show process parameter identified. As can see, there are no true constant values for those parameters.

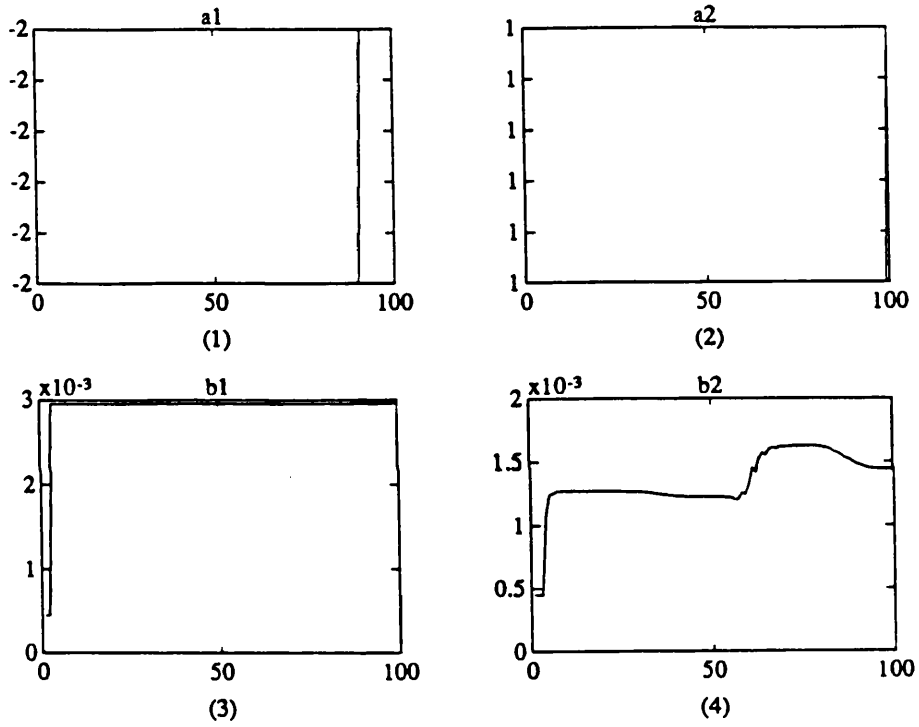



**Fig. 4.13 Second joint parameters identified**

As can be seen from Fig 4.11 and Fig. 4.13, the process parameters $a_1$, $a_2$ and $b_1$, $b_2$ are changing with time. It is also be seen that the process parameters are changing with different reference input r(k). The initial parameter values are chosen as: $a_1$=-2, $a_2$=1 and $b_1$=$b_2$=T*T (T is the sampling time period).

### 4.4.3 Summary

These results provide some general comments on adaptive control of robot manipulators. The lack of persistent excitation does not seem to prevent the

use of adaptive control. In the case of a robot manipulator with the task being specified beforehand, persistent excitation can rarely be satisfied. However, as to a specific physical system (e.g., second order system like an arm joint), it is not difficult to give limitations to the parameters and assign estimated initial values to them. If nearly correct initial values are given to the identification process, less persistent excitation can also give rise to the correct identification of the system parameters (Wang [1991]).

Stability is guaranteed if the parameters are time-invariant, as the pole-assignment method guarantees the closed loop poles being within the unit circle. If the process parameters are slow time-varying or fluctuate around a set value, then within the sampling time linear theory can still be used. However, if the process parameters are fast changing, then the sampling time must be reduced to a value when the process parameters can still be viewed as constant. There is no criteria for deciding this. Moreover, reducing the sampling time period means an even higher computational burden. One method to avoid reducing sampling time T is to compensate for dynamic coupling.

Identification will cause chattering phenomena if so called *true* values of the system parameters do not exist, and so the controller's gains keep changing. Accurate identification of system parameters is not necessary, especially in the robotic manipulator case with high non-linearity and coupling effect. However, slightly increasing the forgetting factor $\lambda$ will get rid of this chattering effect by identifying the average value of the system parameters and stick to these values, resulting in a fixed gain PID controller. As shown in section 3.3 of Chapter 3, a fixed gain PID controller can manage up to 24 per cent model mismatch. This is why existing industrial robotic manipulators work when working conditions are met (limit payload and speed).

## 4.5 BACKWARD SIMULATION SCHEME

A new simulation scheme for self-tuning adaptive control of robotic manipulators is described in this section. This scheme can make direct use of the Newton-Euler dynamic equations, which are thought traditionally as not convenient for control system simulation or for design of robotic manipulator control systems.

### 4.5.1 Description of the Method

As shown in Fig. 4.14, usually it is very easy to derive the inverse dynamics of a robotic manipulator using Newton-Euler (N-E) or Lagrange-Euler (L-E) method. This is a set of highly nonlinear, highly coupled and time varying equations. Linearization and rewriting in state space form is difficult.

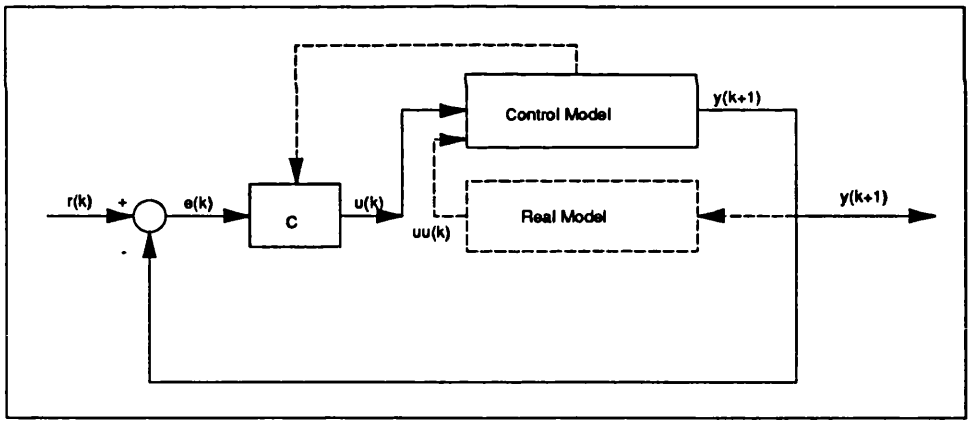



**Fig. 4.14 Backward simulation scheme**

As shown in Fig. 4.14, instead of using the forward dynamics equations to derive the output of the system y(k+1), it is suggested here that the reference *control model* is used to get the system response. This calculated response is of course an approximation, which will be more accurate with smaller $T$. The output of the system is then input 'backwards' to the system, *real model* (the inverse dynamic equations) to acquire the value of the reference system input, uu(k). This uu(k), together with y(k), is then used to identify the process

parameters $a_1$, $a_2$, $b_1$ and $b_2$, which are then used to update the gain parameters of the controller *C*. If the system response can follow the reference input *r* and the control input u can follow uu, then it can give us confidence in the system. Fig. 4.15 shows backward simulation in the situation of feed-forward compensation.

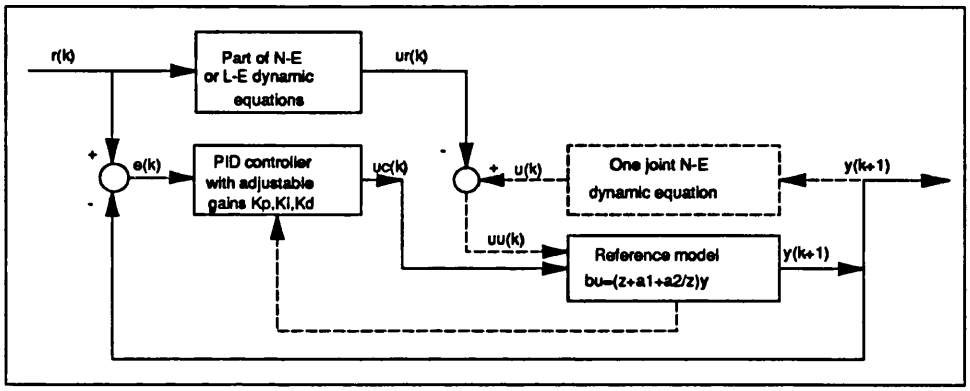



**Fig. 4.15 Compensated backward simulation scheme**

Programming can be achieved by the methods presented in Ogada [1987]:

1). Calculate the input u(k) using (3-8) in Chapter 3.

2). Calculate the output y(k+1) using the control model (3-7) in Chapter 3, that is:

$$y(k+1) = -a_1 \cdot y(k) - a_2 \cdot y(k-1) + b_1 \cdot u(k) + b_2 \cdot u(k-1)$$

3). The previous uu(k) can be calculated using the inverse dynamic equations (4-2), the inverse dynamics.

4). According to uu(k) and y(k+1), the process parameters can be identified using Recursive Least Square (RLS) identification.

5). The process parameters are then used for updating the controller's gains.

6). Return to step 1).

A backward simulation MODULA-2 program for one DOF pendulum, shown in Appendix 3.2 of Chapter 3 is listed in Appendix 4.2.

### 4.5.2 Simulation Results and Analysis

The simulation results using this scheme are presented as following. All the dotted lines express the reference values, and the denominator parameters are $t_1$=-0.9, $t_2$=$t_3$=$t_4$=0, which will give treble roots of 0 and a root of 0.9, if not otherwise stated. The sampling time period is chosen as T=0.03 (30 milliseconds) if not stated otherwise.

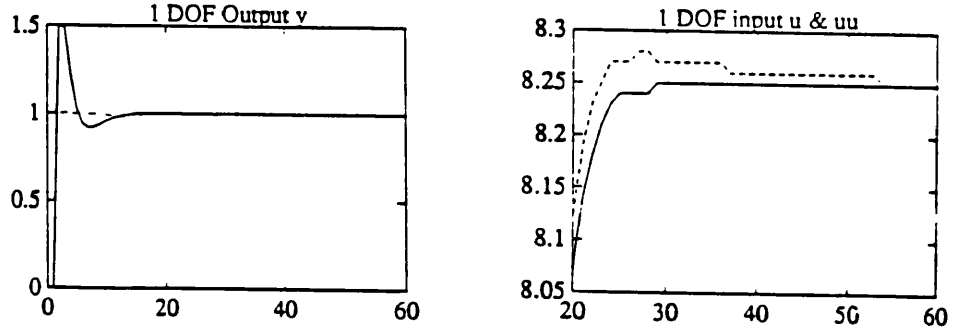



**Fig. 4.16 Backward simulation for 1 DOF pendulum with step reference input**

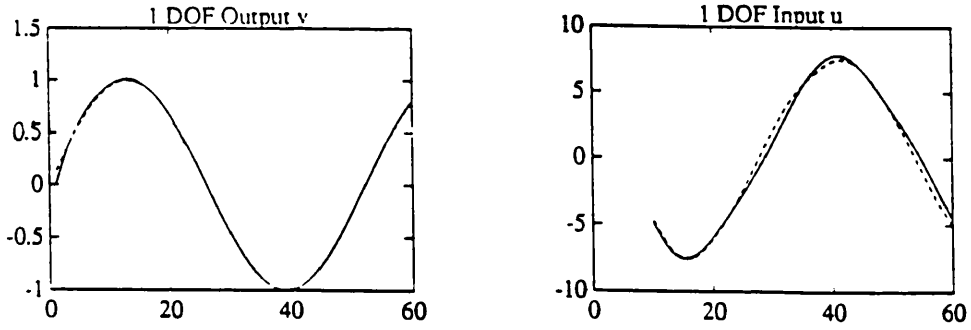



**Fig. 4.17 Backward simulation for 1 DOF pendulum with sin(4t) reference input**

In Fig. 4.16 and Fig. 4.17, calculating condition is T=0.03, $\lambda$=0.98. The SISO system shown in Appendix 3.2 of Chapter 3 is used for the simulation. In Fig. 4.16 step reference input is used and sin(4t) reference input is used for Fig. 4.17. Both figures show that the responses can follow the desired ones and the control inputs can follow the desired ones as well.

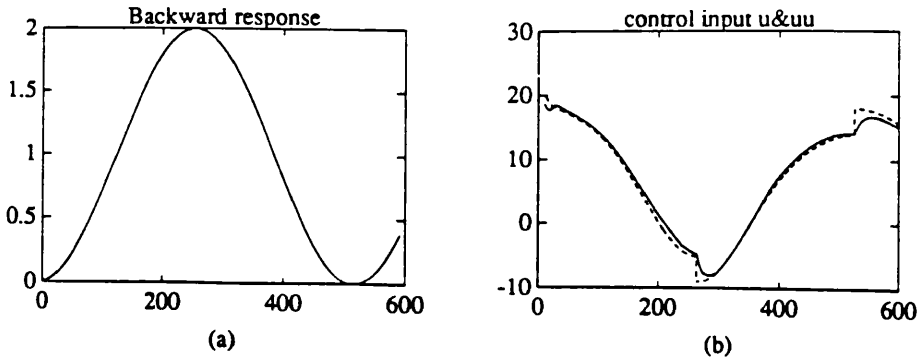



**Fig. 4.18 Backward simulation for 1 DOF pendulum with reference input 1-cos(4t)**

In Fig. 4.18, calculating condition is T=0.003, λ=0.98. A third reference input *1-cos(4t)* is used for simulating purpose.

A two DOF pendulum, shown in Appendix 4.1 which has less non-linearity than that shown in Fig. 4.2, was chosen to validate backward simulation scheme.

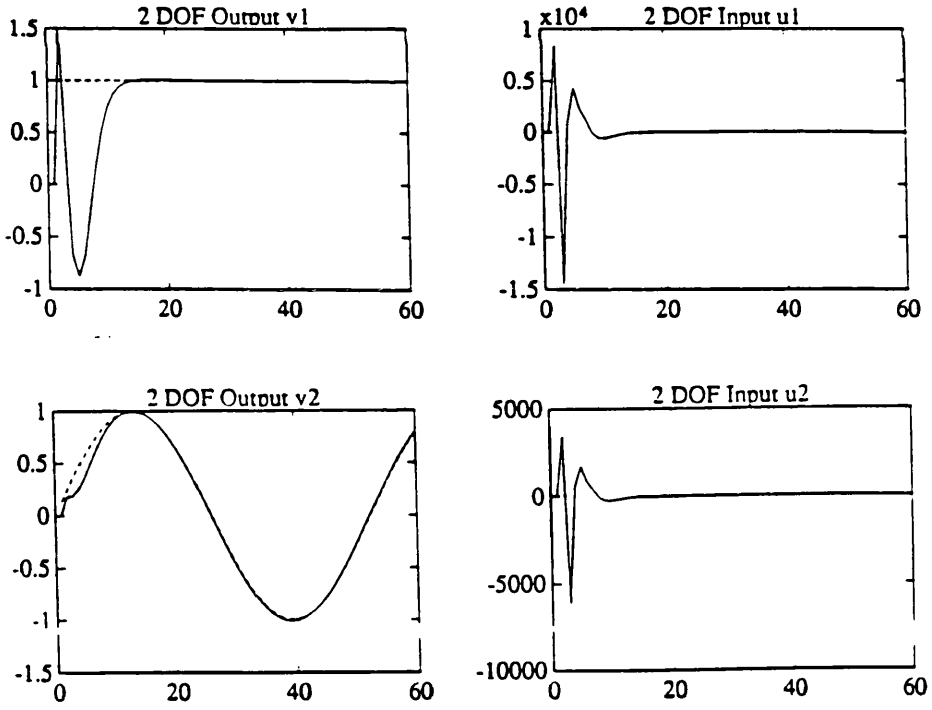



**Fig. 4.19 Backward simulation for 2 DOF pendulum with step input for joint 1 and sin(4t) for joint 2**

In Fig. 4.19, reference input of joint one is step function 1, and reference input of joint 2 is sin(4t). T=0.03 and $\lambda$=0.98.

### 4.5.3 Summary and Conclusions

The Newton-Euler equations can equally well be used in the simulation of the dynamic self-tuning adaptive control for robotic manipulators. The important thing is how to choose the control parameters of the PID controller to give the best error performance. So, a control model of second order is used to represent the dynamics of each joint. As the dynamics of the robot manipulator are highly nonlinear and highly coupled, this simple dynamics equation assigned to each joint are not the real case. So the parameters $a_1$, $a_2$ and $b_1$, $b_2$ are not actually the parameters of the system but only useful for the design of the controller's parameters $s_0$, $s_1$, $s_2$ and $s_3$.

It is interesting to compare the control input u(k), calculated from the controller, and the reference control input uu(k), calculated from the inverse dynamic equations, as shown in Fig. 4.16, Fig. 4.17 and Fig. 4.18, there are only slight difference between them.

Theoretically speaking, the reliable PID controller, which is widely used in industry, can drive the robot manipulator along any trajectory at any speed providing that a suitable set of the control parameters are chosen. However, in the existence of model uncertainty, disturbances and other factors, these control parameters can not be chosen beforehand properly. Adaptive control is necessary. Pole-assignment self-tuning, with its essential advantage --- robustness, is used to update the PID parameters in this scheme.

1). The Newton-Euler equations can equally well be used in the simulation of dynamic self-tuning adaptive control as the L-E equations.
2). This scheme can be used for the evaluation of an adaptive controller to see if the control law is correct when put to practical use.
3). This scheme can be used to provide the initial process parameters for real

time control.

4). All the conclusions and discussions refer only to the self-tuning adaptive controller design described in this thesis. Unfortunately, others model dependant adaptive control laws may still need to rewrite the L-E dynamic equations in the state space form.

5). The control methods proposed here, which are simple to calculate, are well worthwhile for real time practice.

## 4.6 SUMMARY

The conventional methods for investigating nonlinear systems involve investigation of equilibria and analysis of the local behaviour near the equilibria. Such an approach will give only local properties, although in some special cases it may be possible to proceed further and obtain global property (Astrom [1989] pp.235).

In an adaptive system it is assumed that the regulator parameters are adjusted all the time. This implies that the regulator parameters follow changes in the process. However, it is difficult to analyse the convergence and stability properties of such systems. To simplify the problem, it can be assumed that the process has constant but unknown parameters. When the process is known, the design procedures specify a set of desired controller gains. The adaptive controller should converge to these gains even if the process is unknown.

If the input signal to the process is sufficiently excited and the structure of the estimated model is appropriate, the estimates will converge to the true values if the closed-loop system is stable. Conditions for convergence for the different estimation methods are of great importance. Most of the design methods can be interpreted as pole placement design.

The adaptive control strategies have been derived under the assumption that the process parameters are constant but unknown. The practical real process is not always the case.

The notion of equilibrium is essential when dealing with nonlinear differential equations. The existence of true parameter equilibria depends on three elements: the nature of the external driving signal, the true system, and the model used to design the adaptive system. The true parameter equilibrium will always exist if the only excitation is the command signal and there are no un-modelled dynamics. If true parameter equilibrium do not exist, there may still exist equilibrium to the averaged equations. This corresponds to the physical situation in which parameters move around in the neighbourhood of a fixed value.

## Appendix 4.1 A Two DOF Pendulum and Its Dynamic Model

A two degree of freedoms robot manipulator (pendulum) shown in Fig. A4.1 is used to verify the control simulation scheme suggested here. The inverse dynamic equations are (Paul [1982]):

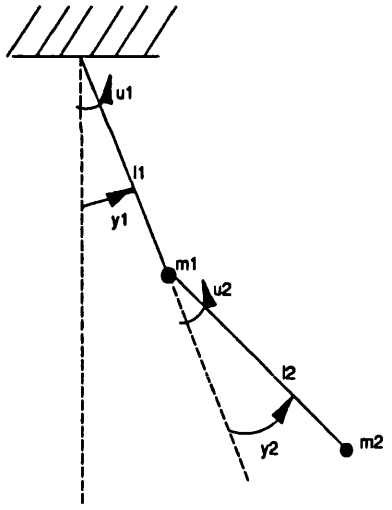



**Fig. A4.1 Two DOF Pendulum**

$$\tau_1 = D_{11}\cdot\ddot{y}_1 + D_{12}\cdot\ddot{y}_2 + D_{122}(\dot{y}_2^2 + 2\dot{y}_1\cdot\dot{y}_2) + D_1$$
$$\tau_2 = D_{12}\cdot\ddot{y}_1 + \ddot{y}_2 + D_{122}(\dot{y}_1^2 + 2\dot{y}_1\cdot\dot{y}_2) + D_2 \qquad \text{(A4-1)}$$

where

$$D_{11} = 3 + 2\cos(y_2)$$

$$D_{12} = 1 + \cos(y_2)$$

$$D_{122} = -\sin(y_2)$$

$$D_1 = 19.6\sin(y_1) + 9.8\sin(y_1 + y_2)$$

$$D_2 = 9.8\sin(y_1 + y_2)$$

suppose $m_1=m_2=l_1=l_2=1$, having unit values.

## Appendix 4.2 MODULA-2 Program for 1 DOF Backward Simulation

```
MODULE Pendu6B;
IMPORT IO, FIO, Lib, Respons1, MathLib0, VecLab;
(*-------------------------------------------------------------------*)
... ...
BEGIN
 Respons1.InputConstants;
 SetInitCondition;
 FOR i:= 0 TO INTEGER(1.8/T) DO
  t:= REAL(i)*T;
  r4:= r3; r3:= r2; r2:= r1; r1:= r;
  r:= 1.0-MathLib0.cos(4.0*t);
(**)
    beta1:= beta + P11*Y1*Y1+P22*Y2*Y2+P33*(uu)*(uu)+P44*(uu1)*(uu1);
    gama[1]:= -P11*Y1/beta1;
    gama[2]:= -P22*Y2/beta1;
    gama[3]:= P33*(uu)/beta1;
    gama[4]:= P44*(uu1)/beta1;
    e:= Y + a1*Y1 + a2*Y2 - b1*(uu) - b2*(uu1);
    a1:= a1 + gama[1]*e;
    a2:= a2 + gama[2]*e;
    b1:= b1 + gama[3]*e;
    b2:= b2 + gama[4]*e;
    P11:= (1.0 + gama[1]*Y1) * P11 / beta;
    P22:= (1.0 + gama[2]*Y2) * P22 / beta;
    P33:= (1.0 - gama[3]*(uu)) * P33 / beta;
    P44:= (1.0 - gama[4]*(uu1)) * P44 / beta;
(**)
     s0:= ( t1 - a1 ) / b1;
     s3:= (b1*b2*t3-b2*b2*t2-b1*b1*t4+b2*b2*a2+b2*b2*s0*b2);
     s3:= s3/(b1*b2*a1-b1*b1*a2-b2*b2);
     IO.WrReal(Y, 5, 9);  IO.WrLn();
     WriteData;
     s1:= ( t2-a2-s0*b2-s3) / b1;
     s2:= ( t4-a2*s3) / b2;
     u2:= u1;          u1:= u;
     u:= - s3*u2 + s0*(r-Y) + s1*(r1-Y1) + s2*(r2-Y2);
  Y4:= Y3; Y3:= Y2; Y2:= Y1;  Y1:= Y;
(**)
  Y:= -a1*Y1-a2*Y2 + b1*u+b2*u1;
(**)
(*Desired Torqe*)
  uu2:= uu1; uu1:= uu;
  uu:=  (1.0+ME)*(Y-2.0*Y1+Y2)/T/T + v*(Y-Y1)
```

```
    + C*VecLab.sign(Y-Y1) + g*MathLib0.sin(Y);
(**)
 END;
... ...
END Pendu6B.
```

## Appendix 4.3 MODULA-2 Program for 2 DOF Robotic Manipulator Simulation

```
(* 2 DOF nonlinear, coupled ST adaptive control*)
(*$o+*)
MODULE RbtCtrl3;
... ...
(*Main Programme*)
BEGIN
 Respons2.InputConstants;
 SetInitCond;
 i:= 1;
 REPEAT
  t:= REAL(i)*T;
  ref:= Motion.InverseKine(t, 0.0); (*think about this. 8/8/91*)
  r:= ref[1];  r4:= ref[2]; (*must be here! 9/8/91*)
  r3:= r2;   r2:= r1;   r1:= r;
  r7:= r6;   r6:= r5;   r5:= r4;
(*  r:= MathLib0.sin(4.0*t); r4:= -MathLib0.sin(4.0*t); *) (*Here!*)
  Respons2.CalcuDesireTorq(i, r4, r, 0.0, 0.0, 0.0, 0.25);
(**)(*Recursive Least-Square Identification for jnt2. 17/8/90*)
      beta2:= beta + P55*Y4*Y4+P66*Y5*Y5+P77*u21*u21+P88*u22*u22;
      gama2[1]:= P55*(-Y4)/beta2;
      gama2[2]:= P66*(-Y5)/beta2;
      gama2[3]:= P77*u21/beta2;
      gama2[4]:= P88*u22/beta2;
      WriteData;
      e2:= Yb + a3*Y4 + a4*Y5 - b21*u21 - b22*u22;
      a3:= a3 + gama2[1]*e2;
      a4:= a4 + gama2[2]*e2;
      b21:= b21 + gama2[3]*e2;
      b22:= b22 + gama2[4]*e2;
      P55:= (1.0 + gama2[1]*Y4) * P55 / beta;
      P66:= (1.0 + gama2[2]*Y5) * P66 / beta;
      P77:= (1.0 - gama2[3]*u21) * P77 / beta;
      P88:= (1.0 - gama2[4]*u22) * P88 / beta;
(**)
(**)(*Recursive Least-Square Identification for jnt1. 17/8/90*)
      beta1:= beta + P11*Y1*Y1+P22*Y2*Y2+P33*u11*u11+P44*u12*u12;
      gama1[1]:= P11*(-Y1)/beta1;
      gama1[2]:= P22*(-Y2)/beta1;
      gama1[3]:= P33*u11/beta1;
      gama1[4]:= P44*u12/beta1;
      e1:= Ya + a1*Y1 + a2*Y2 - b11*u11 - b12*u12;
      a1:= a1 + gama1[1]*e1;
      a2:= a2 + gama1[2]*e1;
      b11:= b11 + gama1[3]*e1;
      b12:= b12 + gama1[4]*e1;
      P11:= (1.0 + gama1[1]*Y1) * P11 / beta;
      P22:= (1.0 + gama1[2]*Y2) * P22 / beta;
      P33:= (1.0 - gama1[3]*u11) * P33 / beta;
```

```
    P44:= (1.0 - gama1[4]*u12) * P33 / beta;
(**)
  s3:=(t1-a3)/b21;
  s30:= b21*b22*t3-b22*b22*t2-b21*b21*t4+b22*b22*a4+b22*b22*b22*s3;
  s30:= s30/(b21*b22*a3-b22*b22-a4*b21*b21);
  s5:= ( t4-a4*s30)/b22;
  s4:= ( t2-a4-b22*s3-s30)/b21;
  s0:=(t1-a1)/b11;
  s00:= b11*b12*t3-b12*b12*t2-b11*b11*t4+b12*b12*a2+b12*b12*b12*s0;
  s00:= s00/(b11*b12*a1-b12*b12-a2*b11*b11);
  s2:= ( t4-a2*s00)/b12;
  s1:= ( t2-a2-b12*s0-s00)/b11;
  u23:= u22; u22:= u21;
  u21:= -s30*u23+s3*(r4-Yb)+s4*(r4-Y4)+s5*(r4-Y5);
  u13:= u12; u12:= u11;
  u11:= -s00*u13+s0*(r-Ya)+s1*(r-Y1)+s2*(r-Y2);
(*Bound input here 10/4/91*)
(**)
  IF (u21>80.0) THEN
    u21:= 80.0;
  END;
  IF (u21<-50.0) THEN
    u21:= -50.0;
  END;
  IF (u11>100.0) THEN
    u11:= 100.0;
  END;
  IF (u11<-100.0) THEN
    u11:= -100.0;
  END;
(**)
  State_Var:= Respons2.Rung_Kut(State_Var, u11, u21, i);
  Y2:= Y1; Y1:= Ya; Y5:= Y4; Y4:= Yb;
  Ya:= State_Var[1];
  IO.WrLn();
  IO.WrStr(' i='); IO.WrInt(i,4);
  IO.WrStr(' Px='); IO.WrReal(position[1],5,9);
  IO.WrStr(' b11='); IO.WrReal(b11,5,9);
  IO.WrStr(' a2='); IO.WrReal(a2,5,9);
  Yb:= State_Var[3];
  i:= i + 1;
 UNTIL (i > INTEGER(5.0/T));
... ...
END RbtCtrl3.
```

## Appendix 4.4 MODULA-2 Program for 2 DOF Runge-Kutta Numerical Integration

```
IMPLEMENTATION MODULE Respons2;
... ...
(**)(* Procedure to caculate response of plant 21/8/90 Q. Wang*)
  PROCEDURE DynaModel(y1, yv1, y2, yv2, u1, u2 : REAL; i : INTEGER);
  VAR
    u1p, u2p : REAL;
    G1, G2, H11, H12, H22, h, F1, F2 : REAL;
  BEGIN
    IF (i<N) THEN
```

```
      H11:=2.219+0.844*MathLib0.cos(y2);
      H12:= 0.332+0.422*MathLib0.cos(y2);
      H22:= 0.332;
      h:=  0.422*MathLib0.sin(y2);
      G1:= 39.518*MathLib0.cos(y1)+9.572*MathLib0.cos(y1+y2);
      G2:= 9.572*MathLib0.cos(y1+y2);
    ELSIF (i>=N) THEN
      H11:= 3.307 + 1.962*MathLib0.cos(y2);
      H12:= 0.861 + 0.981*MathLib0.cos(y2);
      H22:= 0.861;
      h:= 0.981*MathLib0.sin(y2);
      G1:= 22.253*MathLib0.cos(y1+y2) + 52.218*MathLib0.cos(y1);
      G2:= 22.253*MathLib0.cos(y1+y2);
    ELSE
    END;
    (*Friction*)
    F1:= 5.0*yv1 + 2.0*VecLab.sign(yv1);
    F2:= 2.5*yv2 + VecLab.sign(yv2);
    (* Friction is good for stability 12/8/91*)
    (* Does not affect RbtCtrl3!*)
(**)
    F1:= 0.0; F2:= 0.0;
(**)
    (*Gravitational Compensation 20/8/90*)
    u1p:= u1 + h*yv2*yv2 + 2.0*h*yv1*yv2 - F1 - G1;
    u2p:= u2 - h*yv1*yv1 - F2 - G2;
    IF (GravitationCompensation) THEN
      u1p:= u1 + h*yv2*yv2 + 2.0*h*yv1*yv2 - F1 - G1 + u1o;
      u2p:= u2 - h*yv1*yv1 - F2 - G2 + u2o;
    END;
    f1:= (H22*u1p - H12*u2p)/(H22*H11 - H12*H12);
    f2:= (H11*u2p - H12*u1p)/(H22*H11 - H12*H12);
  END DynaModel;
(**)
(*-------------------------------------------------------------*)

PROCEDURE Rung_Kut(Old_Vect : FOURREAL; u1, u2 : REAL; i : INTEGER) : FOURREAL;
VAR
 Y1, Yv1, An1, Bn1, Cn1, Dn1, BTn1, DTn1 : REAL;
 Y2, Yv2, An2, Bn2, Cn2, Dn2, BTn2, DTn2 : REAL;
 j : INTEGER;
 RESPN : FOURREAL;
 h : REAL;
BEGIN
 Y1:= Old_Vect[1]; Yv1:= Old_Vect[2];
 Y2:= Old_Vect[3]; Yv2:= Old_Vect[4];
 h:= 0.001;
 FOR j:= 0 TO INTEGER(T/h) DO
  DynaModel(Y1, Yv1, Y2, Yv2, u1, u2, i);
  An1:=  h/2.0*f1;              An2:=  h/2.0*f2;
  BTn1:= h/2.0*(Yv1 + 0.5*An1);     BTn2:= h/2.0*(Yv2 + 0.5*An2);
  DynaModel(Y1+BTn1, Yv1+An1, Y2+BTn2, Yv2+An2, u1, u2, i);
  Bn1:=  h/2.0*f1;              Bn2:=  h/2.0*f2;
  DynaModel(Y1+BTn1, Yv1+Bn1, Y2+BTn2, Yv2+Bn2, u1, u2, i);
  Cn1:=  h/2.0*f1;              Cn2:=  h/2.0*f2;
  DTn1:= h*(Yv1 + 0.5*Cn1);         DTn2:= h*(Yv2 + 0.5*Cn2);
  DynaModel(Y1+DTn1, Yv1+2.0*Cn1, Y2+DTn2, Yv2+2.0*Cn2, u1, u2, i);
  Dn1:=  h/2.0*f1;              Dn2:=  h/2.0*f2;
  Y1:=   Y1 + h*(Yv1 + 1.0/3.0*(An1 + Bn1 + Cn1));
  Yv1:=  Yv1+ 1.0/3.0*(An1 + 2.0*Bn1 + 2.0*Cn1 + Dn1);
  Y2:=   Y2 + h*(Yv2 + 1.0/3.0*(An2 + Bn2 + Cn2));
```

```
  Yv2:= Yv2+ 1.0/3.0*(An2 + 2.0*Bn2 + 2.0*Cn2 + Dn2);
 END;
 RESPN[1]:= Y1; RESPN[2]:= Yv1;
 RESPN[3]:= Y2; RESPN[4]:= Yv2;
 RETURN RESPN;
END Rung_Kut;
(*-------------------------------------------------------------*)
... ...
```

# CHAPTER 5
# ADAPTIVE COMPLIANT MOTION CONTROL

## 5.1 INTRODUCTION

Compliant motion control is concerned with the control of robotic manipulators contacting with their environment - with an object to manipulate or assemble, a welding seam to follow etc. - and is a major topic of current research. The last few years have witnessed an explosive growth in the study of robot force/compliant control. Reviewing these papers is complicated because of the variety of manipulator geometries studied and the diversity of tasks specified.

As stated in Chapter 1, two main groups can be distinguished: one group takes the compliant motion as a system, and the dynamics of contact include both the manipulator itself and the environment. Simulation including arm dynamics is presented in section 5.5. The other group apply a low stiffness compliant end-effector to be mounted at the robotic manipulator's end-effector, which is the main concern of this chapter.

As this compliant device has much lower stiffness than the arm, the manipulator's dynamic behaviour can be ignored by taking the manipulator and its control system as a unit gain. This research in force control, which will be described in the next few sections, mainly contributes to the second group.

The development of successful strategies and implementations for force control is seen as a crucial step in enabling robots to perform tasks, such as deburring, inspection, and parts assembly, which require significant interaction with the environment. The implementation of high-bandwidth, high-accuracy force-control, however, has proven to be quite difficult, primarily due to stability problems that occur upon contact with a rigid surface.

Whitney [1976] was the first to provide a stability analysis of a force controlled manipulator. The manipulator is modelled as a velocity-input integrator, and it is assumed that proportional position, velocity, and force feedback are implemented in discrete time. The environment is modelled as a spring and the following stability result was derived:

$$0 < TGK_e < 1$$

where $T$ is the sampling time, $G$ is the force feedback gain , and $K_e$ is the combined stiffness of sensor and environment. For a fixed $T$, this indicates that a tradeoff exists between $G$ and $K_e$. In other words, high bandwidth force control requires a compliant sensor or environment.

We observed that if position controlled manipulators are non-backdrivable, as is usually the case (PUMA560 is the case), and the environment is a rigid surface, the conclusion is that even very low force feedback gains should be sufficient to create contact instability. So, an elastic device has been assembled to go between the PUMA560 robot manipulator end-effector and the environment.

Other alternative approaches to improve force control exist, such as the use of passive compliance between the robot and environment. One example of passive compliance is the use of remote centre compliance (RCC), which is well-known as a solution to the peg-in-hole insertion problem. However the

RCC has a few faults: 1). The sensor system weigh$^{s}$ 2 to 3 kg, which will reduce the arm payload; 2). It is expensive; 3). It is only suitable for the peg-into-hole problem.

A typical force control around robot position controller can be modelled as in Fig. 5.1. The position controller normally consists of a high bandwidth (digital) P(I)D-controller; the force controller is a PI or PID controller. The PID controller is reliable and widely used in industry. It is rather difficult to acquire a set of suitable control gains $K_p$, $K_i$ and $K_d$ beforehand, especially when the dynamics of the system are not well known. So, adaptive control is necessary, and an adaptive outer-loop force controller will be discussed in section 5.4.

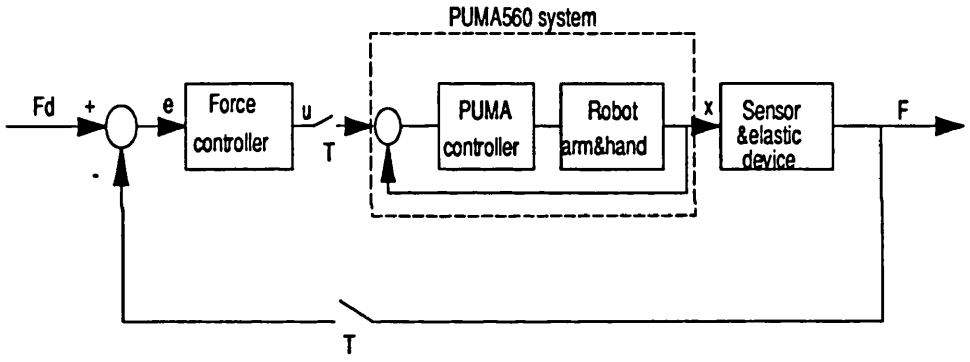



**Fig. 5.1 Force control scheme**

In the force loop around position loop approach, force errors are transformed into position commands, such that the robot essentially remains a position controlled device. Kazerooni [1989] has applied a different form of impedance control to the robotic deburring problem. He forces his end-effector to behave like a passive compliance device. In a simulation study, it was concluded that Kazerooni's impedance control gave improved performances. Bone etc. [1989] uses an active end-effector to get higher control bandwidth of the force control. Instead of adjusting the position of the end effector through the arm (by the arm position controller), the driven active end-effector is directly controlled using digital control (macro/micro manipulators). Higher force control

bandwidth and accuracy can be achieved at higher cost. However, this device is not general, but expensive and only one dimension of force adjustment can be accomplished.

In this study, a general active force control is accomplished with three dimensions of force adjustments (one contact, two twists). A supervisory computer, which runs in parallel with the VAL-2 controller, is linked to the PUMA560 (could be extended to a few robots) using the DEC communication protocol DDCMP. Supervisory control can be used to drive the robot end-effector to the vicinity of the destination. Then the supervisory computer can be engaged to do other tasks like trajectory planning for the next steps, sonar and vision processing etc., which are especially important in sub-sea inspection and other applications where remote operation is needed. The internal ALTER command provided by VAL-2 will read in the force signal and make path modifications on its own. Path modifications can be achieved by the command ALTEROUT. A NONALTER command will re-establish the supervisory control (Refer to Unimate [1985]), send all the force information to the IBM for analysis and wait for the next command from the IBM. Supervisory control will be discussed in more detail in the later chapters.

## 5.2 ROBOTIC CONTACT DYNAMICS

The contact dynamics of the robotic end-effector with its environment is discussed in this section. It is common practice, in the literature, to treat contact as one dimensional contact and static for simplicity. However, as will be pointed out in this thesis, contact itself is of a dynamic nature, and three different cases of typical contacts, which are point, line and surface contacts are defined. These three types of contact dynamics will be discussed in separate sub-sections. Surface contact is especially useful in sub-sea inspections, which will be discussed in Chapter 7. With point contact, all the 6 DOF are individually not coupled. With line contact, 2 DOF are coupled. With Surface contact 3 DOF are coupled.

Careful analysis of robotic contact is necessary for precise robotic manipulation like deburring and inspection etc. As in the inspection example, some kind of probe has to be placed against a surface, and at the same time it is also needed to move around the surface smoothly. Signals coming out from the probe are usually noisy, so it is highly desirable if the manipulator with the probe can be always made to contact the inspection object. And the contact should be maintained as constant as possible to generate as little noise as possible.

This requires the contact control system to be stable and to reject outside disturbances. Typical disturbances are surface friction and surface uncertainties. As will be seen in Chapter 7, for system stability, contact should be less stiff. However, small stiffness will make friction too sensitive to move the probe.

What is analysed here, is only suitable for outer-loop force control, with wrist force sensor and wrist compliant elastic device. The stiffness of the compliant device is usually much lower than the arm servo stiffness, resulting in a contact of dynamic nature. If the compliance is from each joint, or from the controller, the analysis is quite different, and is discussed in section 5.5.

There are two cases, which have to be treated differently, one being that a force sensor is not available. The other is that a force sensor is available. The dynamic relationship between displacement $x$ of the robotic end-effector and $d$ or $F$, the required responses, are obtained here for both situations. In the case of no force sensor, a second order dynamic relationship has been obtained between the drive force $x$ and the response $d$.

It is common practice to treat each dimensional contact individually, although the real thing is a six DOF contact. A point contact is six dimensional, and a surface contact is three dimensional. In the point contact case, the six dimensions can be treated individually. However, in the surface contact, the

three dimensions are coupled. For the purpose of simplifying the dynamic analysis, we suppose the contact is de-coupled, and each contact dimension is treated individually.

### 5.2.1 Contact Impact Effect

Unless the inspection task can be totally pre-planned, the robotic manipulator's approach and contact phases can not be separated. This is especially true in sub-sea inspection where the robotic manipulator has to try many directions to make contact with its working environment. A CCD camera can only provide limited information, which is often degraded because of poor visibility. Tactile sensors can give the computer an approximate model of the working space. Because all of this uncertainty about the workpiece, the robot contact is unpredictable and impact effects can not be avoided.

If the environment is accurately modelled by the computer, the robotic manipulator can be moved first to the vicinity of the environment quickly. Contact can then be approached at a reduced speed, so as to reduce the impact effect on the arm system. However, many applications can not meet this condition. Contact is often made with the robotic manipulator moving at some speed. The end-effector is stopped at the instant of contact, and arm kinematic energy must be transferred to other energy forms like heat etc. But most of the kinematic energy will be absorbed by the weakest part of the arm and sensory system (Zheng etc. [1991]).

The weakest part, which is usually the force sensor, may be damaged if there is no other less compliant device present in the control system. So there is a necessity for a compliant device to be present for sub-sea and any other ill-organised environment. A compliant device is also important for the contact system to be stable, as discussed before. Features of this compliant device will be described in the next chapter, together with its necessary electrical circuits.

### 5.2.2 Dynamics of Point Contact

In the case of point contact, the 6 dimensions are de-coupled and can be treated individually. In this sub-section just one dimension is taken into account and other dimensions are supposed having the same nature. In this thesis, as Kazerooni [1989], the end-effector is modelled as a passive mass. A second order system is modelled to describe the dynamics of contact. In practice, where most of the time the environment can be thought of as infinite stiff, then the system order can be reduced. With an unknown compliant environment, the general contact dynamics of the robotic manipulator and the environment is shown in Fig. 5.2.

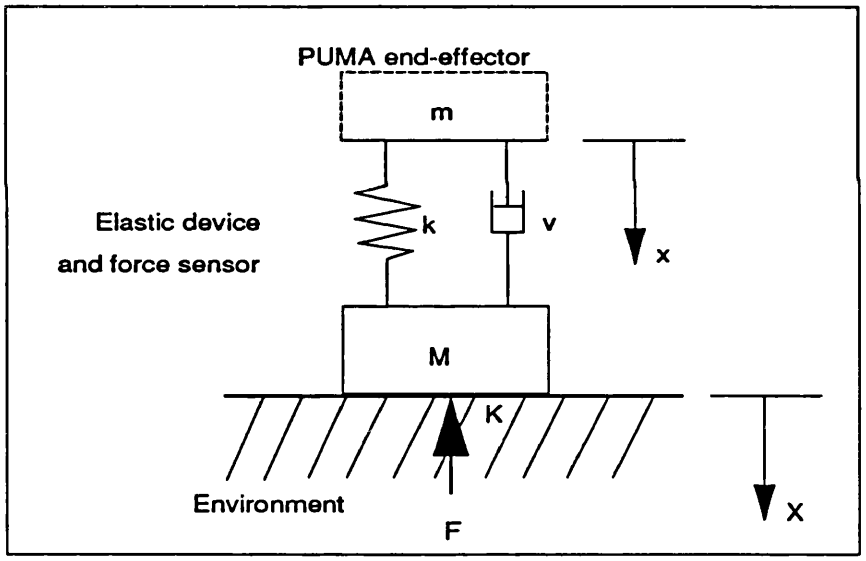



**Fig. 5.2 Dynamic model of contact**

The dynamics equations which govern the above system are:

$$M\ddot{X} + v\dot{X} + k(X - x) = -F \tag{5-1}$$

$$F = K \cdot X \tag{5-2}$$

where

$K$ — stiffness of environment.

$k$ — combined stiffness of the elastic device and force sensor.

$v$ — viscous friction coefficient.

$F$ — force applied by environment.

$X$ — displacement of mass $M$.

$x$ — displacement of robot manipulator end-effector, the drive variable for force control.

$M$ — mass of the probe.

In equation (5-1), substituting $X$ with $F$ in equation (5-2), yields:

$$\ddot{F} + \hat{v}\dot{F} + \hat{k}F = \hat{b}x \tag{5-3}$$

This is a second order differential equation. The system order can be reduced to one if the mass $M$ can be ignored. Where

$$\hat{v} = \hat{v}(M,K,k,v)$$
$$\hat{k} = \hat{k}(M,K,k,v)$$
$$\hat{b} = \hat{b}(M,K,k,v)$$

are all functions of the unknown parameters $M,K,k$, and $v$, which need to be identified. Nonlinear Coulomb friction is not included in the above description, which will be treated as a disturbance. If the environment can be thought of as infinitely stiff, then $X$ in equation (5-1) would be almost zero. Equations (5-1) and (5-2) are simplified as $F=kx$. Only one parameter $k$ needs to be identified in this simplified case. So, a PID controller with constant gains will work well providing $k$ is known a priori.

### 5.2.3 Dynamics of Line Contact

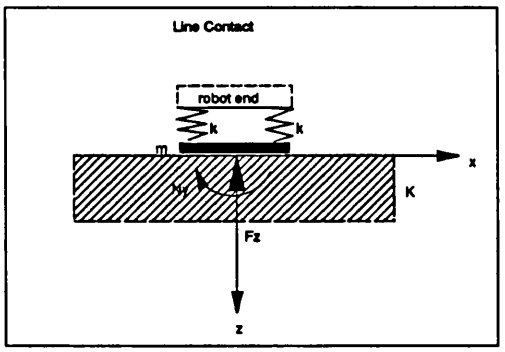



Fig. 5.3 Normal contact

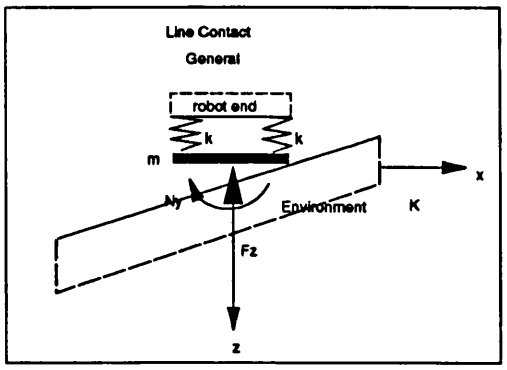



Fig. 5.4 Oblique contact

In the line contact case, the two coupled dimensions are to be taken into account, and the rest can be treated individually as in the former sub-section. As shown in Fig. 5.3, if the robotic end-effector approaches this way to its environment, then twisting moment $N_y$ equals zero and axial force $F_z$ is only dependent on the z-direction displacement $d_z$ of the robotic manipulator.

However, this is a special case, and in general, the approach phase to an unknown environment would be as shown in Fig. 5.4. For a random environment surface, this type of contact will be very often encountered by the robotic end-effector. Fig. 5.4 shows that the right side of the line makes the contact, the left side is equally frequent to make contact with environment. Oblique contact is especially correct when it is needed to move the robotic end-effector around an unknown environment.

The contact force $F_z$ and torque $N_y$ measured by the force sensor will have the following relationship:

$$F_z = \frac{2}{l}|N_y|$$

or (5-4)

$$N_y = \pm\frac{l}{2}F_z$$

This system with the responses being related to each other is called a coupled system.

We learnt in the former sub-section that a compliant contact is a second order system, which means the relationship between system inputs and system outputs are second order differential equations. For a coupled system, these relations would be:

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \cdot \begin{bmatrix} \ddot{F}_z \\ \ddot{N}_y \end{bmatrix} + \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} \cdot \begin{bmatrix} \dot{F}_z \\ \dot{N}_y \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \cdot \begin{bmatrix} F_z \\ N_y \end{bmatrix} = \begin{bmatrix} d_z \\ r_y \end{bmatrix} \quad (5\text{-}5)$$

### 5.2.4 Dynamics of Surface Contact

Surface contact is very useful in many applications, one example being sub-sea inspection. Usually, a probe is mounted to the force sensor or compliant device, which is fixed to the robotic end-effector. The probe, usually a ultra-sonic or other electric transducer, is a rectangle in its surface which is required to move around the working surface maintaining constant normal contact.

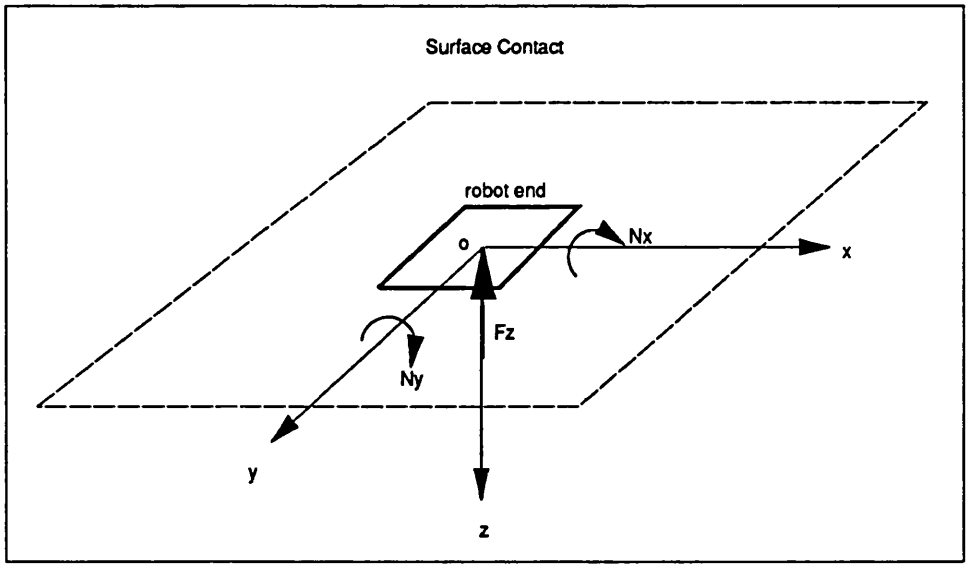



Fig. 5.5 Surface contact

With the background in 5.2.2, it only needs to be extended to three dimensional case for surface contact. The coupled dynamics equations would be:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \cdot \begin{bmatrix} \ddot{F}_z \\ \ddot{N}_y \\ \ddot{N}_x \end{bmatrix} + \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} \cdot \begin{bmatrix} \dot{F}_z \\ \dot{N}_y \\ \dot{N}_x \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \cdot \begin{bmatrix} F_z \\ N_x \\ N_y \end{bmatrix} = \begin{bmatrix} d_z \\ r_x \\ r_y \end{bmatrix} \tag{5-6}$$

Equation (5-3), (5-5) and (5-6) can be simplified if the mass of probe, $M$, can be thought as negligible.

### 5.2.5 Contact Dynamics Without a Force Sensor

In the analysis of the above sub-sections assumed that a wrist-mounted force sensor is available to measure the forces/torques in the operational space. Some times this demand is not possible as a force sensor may be too expensive. In the case of no force sensor, system identification and adaptive control can still be used for the control of the contact force, which has been transferred to displacement, between the robotic end-effector and the environment.

As can be seen in Fig. 5.2, $k$ is the stiffness of the compliant device, $K$, the stiffness of the environment, $x$, the displacement input to the system is the end-effector displacement of the robotic manipulator, and $X$ is the displacement of the environment.

IF $K=0$, then the distance between robot end-effector and its environment $d=0$; if $K$ tends to infinity then $d=x$; In the general case, the position relationship between $x$ and $d$ can be described by the following equation:

$$d = x - X \tag{5-7}$$

And the static force relationship has the following equation:

$$kd = KX \tag{5-8}$$

From equation (5-7) and (5-8), we obtain the following static relationship between $d$ and $x$:

$$d = \frac{K}{k+K} \cdot x \tag{5-9}$$

let

$$b = \frac{K}{k+K}$$

and $b$ is the only parameter to be identified.

If the stiffness of the compliant device is known a priori, the stiffness of the unknown environment $K$ can be calculated after $b$ has been identified.

If the dynamics are to be considered, equation (5-8) is changed to the following relation:

$$kd = KX + v\dot{X} + M\ddot{X} \tag{5-10}$$

from (5-7), substitute $X$ in (5-10):

$$M\ddot{d} + v\dot{d} + (k+K)d = Kx \tag{5-11}$$

supposing the first and the second differential of $x$, which is the position displacement of the robotic end-effector, are zeros.

Equation (5-11) shows a second order system between $d$ and $x$, and the transfer function of which is:

$$G(s) = \frac{K}{Ms^2 + vs + (k+K)} \tag{5-12}$$

## 5.3 CONVENTIONAL OUTER-LOOP FORCE CONTROLLERS

Control of active compliant motion is a complex problem, since the task frame directions do not correspond one-to-one to the joint-space degrees of freedom. As a result, every joint contributes in general both to the execution of motions in the position-controlled and in the force-controlled directions of the task frame, while in the end, whatever the particular control implementation, only a single torque is applied at every joint. Co-ordinate transformations have to be performed in the control loop. In the following few sections of this chapter, we discuss the outer-loop force control scheme with an extra compliant device, which will largely simplify the problem, as the arm dynamics can be ignored.

As an example, Fig. 5.1 shows the general one-dimensional (z direction in TOOL frame) external force-control scheme. The error between desired force $F_d$ and actual force $F$ is fed into the force controller, which generates position commands ($u$). The difference between actual robot position $x$ and the position of the environment $X$ causes a contact force via stiffness $k$. This contact force acts as a disturbance on the position-control loop (PUMA560 controller). The sensor dynamics may be neglected if the structural resonance frequency introduced by the force sensor lies well above the position-loop bandwidth. An extra passive compliance is added to the arm sensor system, and its role in the active impedance control is analysed in depth in this section.

The position controller normally consists of a high bandwidth (digital) P(I)D-controller; the force controller is a PI or PID controller, the gain of which is inversely proportional to the contact stiffness $k$, which combines both the stiffness of the elastic device/sensor and the environment. The behaviour of the original positioning system is not influenced very much by changing $k$ for positioning in contact with the environment when compared with positioning in free space. However in applications where a constant contact

force is to be maintained, too high a contact stiffness $k$ leads to unstable behaviour. On the other hand, the response of the force control loop becomes slower with lower $k$-values.

Fig. 5.1 shows the general one-dimensional external force-control scheme. Various control methods can be applied to this problem. The effect of contact force on the inner position loop may be neglected in many circumstances, because: firstly the stiffness of the compliant device is usually small compared to the servo stiffness; and secondly, the ir-reversibility of most joint drive systems prevents the passing of end-effector force back to previous links in the kinematic chain, ie most industrial manipulators are not back-driveable (Schutter and Brussel [1988]).

### 5.3.1 On-off Control

The advantage for this control is its simplicity and robustness. The control logic of this kind of controller has been discussed in Fei [1987]. This simple controller has also been used for the outer-loop force controller and has shown good robustness and disturbance rejection. The logic sequence of movement is:

$$\text{If } F > F_d, \text{ then } x = x - \delta x \qquad (5\text{-}13)$$
$$\text{If } F < F_d, \text{ then } x = x + \delta x$$

Where $\delta x$ is the motion step. The force control is accomplish by controlling position. This is different from impedance control; the dynamics of the arm and sensor are not included. For a quick motion, $\delta x$ of the above movement logic should be large. However, larger $\delta x$ will cause stability problems if arm, sensor and environment are all very stiff. Smaller $\delta x$ is better for robustness and stability of contact, but results in a sluggish response. The low limit of $\delta x$ is decided by the robotic manipulator overall resolution, i.e. 0.01 mm (transpositional) and 0.005° (rotational) for PUMA560.

### 5.3.2 Impedance Control

Controllers used in this type of force control are described here. Used initially for force control in assembly, a number of simple control laws were developed based on simplified static models for the arm and end environment stiffness. The first was **stiffness control:**

$$\begin{aligned} x &= K_c \cdot (F_d - F) \\ &= K_c \cdot e \end{aligned} \tag{5-14}$$

where $x$ is the commanded position (supposing $x=u$); $K_c$ is the controller stiffness, usually equals to $1/k$; $F_d$ is the desired reference force; $F$ is the measured force, and $e$ is the force error. The robot's positioning errors were not eliminated.

With **damping control:**

$$x = K_b \int e \cdot dt \tag{5-15}$$

where $K_b$ is the controller damping and the force error is integrated to eliminate any steady state offset. A more general form incorporating both stiffness and damping terms is **impedance control:**

$$x = K_c \cdot e + K_b \int e \cdot dt \tag{5-16}$$

In this form, impedance control is identical to conventional proportional-integral (PI) control. Proportional-integral-derivative (PID) control was used in Stepien etc. [1987].

### 5.3.3 Approaching Phase

As stated in sub-section 5.2.1, contact is made mostly when the robotic manipulator is in motion, with a resulting impact. Many references pay little attention to this problem and implicitly make the following assumptions: 1). The approach phase is executed under position control: the robot moves at a reduced speed while the force reading is continuously monitored; 2). The motion, out of force control directions, is purely position controlled.

In these studies, as indicated before, a compliant passive device is going to be assembled between the robotic end-effector and the force sensor. So the impact effect can be limited to a minimum and also the approaching speed is limited to be proportional to the desired force $F_d$ when it is in the non-contact phase.

### 5.3.4 Influence of the Position Resolution

Practical systems have limited position measurement accuracy which depends on the minimum division of the joint encoder in the manipulator. This results in force limit cycles when a constant-contact force is desired in a static situation and no robot motion is involved. The peak-to-peak force amplitude depends on the following factors:

$$\Delta F_{p-to-p} = \alpha k \Delta x_{res} \tag{5-17}$$

where $\Delta x_{res}$ is the resolution of the position measurement, which is dependant on the all joint encoders, $\alpha$ is a constant and $k$ is the stiffness of the compliant passive device.

In general, $k$ and $\Delta x_{res}$ decide the upper bound for amplitude of the limit cycles. Smaller stiffness $k$ and higher joint encoder resolution will give less force variation. Stiffness $k$ has other effect on the outer-loop controller and will be discussed in the next sub-section.

### 5.3.5 Role of Stiffness of the Passive Device

The stiffness $k$ influences the dynamic behaviour of the force-control loop in several ways.

- The influence of the contact force on the position loop is determined by the relative stiffness ($k$ compared with the stiffness of the joint position servo loop).
- It directly affects the force resolution and accuracy in the presence of finite position resolution as described in the last section.
- The disturbance rejection is proportional to $k^{-1}$.
- The stiffness $k$ should be less stiff than the more precise force sensor to protect the arm and sensor system when initial contact is made and to reduce impact.
- It also affects the execution speed of the compliant motion. For example, if it is too flexible, the reaction force will not be high enough to overcome such forces as friction and jamming in assembly.

In conclusion, lower structural stiffness is advantageous for compliant motion control, such as high disturbance rejection and high force resolution etc.

### 5.3.6 Final Remarks

In this study, a supervisory computer is linked to the PUMA560 system. The use of a supervisory system offers the following benefits (Savut etc. [1988]):

1). It enables the analysis of sensory information, which is essential for incorporating visual and sonar feedback.
2). It provide a favourable environment for program development, with use of high level languages, such as PASCAL, MODULA-2 or C, and better editing, printing and storage facilities.

3). Computing power is substantially increased without any penalty in execution time, as the supervisory computer runs in parallel with the VAL-2 controller.
4). It provides the capability to support computer graphics.
5). High level tasks such as operator interfacing and tasks scheduling are performed by the supervisory computer, thus relieving the VAL-2 controller to perform the manipulator control tasks only.

To prevent damage to the arm and the passive end-effector the position command from equation (5-13) to (5-16) is constrained to:

$$x \leq 3.5mm \tag{5-18}$$

The design methods described in sub-section 5.3.1and 5.3.2 do not take the dynamics of contact into account. Also the controller is not adaptive and is not well suitable for an unknown environment. In the next section the author is trying to use self-tuning adaptive control of the end-effector contact force to adapt the robotic manipulator itself to the unknown environment to provide better tracking ability.

## 5.4 ADAPTIVE OUTER-LOOP CONTROLLER

In the former section, we had a detailed discussion of the conventional outer-loop force control schemes. All these approaches ignored the dynamic characteristics of the contact. The environment is simplified as a spring with a stiffness, and the arm is likewise simplified as a spring with stiffness $k$ (often with damping as well). The contact control schemes are based on a static model.

As we discussed in section 5.2, contact itself is of a dynamic nature. Three types of contact have also been identified, ie point contact, line contact and surface contact. The last one is of most practical use as most contacts are

between two surfaces. In the case of surface contact, three degrees of freedom (direction of contact, two associated twist directions) are coupled. What is more, all the contact dynamics are of second order.

As said in the introduction section, all the discussion in this chapter, except section 5.5, is based on the fact that the inner-loop of the manipulator system can be neglected by adding an extra compliant passive device between the robotic manipulator end-effector and the force sensor. Because of this, we can put our main effort into dealing with the contact alone. The whole system of manipulator and contact will be treated in the next section.

The self-tuning adaptive control strategy given in Chapter 3 is used for the outer-loop force control. Adaptive control strategies are especially useful when the environment is not clear, with some parameters to be identified.

### 5.4.1 Adaptive Force Control

All the control approaches discussed in the former section can be used to form the adaptive control structure. The difference is that these control gains are to be updated using some means of estimating the contact dynamics. Each direction of the robotic manipulator end-effector is treated individually, although they are coupled.

Supposing the environment has unknown compliance with finite stiffness $K$, then adaptive control is necessary. As shown in Fig. 5.6, the gains of PID controller are continually updated using the most recent estimation of the contact dynamics. A linear reference model is assigned to the system and it is assumed that the system will behave as a linear time-invariant model during the sampling interval $T$. Within the sampling time period linear control theory can be used to design the feedback law, which is pole assignment in this research.

As stated before in section 5.2, the contact dynamics can be described as a second order differential equation, which has the following discretize transfer function (Wang [1990]):

$$G(z^{-1}) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{5-19}$$

$$F(k) = G(z^{-1})u(k) + d(k) \tag{5-20}$$

Where z is the shift operator. This is the general discretized form for a second order system, with four parameters to be identified. Where *F(k)* and *u(k)* represent the measured force variable and the driving signals (i.e. PUMA560 end-effector position displacements or angle displacements and *d(k)* is the disturbance at instant *k*. Thus for each controller the process parameters are continually estimated in order to adapt the PID gains.

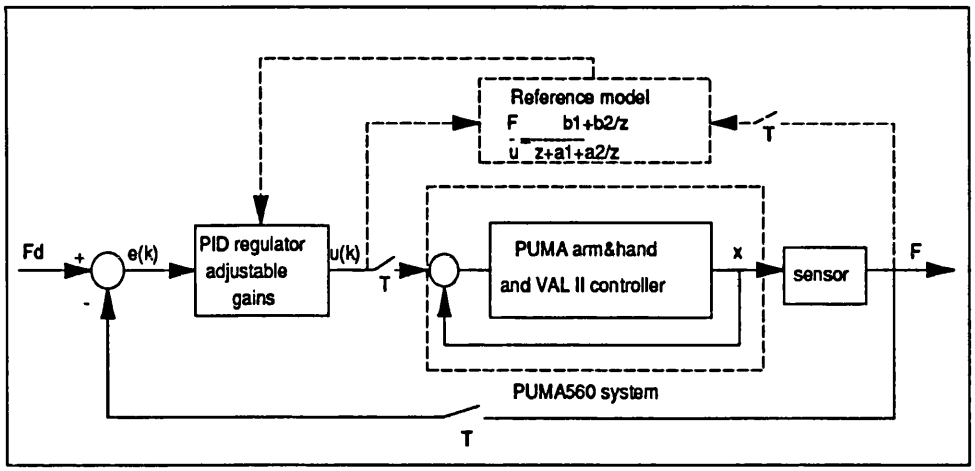



**Fig. 5.6 Adaptive outer-loop force control**

The approximate extended Recursive Least Squares (RLS) method described in Chapter 3 is used to estimate the process parameters on-line. The controller gains are calculated using the most recently estimated process parameters in such a way as to set the position of the closed loop poles to user definable

values within the unit circle, thus largely defining the dynamics of the closed loop response. By joining (3-6) in Chapter 3 and (5-20), the closed-loop transfer function of the system is:

$$F(k)=\frac{(b_1\cdot z^{-1}+b_2\cdot z^{-2})\cdot(s_0+s_1\cdot z^{-1}+s_2\cdot z^{-2})}{1+(b_1\cdot s_0+a_1)\cdot z^{-1}+(b_1\cdot s_1+a_2+b_2\cdot s_0+s_3)\cdot z^{-2}+(b_2\cdot s_1+b_1\cdot s_2+a_1\cdot s_3)z^{-3}+(a_2\cdot s_3+b_2\cdot s_2)z^{-4}}\ F_d(k)$$

(5-21)

Let the denominator be:

$$T(z^{-1})=1+t_1\cdot z^{-1}+t_2\cdot z^{-2}+t_3\cdot z^{-3}+t_4\cdot z^{-4}$$

Then, an explicit relation between the process parameters and the controller's gains can be acquired as equation (3-9) in Chapter 3.

### 5.4.2 Adaptive Compliant Motion Control

In sub-section 5.2.5, the dynamics of contact were discussed when a force sensor is not present in the contact situation. The force of contact is transferred from some position displacement measurements, which are available at the compliant passive device, as will be seen in the next chapter. What was discussed in the former sub-section can be applied in this situation. It is simply necessary to substitute $F$ in the equations (5-20), (5-21) with displacement $d$. As we had already seen in sub-section 5.2.5, the relation between robotic end-effector displacement and the resulting position outputs has dynamics of second order.

There are many situations where a force sensor is not necessary. Take sub-sea inspection as an example, as there is marine growth and surface roughness etc., precise force measuring is more than necessary. A multi-degree of freedom (at least five) compliant device with positional measurement is enough for the sub-sea inspection task. This will be further explained in Chapter 7.

## 5.5 FORCE CONTROL WITH MANIPULATOR DYNAMICS

### 5.5.1 Introduction

In the former sections, it has been supposed that the manipulator dynamics can be ignored, since there is a compliant elastic device between manipulator end-effector and its environment. The whole manipulator can be taken to be a unit constant gain because the stiffness of the compliant device is much less than the manipulator itself, which will not, subsequently, affects the manipulator's dynamic behaviour.

As we shall see in the next chapter, a lower stiffness compliant device will reduce the manipulator operational speed and operation itself by making the effect of friction too sensitive. However, higher stiffness will affect the manipulator dynamics and contact stability. In this section we will discuss various approaches for robotic force control with the manipulator dynamics, together with contact dynamics, being taken into account.

In Chapter 4 it was supposed that the control scheme is in the robot's joint space, or inner-loop of the robot control system. The control outputs are joint positions, velocities and accelerations and control inputs of the joint control scheme are the joint forces/torques. A joint-space control scheme does not consider the overall robot end-effector's path and velocities. The actual position, path control of the robot manipulator's overall motions are performed by the supervisory computer by transforming the operational (task) space trajectory into robot's joint space for the joint inner-loop servo. In this sense, force control is in operational space.

### 5.5.2 Some Simulation Results and Analysis

The manipulator described in section 4.3 of Chapter 4 is used for simulation of force control. As shown in Fig. 5.7 the manipulator's trajectory is constrained in the x direction. The desired working condition is defined as

to move the manipulator in the vertical line maintaining a constant force normal to the surface. In this simulation no friction is considered. A piece of MODULA-2 program is listed in Appendix 5.1 for this simulation.

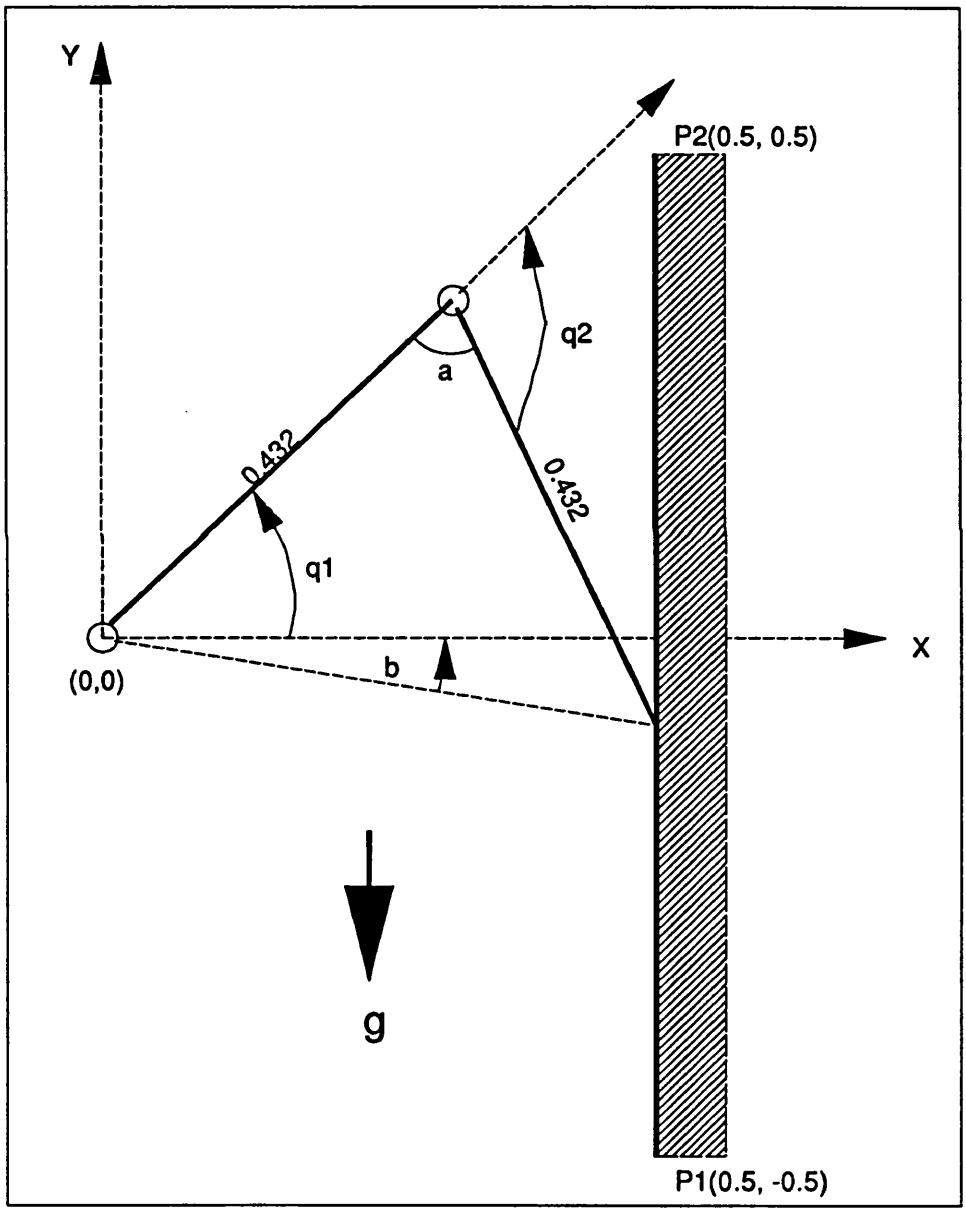



Fig. 5.7 Model for force control simulation

Similar hybrid force/position control suggested by Raibert and Craig [1981] is used for the simulation. Constant PID controller for joint servo was used, and the controller's gains are decided by the simulation of Chapter 4, which has resulted in a constant PID controller by letting the forgetting factor be slightly bigger than one. Sampling period is chosen as T=0.005, and desired force is set to be 10 Newtons.

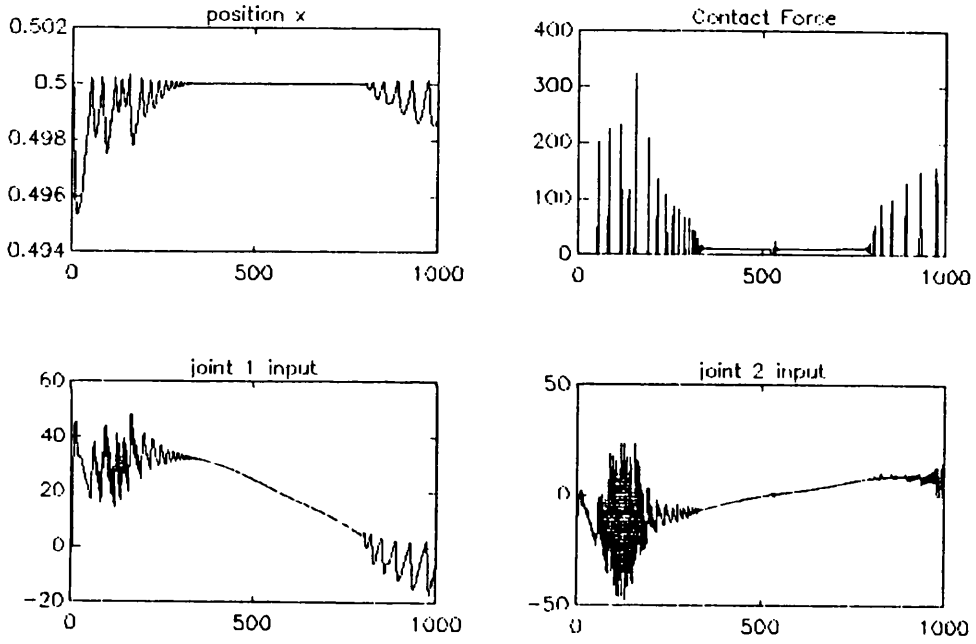



**Fig. 5.8 Stiff environment, $K_e = 10^6$**

Fig. 5.8 shows results of a stiff environment, with stiffness $K_e = 10^6$. As can be seen, the manipulator bounces against the environment uncontrollably. An and Hollerbach [1989] had a thorough analysis of hybrid force/position control. According to their results, hybrid control can only control polar manipulators as the successful control in Raibert and Craig's [1982] original paper. For revolute manipulators, instability is an inherit characteristic

because of the calculation of inverse Jacobean matrix. An and Hollerbach [1989] thus used $\hat{a}$ dynamic model for feedback compensation, formed the so called resolving acceleration control.

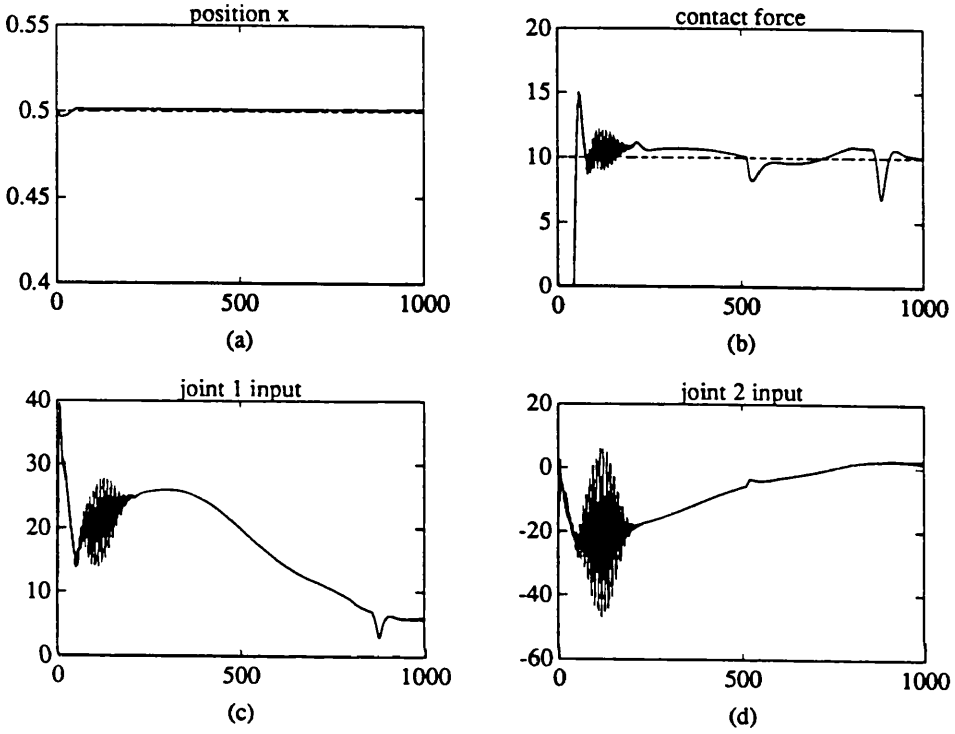



**Fig. 5.9 Less stiff environment, $K_e = 10^5$**

Fig. 5.9 shows simulation results of a less stiff environment. As can be seen, the stability problem is much improved. The contact can be controlled within reasonable range. From plot (a), it can be seen that the actual position in **x** direction slightly larger than 0.5 m, the desired one. That is, force is maintained by compress the environment.

However, as shown in Fig. 5.10, if the environment becomes more flexible, the constant contact can not be maintained, resulting in a sluggish control effort. This shows the limitation of hybrid force/position control. From plot (a), it can be seen that force control demands the arm move 'inside/into' the environment. However, position control demands the arm to be positioned at x=0.5 as pre-planned resulting in a fluctuating contact force as in plot (b).

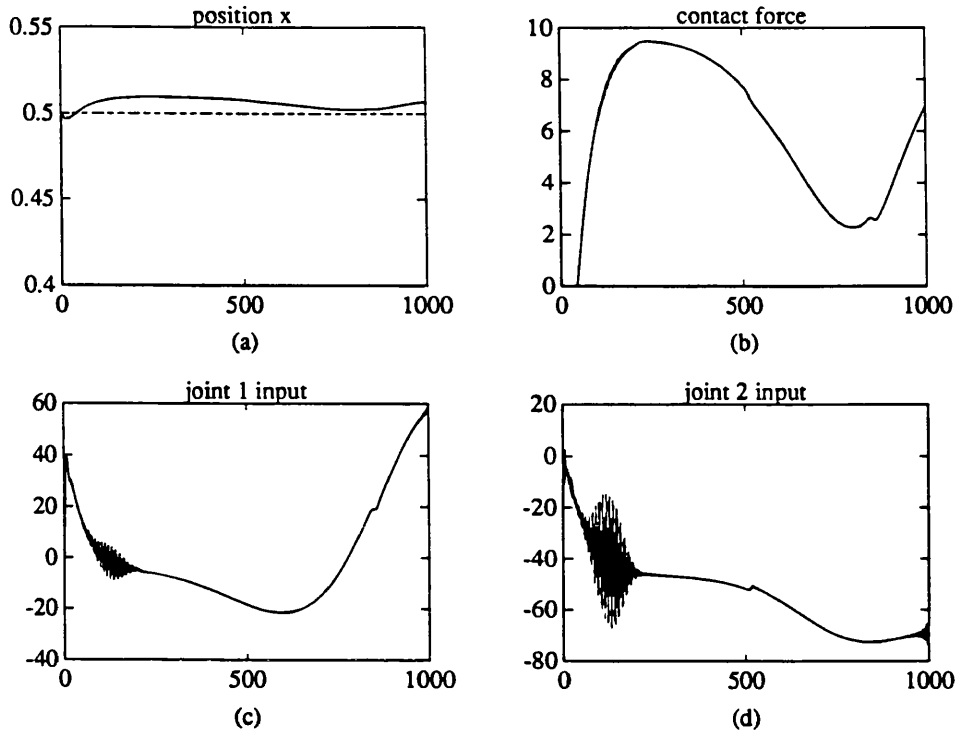



**Fig. 5.10 Flexible environment, $K_e = 10^4$**

### 5.5.3 Final Remarks

The main joint position servo controllers are of P(I)D structure with gains being adjusted by using self-tuning adaptive strategy as discussed in Chapter 4. Force controller is mixed up with the position servo in each joint. When there is no force demand from environment, only the position controller of each joint takes effect, while if there is contact force demand from the robotic end-effector, force controller, joining with position controller, controls the robotic manipulator to accomplish pre-specified task.

In Chapter 1, detailed description of various force control schemes are reviewed. Here, only hybrid force/position control is simulated. From the simulation results, it can be concluded that hybrid force/position control is based on the assumption that the environment is infinitely stiff with no compliances either in the arm or in the environment. This restricts wide

application of this scheme, especially when flexible arm is attracting more and more interests. Improvement has to be placed on this scheme. Hybrid impedance control, as described in the introduction chapter, may be a solution.

For a rigid arm and a rigid environment, stability of hybrid force/position control is still an unsolved problem.

## 5.6 SUMMARY

This chapter has described robotic compliant motion control. The first few sections detailed compliant motion control without considering the manipulator dynamics itself, and the compliances are from a compliant end-effector. This is the easiest impedance control, the impedance is from a compliant device rather than attempting to control the compliance of the whole manipulator. Contact dynamics are discussed in detail. Surface contact is of especial importance for robotic sub-sea inspection as can be seen from Chapter 7. This kind of research is very useful as the computation power is not high enough at this stage.

A compliant device is necessary to absorb the impact energy when the robotic manipulator is moving from free space to the point where contact is being made (Zheng [1991]). The compliant device can function to protect the precise and expensive force sensor and the arm itself. The contact is a short transition from the position control in free space to the force control in a constrained sub-space, and the manipulator is stopped from moving at some velocity, so kinematic energy must be transferred to other damaging forms. In an unstructured environment, collision of a robot with any obstacle is unpredictable. This is particularly true in the case of sub-sea inspection where other sensors like vision and sonar are detuned by various sub-sea operation conditions.

Impact effect can be solved by carefully preparing the operational environment. Also, as computers are becoming more powerful and inexpensive, fast sampling is available without computational delay. More

sophisticated force controller can then be implemented for various compliant control for robotic manipulator doing contact tasks. A very straight forward method to improve force control is to increase the computer power. Take PUMA560 for an example, the present PDP-11/73 processor can make supervisory control over the six 6503 micro processors in every 28 ms. If this supervisory computer can be more powerful, then quicker sampling rate can be applied, thus the compliant device can be removed or at least its stiffness can be increased for some operations where this is necessary.

## Appendix 5.1 MODULA-2 Program for Force Control Simulation

```
(* 2 DOF nonlinear, coupled force control Aug. 1991*)
(*$o+*)
MODULE ForCtrl;
IMPORT IO, FIO, MATHLIB, MathLib0, Lib;
(*-------------------------------------------------------------*)
PROCEDURE InverseKine(t, xf : REAL); (* xf is the position adjustment
                        for force control 11/7/91 *)
VAR
 y2, x2, alpha, a1, a2 : REAL;
BEGIN
 x2:= 0.5 + xf;  y2:= -0.5;
 IF (t<1.0) THEN  y2:= -0.5 + 0.25*t*t/2.0;
 ELSIF (t>=1.0) AND (t<4.0) THEN  y2:= -0.375 + 0.25*(t-1.0);
 ELSIF (t>=4.0) THEN y2:= 0.375 + 0.25*(t-4.0) - 0.25*(t-4.0)*(t-4.0)/2.0;
 END;
 a1:= 2.0*0.432*0.432-x2*x2-y2*y2;
 a2:= 2.0*0.432*0.432;
 alpha:= VecLab.acos2(a1, a2);
 r4:= - 3.1416 + alpha;
 r:= VecLab.atan2(y2, x2) - r4/2.0;
END InverseKine;
(*-------------------------------------------------------------*)
PROCEDURE ContactForce(Ya, Yb, Ke : REAL) : TWOREAL;
VAR
 x2, Fx, Fy : REAL;
BEGIN
 x2:= 0.432*MathLib0.cos(Ya) + 0.432*MathLib0.cos(Yb+Ya);
 Fx:= Ke*(x2-0.5); (* Ke is the stiffness of evironment. 11/7/91 *)
 IF (Fx<0.0) THEN
   Fx:= 0.0;
 END;
 Fy:= 0.1*Fx;
 Fy:= 0.0; (*No friction 15/7/91*)
 ContForce[1]:= Fx;
 ContForce[2]:= Fy;
 RETURN ContForce;
END ContactForce;
(*-------------------------------------------------------------*)
PROCEDURE ContactForceOnActuators(Ya, Yb, Fd: REAL); (*Jacobian transfer*)
```

```
VAR
 tau1, tau2 : REAL;
BEGIN
 tau1:= (-0.432*MathLib0.sin(Ya)
      - 0.432*MathLib0.sin(Yb+Ya))*(Fd-ContForce[1])
      +(0.432*MathLib0.cos(Ya) + 0.432*MathLib0.cos(Yb+Ya))*ContForce[2];
 tau2:= -0.432*MathLib0.sin(Yb+Ya)*(Fd-ContForce[1])
      +0.432*MathLib0.cos(Yb+Ya)*ContForce[2];
 ReactTau[1]:= tau1;
 ReactTau[2]:= tau2;
END ContactForceOnActuators;
(*-------------------------------------------------------------*)

(*Main Programme*)
BEGIN
 ... ...
 i:= 1;
 REPEAT
  t:= REAL(i)*T;
  r3:= r2;   r2:= r1;   r1:= r;
  r7:= r6;   r6:= r5;   r5:= r4;
  InverseKine(t, 0.0);
  (*use constant PID controller*)
  s30:= -0.4958;  s3:= 26400.0; s4:= -37531.0; s5:= 12354.0;
  s00:= -0.4677;  s0:= 32645.0; s1:= -45749.0; s2:= 14754.0;
  u23:= u22; u22:= u21;
  u21:= -s30*u23+s3*(r4-Yb)+s4*(r4-Y4)+s5*(r4-Y5);
  u13:= u12; u12:= u11;
  u11:= -s00*u13+s0*(r-Ya)+s1*(r-Y1)+s2*(r-Y2);
  (*Bound input here 10/4/91*)
  IF (u21>100.0)  THEN u21:= 100.0;  END;
  IF (u21<-100.0) THEN u21:= -100.0; END;
  IF (u11>100.0) THEN  u11:= 100.0;  END;
  IF (u11<-100.0) THEN u11:= -100.0; END;
  ContactForc:= Motion.ContactForce(Ya, Yb, Ke);
  ContactForceOnActuators(Ya, Yb, Fd);
  DesiredContactForceOnActuators(Ya, Yb, Fd);
  (* First try non force control 12/7/91 *)
  Tauf[1]:= Tauf[1] + 0.1/b1*ReactTau[1] ;
  Tauf[2]:= Tauf[2] + 0.1/b2*ReactTau[2] ;
  State_Var:= Resp2F.Rung_Kut(State_Var,u11+Tauf[1]+DesiredReactTau[1],
                  u21+Tauf[2]+DesiredReactTau[2], i);
  Y2:= Y1; Y1:= Ya; Y5:= Y4; Y4:= Yb;
  Ya:= State_Var[1];
  Yb:= State_Var[3];
  i:= i + 1;
 UNTIL (i > INTEGER(5.0/T));
... ...
END ForCtrl.
```

# CHAPTER 6
# COMPLIANT CONTROL FOR ROBOTIC ASSEMBLY

## 6.1 INTRODUCTION

Industrial applications involving force feedback are rare. This is in sharp contrast with the growing penetration of vision systems. The main reason is that force sensors are difficult to incorporate into commercially available robot controllers. Although a vision system is more complex than a force sensor, its interaction with the robot controller is mainly geometrical. It only influences target co-ordinates. These data are easily understood by traditional controllers provided with suitable digital inputs. Force sensor data, however, influences the dynamics of the joint servo loops. Most commercially available robot controllers do not allow such low level intervention.

As has been described in the introduction chapter, most commercially available robot manipulators solely function as position control devices with no means of directly controlling the contact force between the robot's end effector and the environment. In a number of important manufacturing applications, in particular deburring and assembly etc., control of the interacting forces is critical.

Global competition forces manufacturing industries either to die or to automate the production processes. This is why Computer Aided Manufacturing (CAM), Computer Incorporated Manufacturing (CIM) and robotic assembly have quickly developed recently (Besant [1986]). The use of robots in industries has increased rapidly in the last decade, apart from being employed economically in activities like spot and arc welding, paint spraying and material handling, other novel applications such as inspection and maintenance are being developed to complete with traditional methods.

Probably the most versatile commercially-available assembly robot is the anthropomorphic PUMA (Programmable Universal Machine for Assembly), which has six revolute joints. PUMA type robotic manipulators provides a maximum of six degrees of freedoms, and are able to reach any position/orientation within their working space. They are capable of performing insertions from virtually all directions. However, one frequent criticism of the PUMA is, ironically, its over versatile geometrical configuration, which is functionally redundant for many assembly tasks, since over 80 percent of these tasks are in orthogonal directions. To control such versatility, not only does the robot need a more powerful control system to compute its co-ordinate transformations in real-time, but the arm also becomes less rigid and repeatable, especially when it is fully stretched and loaded. Although PUMA is not the best choice for assembly, six DOF is necessary for sub-sea inspection where the environment is unknown a priori and the manipulator has to be versatile to carry out three dimensional inspection operations, as will be discussed in the next chapter.

The features of SCARA type robot, Adept One, which has four degrees of freedom and is dedicated for vertical insertion assembly, are compared with PUMA560 as shown in Table 6-1.

Both robots use VAL-2, which perhaps represents the state of the art of manipulator level language. Apart from providing three levels of commands

to control the robot's operating system, interactive ability and end-effector movements, its software is structured to process most of the operating commands (including those for controlling the status of the system, defining the robot locations, storing and retrieving information on floppy diskette and creating and editing robot control programs) in parallel to user-program executions.

**Table 6-1 Comparison between Adept One and PUMA560**

| | Adept One | PUMA560 |
|---|---|---|
| Payload | 6 kg | 2.3 kg |
| Velocity (tip, maximum) | 30 m/s | 1 m/s |
| Acceleration (tip, maximum) | 5 g | 1 g |
| Repeatability | 0.05 mm | 0.1 mm |

This chapter discusses robotic assembly using active force feedback instead of mechanical method (passive). The experimental task for testing the force control for robotic assembly strategies is to make a robotic arm put a cylindrical peg into a cylindrical hole. Section 6.2 describes the formalism of this task and identifies the natural constraints defined by Mason [1981]. Section 6.3 will be devoted to trajectory planning to move the robotic manipulator end-effector from any initial position/orientation to the destination. A PASCAL program package running on a VAX workstation is described in section 6.4 for simulating putting a peg into a hole. In section 6.5, the real-time experimental work carried out at University College London is described. The author has succeeded in putting a cylindrical peg into a cylindrical hole with maximum clearances of 0.02 mm. In the last section, some general remarks and drawbacks of this active robotic assembly are given.

## 6.2 TASK FORMALISM FOR ROBOTIC ASSEMBLY

In force control, the position controlled directions and force controlled direction are orthogonal and complementary with each other. The process in deciding which directions are position controlled, which directions are force controlled directions is called task formalism. The two phases should be treated separately, one being gross motion control and the other fine motion control. In the first phase, contact has not been established and only position control is involved. However, in the second phase force and position are controlled together and in this sense force control is also called hybrid control.

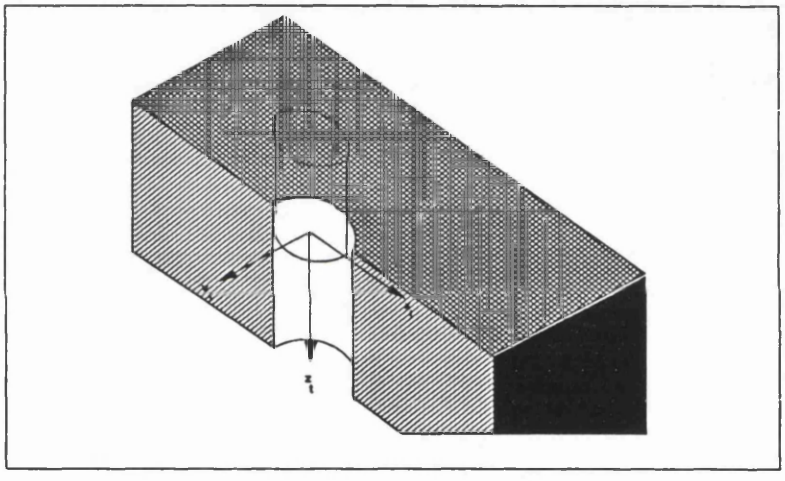


**Fig. 6.1 Task formalism of peg-to-hole**

As will be shown, the control strategy used in this project is actually impedance control, i.e. force control is achieved by adjusting the position/orientation of the manipulator end-effector. Task formalism is not so crucial as hybrid force control as described in Chapter 5. However, VAL-2 uses the command ALTOUT for directions which need real-time path modification and command MOVES for position/orientation control. So, task formalism becomes much easier as these two groups are simply divided without considering compliance at the contact point and contact friction. That is, force-controlled directions and position-controlled directions are defined assuming a desired working condition (no contact compliance and no friction in the contact point).

As shown in Fig. 6.1, in the tool co-ordinate, which is original attached at the flange of the robotic end-effector, is attached at the tip of the peg by simply re-define TOOL. **x**, **y**, $\mathbf{r}_x$ and $\mathbf{r}_y$ are force-controlled directions, whereas **z** and $\mathbf{r}_z$ are position-controlled directions for this assembly task.

## 6.3 TRAJECTORY PLANNING

In this section, various trajectory planning methods are suggested. It is recommended that the position trajectory is planned in Cartesian space when there is a constraint, whereas the orientation trajectory planning can usually be planned in joint space. So the separation of a robotic manipulator into an arm and a hand for kinematic and dynamic analysis described in Chapter 2 can also be used in the trajectory planning. As it is usually the case that geometrical constraint of manipulator constrains the position movement of a robotic manipulator. Cartesian space and joint space trajectory planning methods are given in the following sub-sections.

### 6.3.1 Introduction

Robot manipulator motion control is mainly dependant on properly planning the trajectory. It can be considered that trajectory planning has the same importance as arm dynamics and interacting with the world. Especially in force control, fine trajectory planning --- so called task formalism --- is of essential importance. Computer graphics can play an important role in trajectory planning as this is mainly a geometrical exercise.

Basically there are two approaches used for planning a manipulator trajectory, namely trajectory planning in the joint space or trajectory in Cartesian space. The first approach requires the user to specify explicitly a set of constraints (e.g., continuity and smoothness ) on position, velocity and acceleration of the manipulator's generalized co-ordinates at selected locations (called knot points or interpolation points) along the trajectory. A parameterized trajectory

is selected from a class of polynomial functions that interpolates and satisfies the constraints at the knot points. Since no constraints are imposed on the manipulator's hand, it is difficult to trace the path of the hand and guarantee it to be a collision-free motion in a complex environment. Therefore, this method is only suitable for gross motion trajectory planning. In fine motion trajectory planning, as when the robot manipulator's end effector makes contact with the environment, the second method should be used.

In the second approach, the path of the end effector is specified explicitly as a straight line or a circle with the path constraints in Cartesian co-ordinates. The desired trajectory can be then computed in either (a) joint co-ordinates or (b) Cartesian co-ordinates which is quite often dictated by the subsequent control algorithm (Cartesian space control or joint space control) to ensure the desired path tracking. For Cartesian space-oriented planning (b), the time history of the manipulator hand's position, velocity, and acceleration are planned. This is suitable for the fine motion trajectory planning.

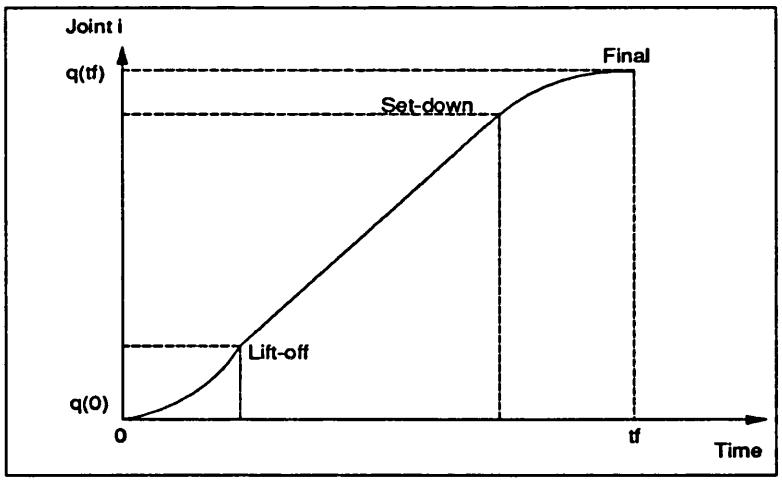



**Fig. 6.2 Position displacement of a joint trajectory**

Cartesian space path planning requires transformations between the Cartesian and joint co-ordinates in real time. This is a task that is

computational intensive and quite often leads to longer control intervals (Mon [1988]). In the case of the PUMA560, both methods can be used and the transformations are performed within the PUMA controller.

From the control analysis point of view, the movement of a robot arm is usually accomplished by two distinct control phases. The first is gross motion control in which the arm moves from an initial position/orientation to the vicinity of the desired target position/orientation along a planned trajectory. The second is the fine motion control in which the end-effector of the arm dynamically interacts with the objects/environment using sensory feedback information to complete the task. Fig. 6.3 shows computer graphic representation of the robotic manipulator and the assembly hole, i.e. the destination specified (in Cartesian co-ordinate with origin at the robot's base point).

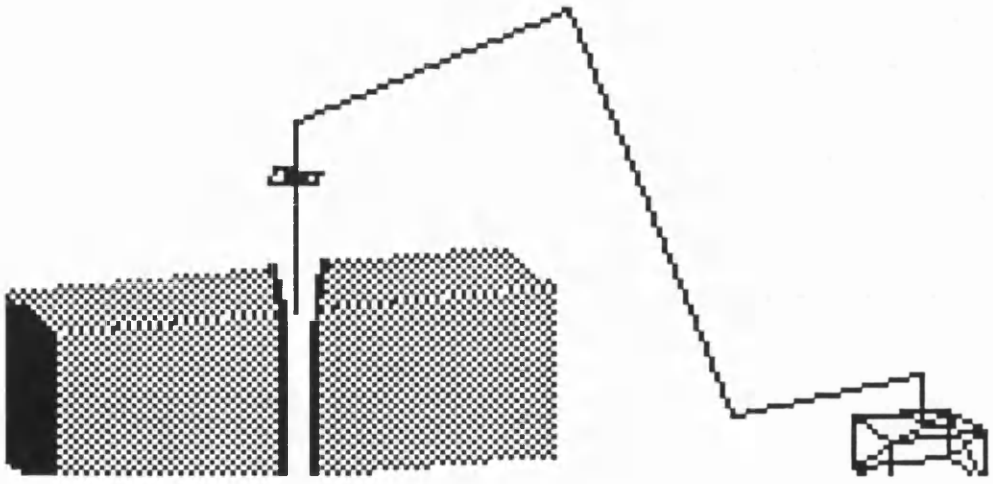


**Fig. 6.3 The vicinity of PUMA560 gross motion**

The initial position of the robot manipulator is specified by a state vector:

$$[q_1(0), q_2(0), \ldots, q_6(0)] \qquad (6\text{-}1)$$

Another state vector specifies the final position and orientation of the end-effector of a robot manipulator (e.g. x= -111 mm; y = 83 mm; z = 38 mm; $\mathbf{x}_t$ = (1, 0, 0); $\mathbf{y}_t$ = (0, 1, 0); $\mathbf{z}_t$ = (0, 0, -1) ):

$$[q_1(t_f), q_2(t_f), \ldots, q_6(t_f)] \tag{6-2}$$

The displacements of each joint can be decided by subtracting (6-1) from (6-2). Each joint can then move according to trajectory planning in a joint-interpolated trajectory as shown in Fig. 6.2.

### 6.3.2 Joint-interpolated Trajectory

Mon [1988] provides a good summary of joint-interpolated motion trajectory planning. There is, however, a mistake in the polynomial formulations for the 4-3-4 joint trajectory which originated from Fu K.'s [1987] textbook of robotics. The formula from pp166, Table 4.3, second trajectory segment of Fu K.'s textbook is rewritten as following:

$$h_2(t) = (\delta_2 - v_1 t_2 - a_1 t_2^2/2)t^3 + (a_1 t_2^2/2)t^2 + (v_1 t_2)t + \theta_1$$
$$v_2 = \dot{h}_2(1)/t_2 = 3\delta_2/t_2 - 2v_1 - a_1 t_2/2$$
$$a_2 = \ddot{h}_2(1)/t_2^2 = 6\delta_2/t_2^2 - 6v_1/t_2 - 2a_1 t_2$$

By dimensional analysis, it can be shown that the last term of the third equation has a extra time component $t_2$. From the graphics display, it can be seen that in the first segment and the second segment, the robot manipulator moves smoothly and correctly except the third segment, where it moved rapidly. Table 6.2 is obtained according to Fu K. [1987] pp166 Table 4.3.

**Table 6.2 Polynomial equations for 4-3-4 joint trajectory**

| First trajectory segment |
|---|
| Define: $\delta_1 = [q(t_f) - q(0)]/8$<br>Position: $h_1(t) = (\delta_1 - \sigma)t^4 + \sigma t^3 + q(0)$<br><br>Velocity: $v_1 = \dot{h}_1(1)/t_1 = 4\delta_1/t_1 - \sigma/t_1$<br>Acceleration: $a_1 = \ddot{h}_1(1)/t_1^2 = 12\delta_1/t_1^2 - 6\sigma/t_1^2$ |

| Second trajectory segment |
|---|
| Define: $\delta_2 = 5\delta_1$<br>Position: $h_2(t) = (\delta_2 - v_1 t_2 - a_1 t_2^2/2)t^3 + (a_1 t_2^2/2)t^2 + (v_1 t_2)t + q(t_1)$<br><br>Velocity: $v_2 = \dot{h}_2(1)/t_2 = 3\delta_2/t_2 - 2v_1 - a_1 t_2/2$<br>Acceleration: $a_2 = \ddot{h}_2(1)/t_2^2 = 6\delta_2/t_2^2 - 6v_1/t_2 - 2a_1$ |

| Third trajectory segment |
|---|
| Define: $\delta_3 = \delta_1$<br>Position: $h_3(t) = (9\delta_3 - 4v_2 t_3 - a_2 t_3^2/2)t^4 + (-8\delta_3 + 3v_2 t_3)t^3$<br>$+(a_2 t_3^2/2)t^2 + (v_2 t_3)t + q(t_2)$ |

Where: $\sigma = f/g$

$$f = 2\delta_1(4 + 2t_3/t_2 + 2t_3/t_1 + 3t_2/t_1) - \delta_2 t_1/t_2(3 + t_3/t_2) + 2\delta_3 t_1/t_3$$

$$g = t_3/t_2 + 2t_3/t_1 + 2 + 3t_2/t_1$$

$$t = (t_r - \sum_{j=1}^{i-1} t_j)/t_i$$

t : the normalized time variable, belong to [0,1].

$t_i$ : real time required to travel through the $i$th segment.

$t_r$ : real time at the end of the $i$th trajectory segment and real time in seconds respectively.

In the discrete form the above equation must be discrete. The sampling period T = 0.028 second in the PUMA560 case. Then $t_1 = 64T$ ; $t_2 = 192T$ ; $t_3 = 64T$. So, in the first segment $t = (j \cdot T - 0)/t_1$ in the second segment $t = (j \cdot T - 64T)/t_2$ and the last segment $t = (j \cdot T - 256T)/t_3$

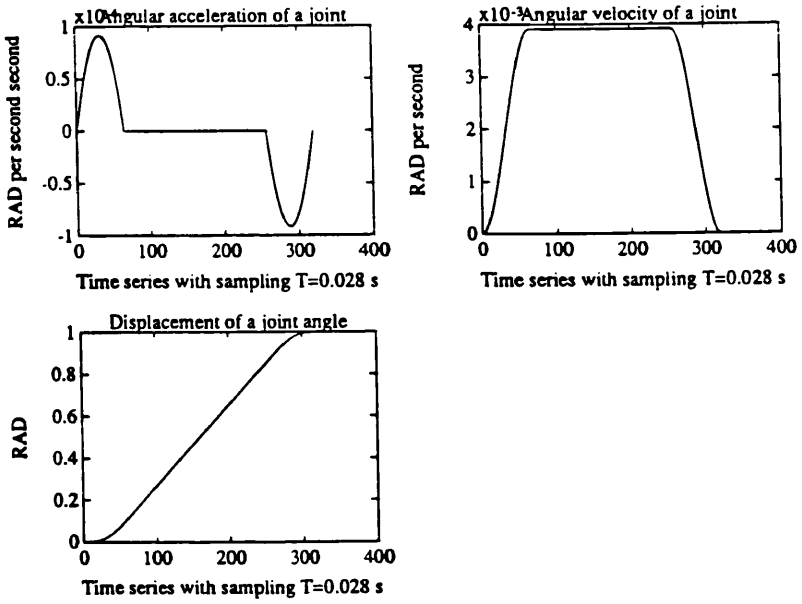



**Fig. 6.4 Joint acceleration, velocity and displacement profiles for joint-interpolated trajectory planning**

Fig. 6.4 are draw using MATLAB according to the calculation of Table 6.2. Only one joint is given, the others have the same profiles (displacement, velocity and acceleration) with different scales.

### 6.3.3 Fine Motion Trajectory Planning

After the end effector of the robot manipulator has been moved into the vicinity of the environment with which it is to interact, the task has changed to fine motion planning. Two group of figures need to be specified in this case: one is the position of the end effector, the other is the orientation of the hand. Using inverse kinetics, each joint's displacement can then be calculated. In PUMA560, VAL-2 does the inverse kinematics in Unimate's protocol. It takes 28 ms for every position command to pass to the joint servo.

In a dedicated assembly, the task frame is fixed relatively to the world co-ordinate. For example, in the vertical insertion of a peg into a vertical hole, the task frame has three axis parallel to the fixed world co-ordinate. The difference is that the origin of the frame is moving in the vertical direction. Because of this, the definition of task frame becomes much easier with the co-ordinate fixed to the tip of the peg. Motion planning can be accomplished referring to world co-ordinates (Cartesian space) instead of tool co-ordinates, or joint space as described in sub-section 6.3.2. More discussion will take place later in this chapter.

## 6.4 SIMULATION PROGRAM

A PASCAL program for simulating the process of robotic assembly has been written on a VAX work station. The robotic manipulator is located in a random initial position and orientation, it is supposed that the robotic manipulator can move to the destination and put the peg into the hole. Two phases are distinguished in the operation: gross motion and fine motion control as discussed in the former section. The PASCAL simulation programs were later translated to MODULA-2 on the IBM PC (a simply description is appended at end of this chapter Appendix 6.3) as a part of the PUMA560 supervisory control program package.

### 6.4.1 General View of the Program

**Table 6.3 Description of a simulation program package**

| *Packages* | XINGYUAN<br>MMAIN | VecLib | VtLib<br>PltLib<br>Chario | Mview11<br>Mview22<br>mval | Mmarm<br>Mforce | PumaDraw<br>SolPuma |
|---|---|---|---|---|---|---|
| *Function* | Man machine Dialogue | Mathe-matics library | Packages to drive the work- station to draw | Packages to get cubic view effect | Arm kine-matics and trajectory planning | Drawing a graphic PUMA560 |

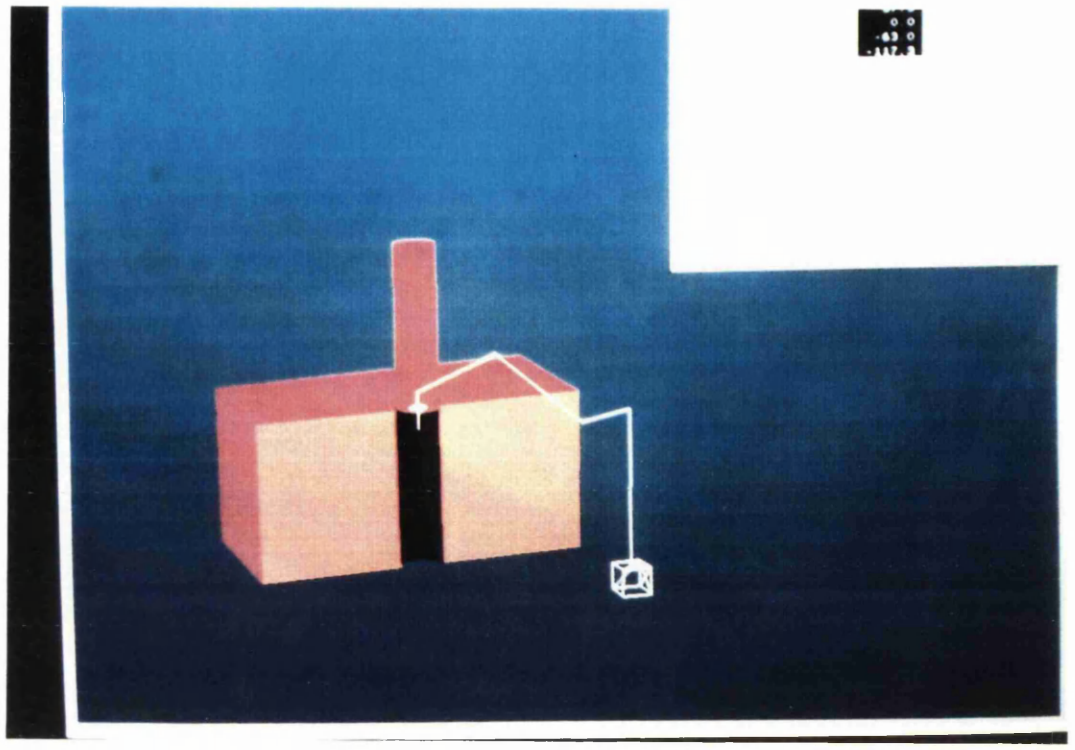


**Fig. 6.5 Picture of the VAX workstation simulation package**

The programs have a total of over 200 pages listing on the VAX printer (370 mm X 280 mm) including explanation lines. There are a total of 12 packages, which can be grouped into 6 larger groups according to their functions simply shown in Table 6.3.

Each of the 12 packages contains a number of procedures which are not going to be described in great detail. All the simulation programs are based on kinematics --- JOINT, TOOL, WORLD moving of PUMA560 trajectory planning and following using graphic displaying on the work station. There is no dynamic consideration. As the screen is of only two dimensions, the actual cubic effect is simulated by clipping a quadrilateral against a 3D viewing volume and carry out a perspective transformation (in package *Mview11*, *Mview22* and *mval*). The package *Mforce* is for doing trajectory control described in 6.3.2.

Fig. 6.5 is a picture taken to show the computer graphics for simulating robot motion.

### 6.4.2 Simulation of Gross Motion

As shown in Fig. 6.6 and 6.7, the robotic manipulator should be able to move from its any initial position to the destination specified beforehand, according to a certain trajectory in joint space as discussed in the former section (section 6.3).

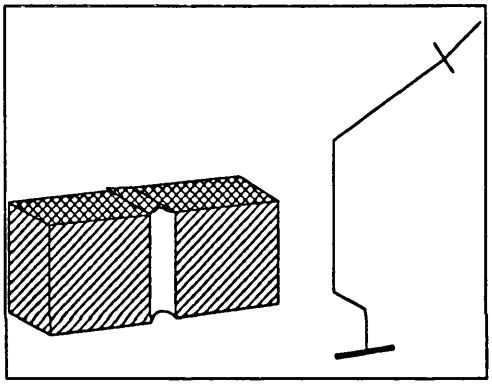


**Fig. 6.6 Drawing a initial position of manipulator**

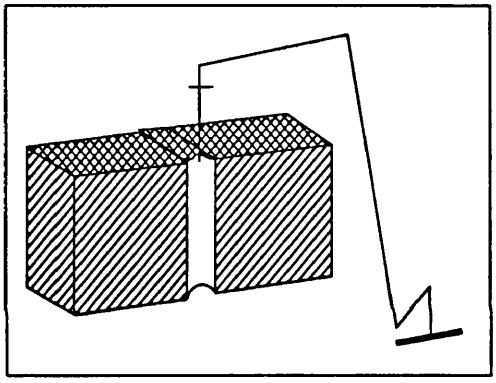


**Fig. 6.7 Drawing a vicinity of destination of manipulator**

In this simulation, the gross motion of the robotic manipulator moves alone the joint interpolated trajectory planning, without concerning the actual trajectory of end-effector of manipulator in the Cartesian space. The underlying assumption is that there is not an obstacle present. The robot records the initial six joint angles, and by inverse kinematics it can calculate another six joint angles in the vicinity of destination. When both sets of joint angles are acquired, joint space trajectory planning described in section 6.3 can be applied to find the movement of the robot manipulator in joint space.

The advantage of joint space trajectory planning is that the dynamic response remains the best and the robotic manipulator moves smoothly with no inverse

kinematic calculation during the motion. The disadvantage is that the actual Cartesian trajectory is not taken into account. The specific task can be achieved by fine motion control as will be seen in the next sub-section.

### 6.4.3 Simulation of Fine Motion (Contact Phase)

When a real task is specified to the robotic manipulator, fine motion trajectory planning has then to be used to plan the robotic manipulator to move at user's demand. Examples are numerous, and putting a peg into hole is one of them. The manipulator is required to move in a straight line in Cartesian space. Joint space trajectory planning can not meet this requirement.

In the simulation package no dynamics were taken into account. The trajectory is wholly dependant on the inverse kinematics, resulting in a purely geometrical problem. Of course, there is no force sensor being able to be present in the simulation. This makes the problem quite non-straight forward, as if it is a real-time problem, force signals are then continually measured.

If the arm dynamics are taken into account in simulation, then the contact force can be calculated by the desired position and the practical position of the robotic manipulator, which is always sightly different from the demanded one as the controller can not make the robotic manipulator track the desired trajectory with no error. The position error can then be used to calculate the contact force by multiplying it by the stiffness, as it had already been practiced in the simulation present in section 5.5 of Chapter 5.

However, as said before, in this simulation on a VAX work station, no dynamics are taken into account (dynamic simulation is separated from this package as already described in Chapter 3 and 4). And inverse kinematics can very precisely calculate the robotic manipulator position and joint angles, i.e. no position error will arise. So, the reaction force is generated by a random generator. The purpose is to test to see if the program can respond to the reaction force and make subsequent adjustment to the contact force/torques.

## 6.5 DESIGN OF A COMPLIANT END-EFFECTOR

The original reason for using a compliant device is that a compliant flexible device is necessary to provide contact stability. Computation power is not yet high (or inexpensive) enough and contact stability problems are still encountered in contact task operations. Contact stability is the basic requirement for any contact operation, as described in Chapter 5.

For sub-sea inspection manipulation, two surfaces are usually needed to make full contact for the probe to carry out the inspection job properly. Contact should be normal to the inspecting surface all the time. It was decided to make a three degree of freedom compliant device. The device has one contact compliance and two twist compliances, which will provide the necessary compliances for a stable normal contact. The real-time experiments described in this chapter and in Chapter 7 use this compliant device, whose structure and features are described here.

### 6.5.1 Mechanical Structure

A set of technical drawings is attached at Appendix 6.4. The first drawing shows the general assembly. There are four elements, which are drown individually. All dimensions are in millimetres, and most of the material used was aluminium alloy with the one exception being the brass bolt for better mating of parts. The distance between two faced position sensors is 110 mm. The four position sensors are from "*RS Component*", and others are made at University College London.

This compliant device was designed to be assembled between the robotic end-effector and the work environment. Three dimensional compliances were originally chosen for the device, one contact and two twists. Fig. 6.8 shows the structure. There is a ball joint at the centre of the contacting surface, which can provide the two rotational compliances. The axial or contact compliance

is obtained from the translation joint as shown in the figure. The range of rotation of the ball joint around the axis of the bolt is restricted by four small holes at the contact plate.

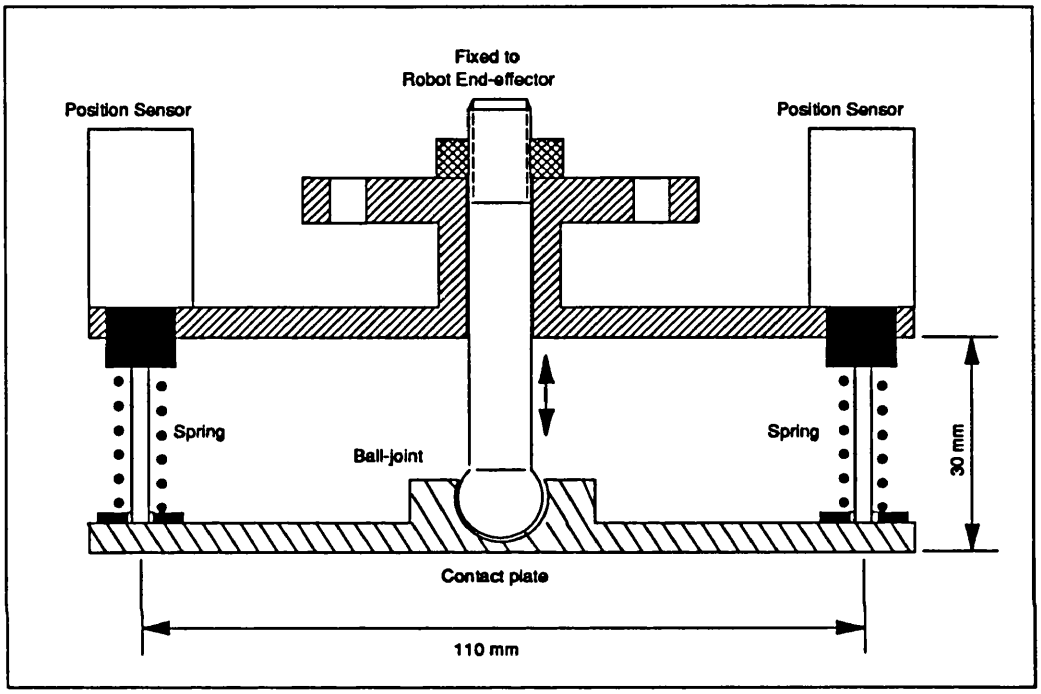



**Fig. 6.8 Structure of the compliant device**

The mechanical transitional displacement of the device is made to be 11 mm, slightly less than the position sensor's mechanical displacement limit. The maximum rotation displacements in both twist directions are limited by the mechanical restrictions of the spherical bearing, which is about ±5.25°.

There are four positional sensors, one at each corner of the device. Their features, electrical connections and calibrations are described in the following sub-sections. The positioning pins are wound by four springs to provide more stiffness. The probe, which does not appear in the figure but in the picture (Fig. 6.9), is attached to the contact plate.

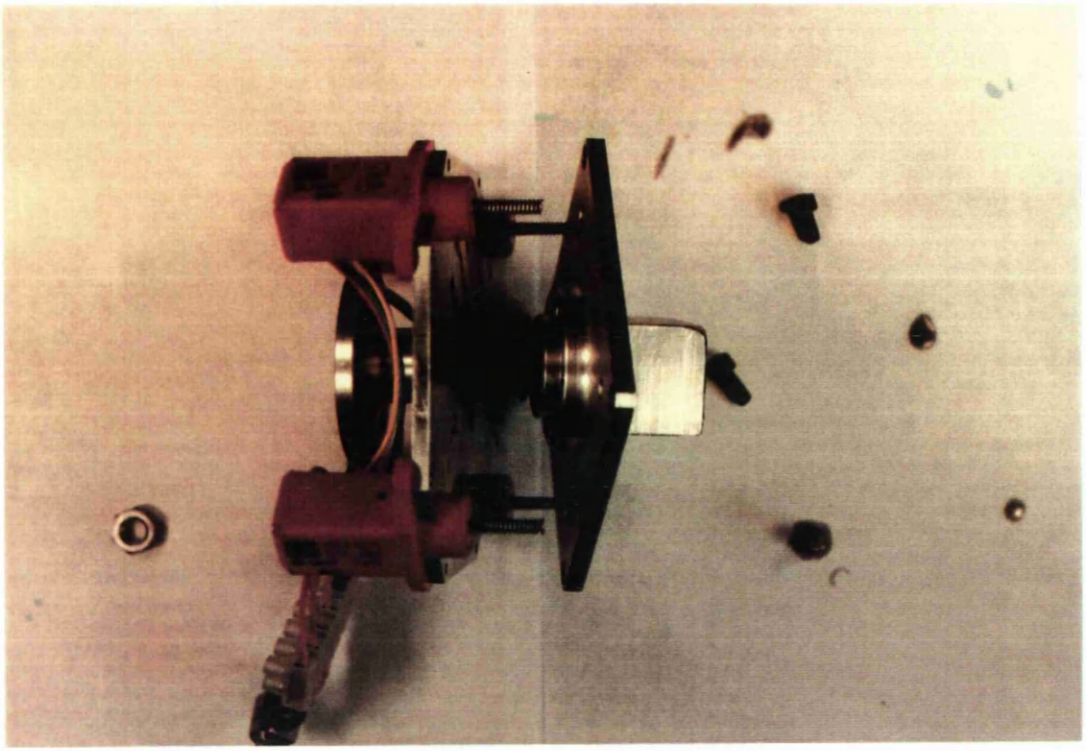


**Fig. 6.9 Photo to show the compliant device**

### 6.5.2 The Linear Position Sensor and Its Calibration

Position Sensor Description

The four point compliant device is made up of four linear position sensors whose technical specification is given in the following with the geometrical shape shown in Fig. 6.10.

This is an economical linear position sensor of 10 *mm* electrical stroke with a spring loaded plunger. Electrical connections to the 5*K* conductive polymer sensing element are by flying leads. For best results use as a potential divider. Mechanical fixing is by two holes enabling the sensor to be simply fixed using easily fabricated brackets. Applications include both hand and foot controls, valve and actuator position sensors, etc.

| | |
|---|---|
| Linearity | 2 percent |
| Electrical travel | 10 mm |
| Mechanical travel | 12.5 mm |
| Operating force | 2 to 7.5 N |
| Mechanical life | $5x10^6$ cycle |

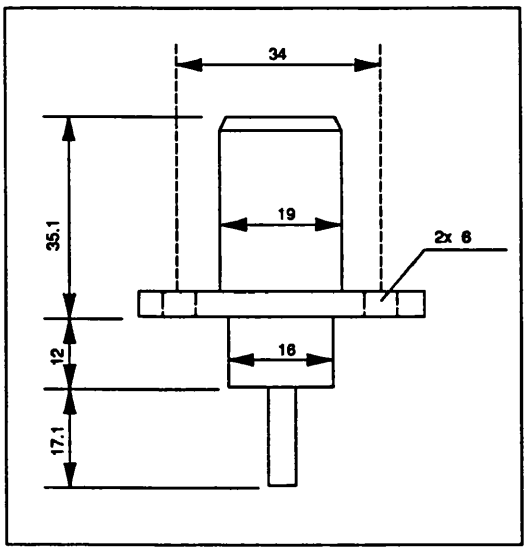



**Fig. 6.10 Linear position sensor**

Five volts is chosen as the supply voltage, so that the sensor's output would range from zero volts, if no displacement, to -5 *v* when fully pressed (or +5 *v* according to the reference chosen). In this range, the output is supposed to be linear, i.e. proportional to displacement. This linearity was further validated in the calibration process described below. There are two dead zones which have to be realized: 1). Initial small displacement does not trigger any output from the sensor; 2). Fully range output is reached and some mechanical displacement can still be achieved without any more increase in the output.

Sensor Calibration

To check the linearity and acquire the function relationship between displacement and output voltage of each sensor between the two dead zones, calibration was performed. The sensor was clamped in a vice and supplied with 5 volts. As each sensor was pressed, corresponding displacement and voltage output was measured. The reference voltage was chosen as the positive end of the constant power supply, so negative output was attained.

In order to make sure that the measurements were taken in the linear region, only the reading which are well away from the two dead zones were taken into account. The measuring data actually acquired is listed in Tab. 6.4

**Table 6.4 Sensor calibration data measured**

| | Displacement (in millimetres) | | | |
|---|---|---|---|---|
| **Voltages** | **Sensor1** | **Sensor2** | **Sensor3** | **sensor4** |
| -1.4 | 2.90 | 3.10 | 2.69 | 2.77 |
| -1.8 | 3.48 | 3.64 | 3.79 | 3.53 |
| -2.2 | 4.20 | 4.63 | 4.55 | 4.39 |
| -2.6 | 5.05 | 5.53 | 5.40 | 5.22 |
| -3.0 | 5.79 | 6.11 | 6.20 | 5.94 |
| -3.4 | 6.63 | 6.99 | 7.03 | 6.88 |
| -3.8 | 7.50 | 7.79 | 7.80 | 7.67 |
| -4.2 | 8.40 | 8.68 | 8.46 | 8.51 |
| -4.6 | 8.91 | 9.08 | 9.05 | 8.90 |

Using the calibration data listed in the above table, the relationship between mechanical displacement and output voltages can be derived in terms of a linear function for each sensor. These function equations are then used in software development stage to work out the displacement by reading the voltage output of each pot from A-to-D convertor.

The Least Square method is used to find the function equations. MATLAB was used to solve this problem by providing a simply easy to understand routine:

**polyfit(x,y,n)**

where, x variable is a data vector (e.g. voltages in the above table), y variable is another data vector (e.g. displacement) and n is the order of polynomial. This routine finds a polynomial $p$ such that $p(x)$ fits the data in the vector y in a least square sense. This is shown in the following figure.

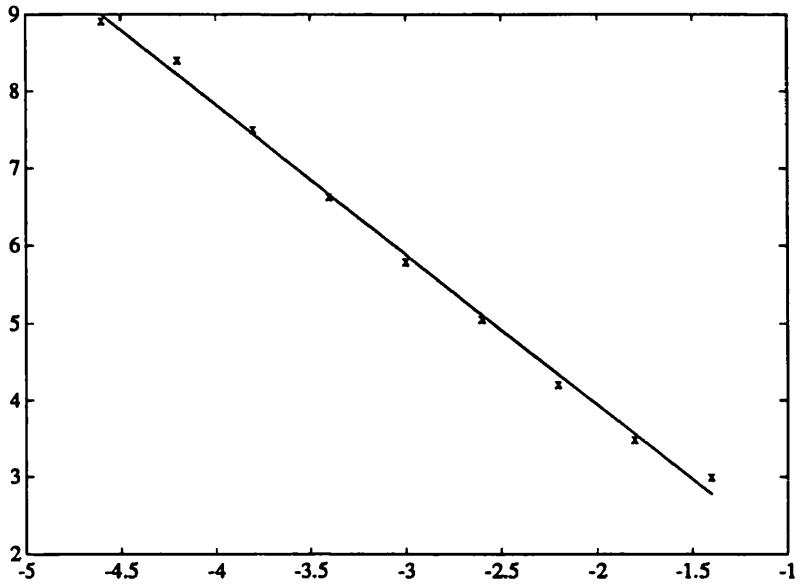



**Fig. 6.11 Data fitting**

The final functions for the four sensor were obtained using MATLAB as follows:

$$d_1 = -1.9575v_1 + 0.0008$$

$$d_2 = -1.9508v_2 + 0.3197$$

$$d_3 = -1.9825v_3 + 0.1603$$

$$d_4 = -1.9867v_4 + 0.0189 \qquad (6\text{-}3)$$

Where $d_i$ and $v_i$ are absolute displacement and voltage output of $i$th pot respectively. $v_i$ is read from the corresponding port of the A/D converter scaled by a factor $\frac{10}{2047}$ (Unimate [1985], pp.4-12). VAL-2 program to calculate the displacements of the four position sensors are listed in Appendix 6.2

### 6.5.3 Description of the Electrical Connections

An electrical circuit is required to process the signal coming from the sensors. The electrical connections are kept to a minimum by feeding all the connections through a 9 way D socket, with the pin connections shown below:

pin 1 ---------- output of sensor 1
pin 2 ---------- output of sensor 2
pin 3 ---------- output of sensor 3
pin 4 ---------- output of sensor 4

pin 7 ---------- ground
pin 8 ---------- + power supply
pin 9 ---------- - power supply

The sensor outputs are referred to the ground, which is connected to the pin 8 with +5 *v*. So, the initial state with no displacement the output will be zero, and fully pressed output will be -5 *v*.

A screened cable runs through the joints of the PUMA560 for electrical connections for various applications. One connector of this 15 way cable is at the forearm and the other end is at the base of the robot.

### 6.5.4 Measurement of Compliances

There is no force sensor actually present in the robotic end-effector, and the contact is only judged by the position/rotation displacements. But names for force/torque are used to represent them. The actual force/torques can be acquired by multiplying the stiffness of the compliant device by the measured displacements.

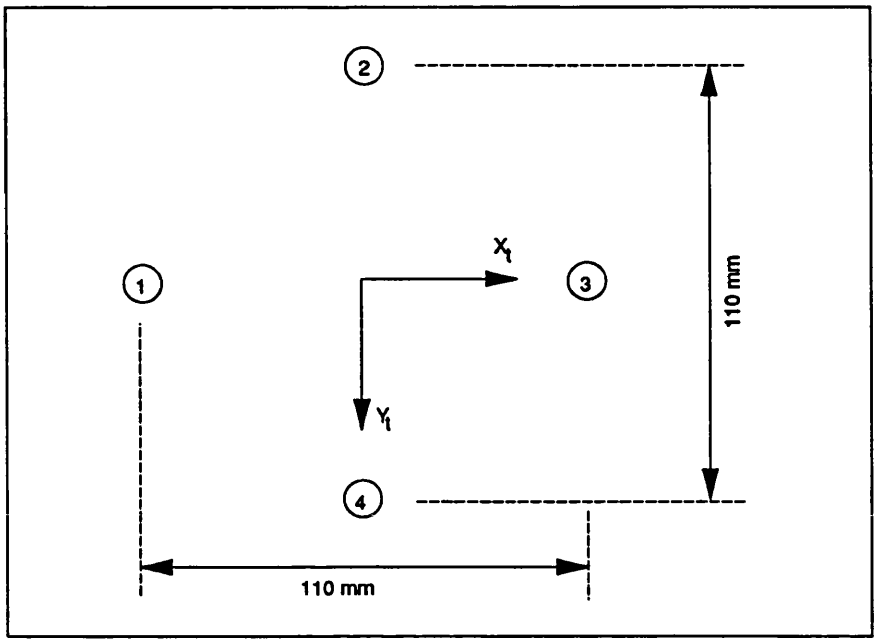



**Fig. 6.12 Position sensors arrangement**

The four position sensors have four different numbers on the ends and are arranged as shown in Fig. 6.12. The axis from sensor 1 to sensor 3 coincides with the x axis of the tool co-ordinate. And the axis from sensor 2 to sensor 4 coincides the y axis of the robot tool axis. In mounting the compliant device, attention should be paid to this arrangement to get the right measurements.

The amount of contact is judged by the average of the four position sensors' reading: $F_z = (d_1+d_2+d_3+d_4)/4$. Where $d_1$, $d_2$, $d_3$ and $d_4$ are the displacements of the four position sensors, measured by the reading of A/D converter. $F_z$ is not the actual force, but here it is stands for the contact amount. The force has to be calculated by multiplying a stiffness constant.

As can seen from Fig. 6.13 and Fig. 6.14, there are two situations which have to be distinguished. One is that full contact has been made, in which case both position sensors have been compressed. However, in the case of partial contact, only one position sensor has been compressed. These two cases have also been discussed before in Chapter 5, where they are called normal contact and oblique contact. Oblique contact, as stated before, is frequently

encountered in sub-sea inspection as the environment is unknown and also the probe has to move around the surface. Here gives the amount of analysis for those two situations.

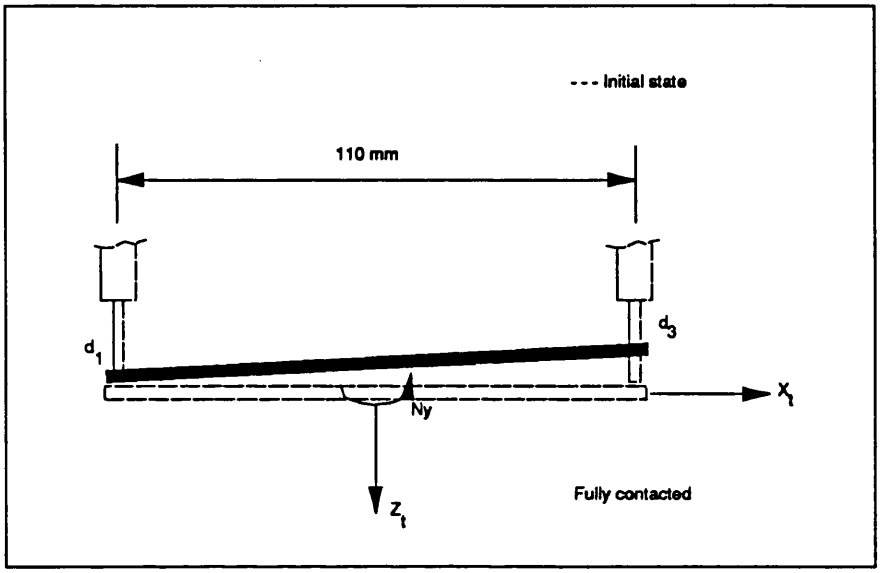



**Fig. 6.13 Measuring rotation compliances with full contact**

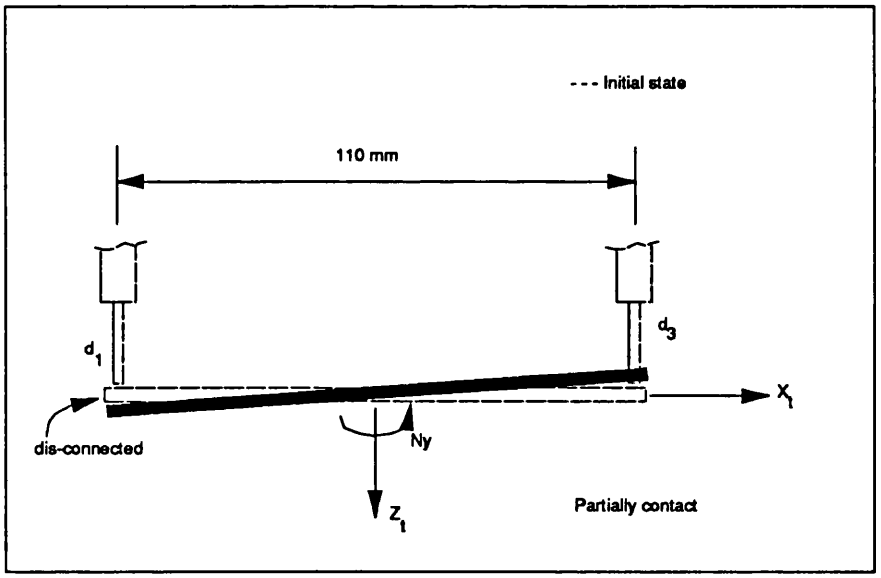



**Fig. 6.14 Measuring rotation compliances without full contact**

The rotation compliance in y axis, as shown in Fig. 6.13 and 6.14, is $N_y = (d_3-d_1)/110*(\frac{180}{\pi}) = 0.52(d_3-d_1)$ for the full contact situation. And $N_y = (d_3-d_1)/55*(\frac{180}{\pi}) = 1.04(d_3-d_1)$ for the partially contact situation. The same referring can be applied to the other direction: $N_x = (d_2-d_4)/110*(\frac{180}{\pi}) = 0.52(d_2-d_4)$ for the fully contact situation. And $N_x = (d_2-d_4)/55*(\frac{180}{\pi}) = 1.04(d_2-d_4)$ for the fully contact situation. Here, $N_x$ and $N_y$ have unit of degrees for later uses. These formulations are listed in Table 6.5 below:

**Table 6.5 Compliances of two contacts**

| | *Fully Contacted* | *Partially Contacted* |
|---|---|---|
| $N_x$ | $0.52(d_2-d_4)$ | $1.04(d_2-d_4)$ |
| $N_y$ | $0.52(d_3-d_1)$ | $1.04(d_3-d_1)$ |

## 6.6 REAL-TIME EXPERIMENTATION

The real-time experimentation was carried out using the end-effector described in the last section. This compliant device is used for robotic assembly described in this chapter and robotic inspection, described in the next chapter.

### 6.6.1 Introduction

Assembling components or parts has traditionally been a major obstacle to the automation of assembly for close-tolerance components because of the difficulties in assembling the relative positions of the parts. This problem is critical in general-purpose assembly, since most software-controlled robots are less accurate and repeatable than dedicated machines. Further conventional 'floating' mechanisms designed to absorb lateral errors between work carriers and assembly work heads are often inadequate in robotic assembly since the multi-axis robot wrists are more liable to angular errors than fixed workheads. These problems have led to extensive study of parts mating theory in recent years.

## Active Compliance

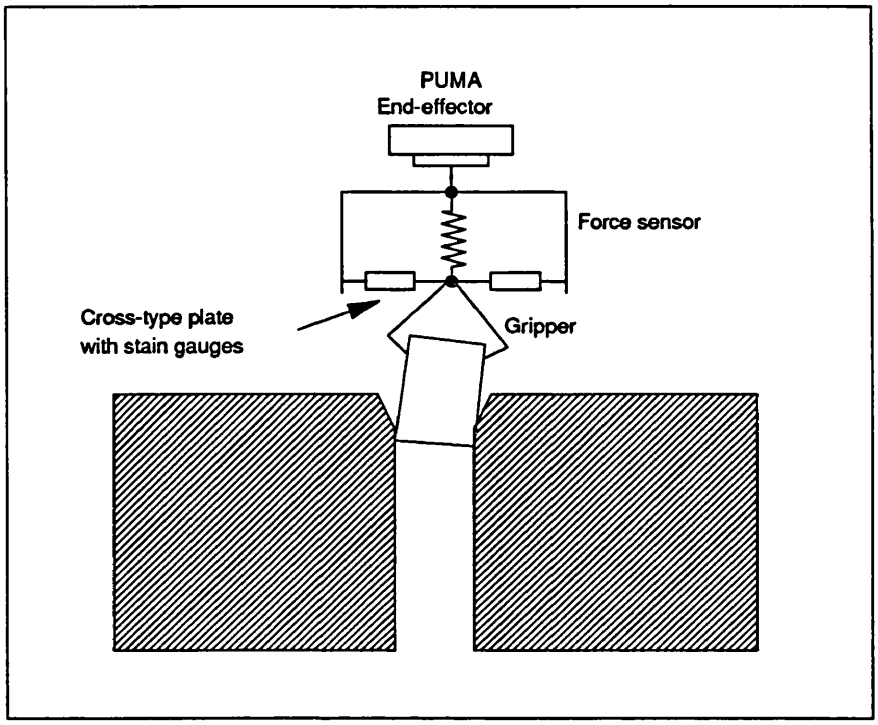



**Fig. 6.15 Force control configuration**

The need to accomplish consistent part-mating has prompted the development of active-control systems which can monitor the fine movements of the robot arm using contact-force information. Two major methods have so far been used, remote sensing of reaction forces at the manipulator joint drives (by measuring motor armature currents), and wrist/pedestal force sensing. The latter is superior in both accuracy and reliability. The Hi-T-Hand developed at Hitachi has a flexible wrist consisting of a cross-type plate spring and four tactile-sensing strain gauges (Redford and Lo [1986]).

By detecting the relative linear displacements of the mating parts, and by processing and feeding back the error magnitude to the robot through a sequence controller, the system is capable of inserting misaligned pegs into chamferless holes with clearances ranging from 7 μm to 20 μm in less than 3 seconds.

The general strategy is to make use of force feedback to generate a vector of motions at the tip of the moving part until the sensed forces become zero. One of the early demonstration showed that the system is marginally adequate for mating pieces with a clearance of 5 μm.

Although force sensors provide valuable information to control assembly operations, their role in part-mating is often limited by the system's overall speed of response and the robot's low positional resolution. To overcome this, an anxiliary servo-loop, motorized device attached to the robot to make small, high-speed, high-resolution wrist motions may be essential, forming the so called micro/macro manipulator systems.

Passive Compliance

An alternative approach to force measurement and on-line control is the use of passive compliance. This uses a mechanical device whose geometry and elasticity is designed to allow temporary structural deformations in both translation and rotation in response to force and moments. The principal design parameters are the location of the centre of compliance (i.e. the point where an applied lateral force causes only a lateral displacement and an applied torque causes only a rotation displacement), and the magnitude of the compliance (i.e. displacement per unit force/torque). One good example is the Remote Centre Compliance (RCC), which is capable of ensuring successful insertion of a bearing into a 40 mm housing with a clearance of 10 μm in 0.2 s, starting from a lateral error of 1 mm and an angular error of 1.5°.

Robot accuracy, which plays a main role in many applications, can be influenced by a number of factors, which are classified as follows:

1). Environmental, such as temperature.

2). Parametric, including kinematic parameters (such as robot link length, joint zero-reference angles etc.) and dynamic parameters (structure compliance, friction, backlash etc.).

3). Measurement, such as resolution and nonlinearity of encoders and

resolvers, which are the most common position feedback devices for robot, usually mounted directly to the motor shaft.

4). Computational, digital computer's round-off etc.

According to the factors listed above, it is possible to find methods to improve robot accuracy.

1). Calibration.

2). Kinematic parameter identification — usually called open-loop method. Dynamic parameter identification and control require computations which are still not yet economically attainable in most industrial robots.

3). Incorporate external sensors that detect end-effector positions — close-loop method. The pose of the end-effector is back fed to the robot controller on-line and an additional control loop modifies robot pose to meet requirement.

### 6.6.2 Real-time Gross Motion Control

The gross motion trajectory planning described in section 6.3 is adopted for real control of the PUMA560 from the supervisory computer. Joint angles are sent to the PUMA560 VAL-2 controller. It takes about 100 milliseconds to send one set of joint angles (six angles). As a result, PUMA560 carries move and stop movement patterns. The initial joint angles are read from VAL-2 and the vicinity of destination is specified by the user. The destination is the place the manipulator is demanded to go.

One thing which has to be taken into account is that VAL-2 uses Euler angles to express orientation of end-effector. As described in Chapter 2, in this project, manipulator orientation is specified by the direction of the axes of the tool co-ordinates in the world co-ordinates. Usually any two of the axes of the tool co-ordinate will be enough to specify the whole tool co-ordinate. $\mathbf{z}_t$ and $\mathbf{y}_t$ of the tool co-ordinate are chosen to stands for the tool co-ordinate, and $\mathbf{x}_t$ can be calculated by the cross product of $\mathbf{z}_t$ and $\mathbf{y}_t$.

$z_t$ always coincides with link 6 of manipulator and $y_t$ originally coincides with axis of joint 5. In many applications, $y_t$ can be the same as joint 5 in the whole manipulation. $z_t$ is specified by its three projections to the **x**, **y** and **z** axis in the world co-ordinate. For example, $z_t$ equals (1, 0, 0) if it coincides with **x** axis of the world co-ordinate. For more detail, please refer to Greig [1992].

If the position/orientation has been thus specified, the final set of joint angles for the destination can then be calculated using inverse kinematics given in Chapter 2. The supervisory computer is then able to move the PUMA manipulator in joint space along the joint trajectory as stated in section 6.3.

### 6.6.3 Design of a Peg and Hole

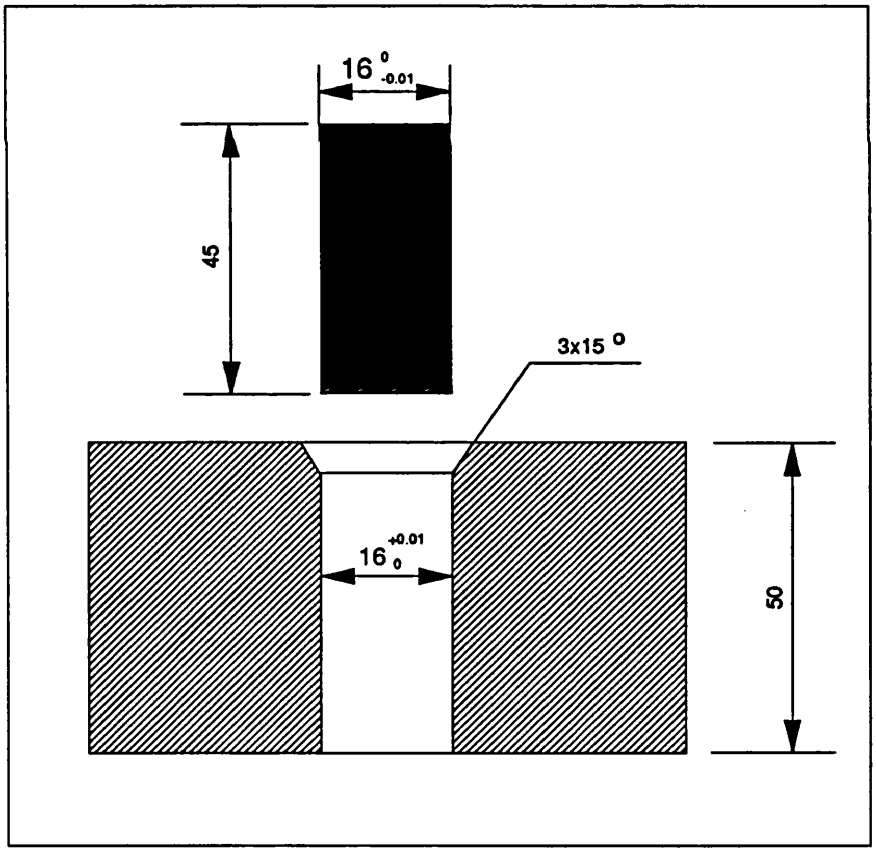



**Fig. 6.16 Drawing of the peg and hole**

The minimum clearances a robotic manipulator can do the assembly of putting a peg into hole is decided by the robot's overall resolution. The minimum division of each joint encoder contributes to the system over-all resolution. PUMA560 has a resolution of 0.01 mm (Greig [1992]), which means that motion demands less than 0.01 mm will be ignored by the PUMA controller.

As can see from Fig. 6.16, the maximum clearance between the peg and the hole is 0.02 mm. The process in putting peg into hole is going to be described in the next sub-sections.

### 6.6.4 VAL-2 Program for Peg-into-Hole Problem

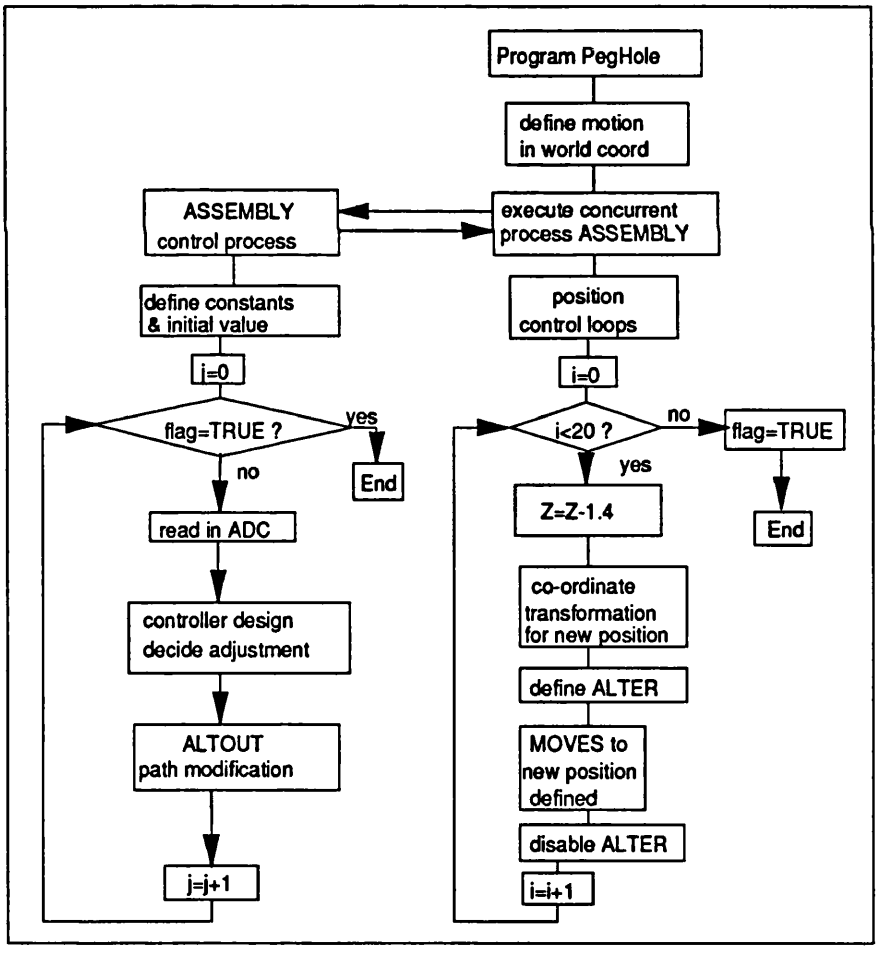



Fig. 6.17 Peg-into-hole VAL-2 program flow chart

Fig. 6.17 shows the VAL-2 program structure for peg-into-hole problem. And the actual program is listed in Appendix 6.1. As can see, there are two processes, one being the position control loop — the right one — using VAL-2 command MOVES (total 25 lines). The other is for compliant motion control using ALTOUT command (total 46 lines). The VAL-2 program for reading four position sensors, described before, are listed in Appendix 6.2 (total 21 lines).

### 6.6.5 Control Strategy for Assembly Process

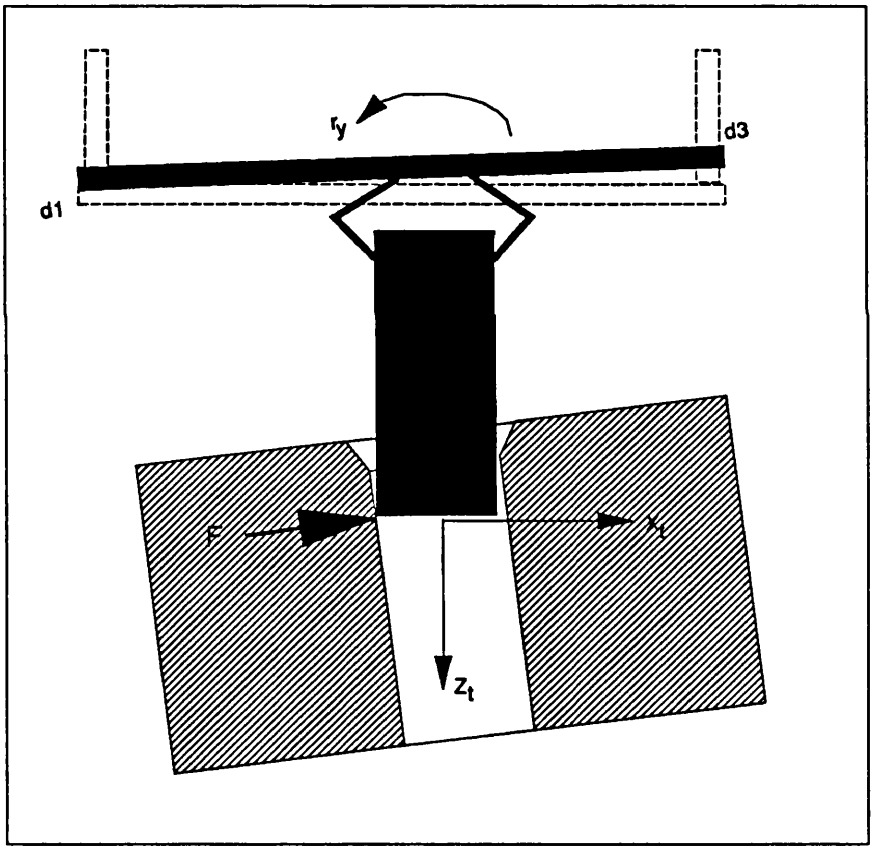



**Fig. 6.18 Control strategy for assembly**

In Fig. 6.18, the hole is vertically fixed to the world co-ordinate. To show the control strategy, the figure is drawn exaggerated. The idea is that when the peg hits the wall as shown in Fig. 6.18, the contact force F will compress the

right position sensor and stretch the left sensor. To make the peg align with the hole, the rotation adjustment has to be in the same direction as $\mathbf{r}_y$ shown in Fig. 6.18. So the adjusted rotation displacement is:

$$r_y = 0.52(d_3 - d_1)$$
$$r_x = 0.52(d_2 - d_4) \qquad (6\text{-}4)$$

This control strategy has the same form as the measuring of pseud twist torques described in section 6.4.

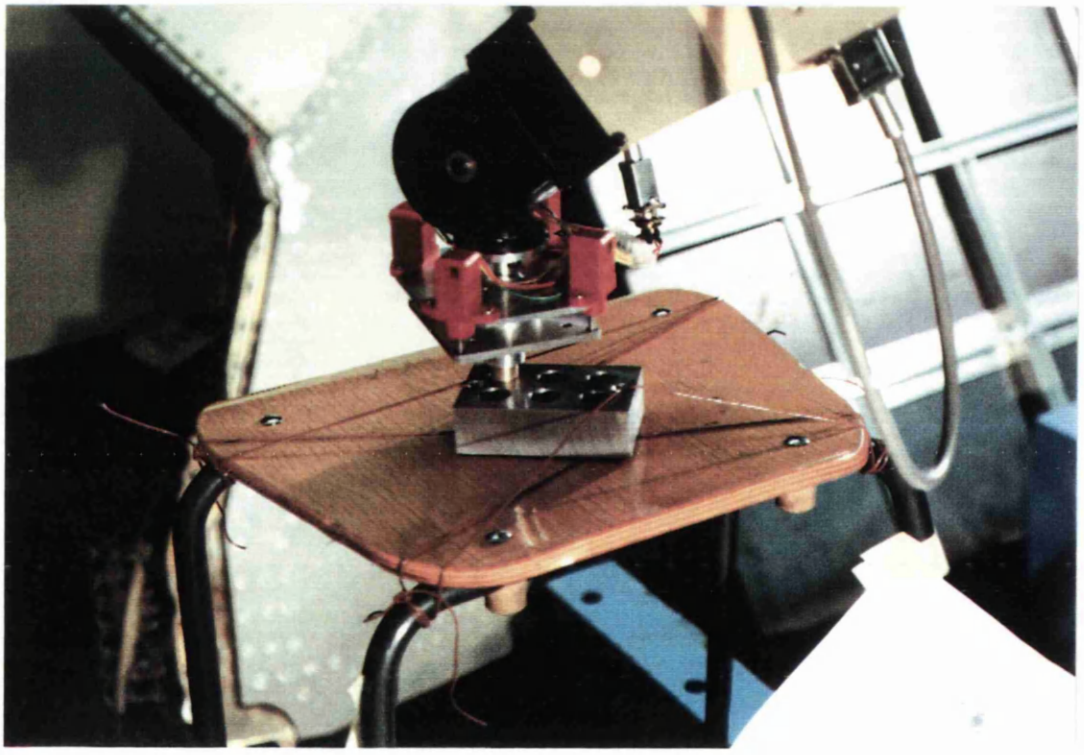


**Fig. 6.19 Picture to show successful peg into hole assembly**

Fig. 6.19 shows successful experimentation. The PUMA560 robotic manipulator has put a peg into a hole with maximum clearance 0.02 mm.

## 6.7 SUMMARY

In this chapter, a few less relevant subjects are described: 1). Two kinds of trajectory are presented; 2). A simulation program package on a VAX work station is detailed, emphasis being placed on the simulation of the assembly process; 3). Design of a compliant device is detailed, which is used for real-time experiment described in this chapter and will be in the next chapter; 4). Real-time experiment of peg-into-hole of 0.02 mm maximum clearances was carried out successfully.

The compliant end-effector can be thought as a passive device. The control strategy is to actively control this passive compliance to solve practical problems. A less stiff compliant device is necessary for stability. However, in the assembly operation, less stiff compliant device will reduce the manipulator overall stiffness resulting in a more sluggish response. As a matter of fact, in the process of putting peg into hole the manipulation speed has been reduce to avoid jamming.

To increase the operation speed of the manipulator in the assembly process, more stiff compliant device is recommended on the condition that this will not affect the arm dynamics. Obviously further research is needed in this area.

Joint-interpolated trajectory has the disadvantage of being unable to control the overall Cartesian trajectory. However, using Euler angles to specify end-effector orientation presents some problem as an invalid solution can be easily found. This restricts its wider use. The existing robotic manipulator uses the TEACH pendant to teach the manipulator's motion. A set of intermediate positions are recorded for use afterwards. In many situations teaching is rather difficult and also teaching is itself a time consuming and skillful task for the operator.

A method combining both joint space and Cartesian space trajectory planning is worth persuing. The idea is that the position of the manipulator which is

decided by the first three joints (1, 2 and 3) is planned by the Cartesian space trajectory planning whereas the orientation can always be planned using the joint space trajectory planning as the orientation will rarely meet any constraints.

## Appendix 6.1 VAL-2 Program for Assembly

```
. PROGRAM PegHole
1           mode=19
2           PCEXECUTE Assembly, -1, 0
3           HERE wh
4           DECOMPOSE wh[]=wh
5           k=0
6           SPEED 30 MMPS ALWAYS
7           DO
8                   FOR j=0 TO 20 STEP 1
9                       wh[2]=wh[2]-1.4
10                      SET loc=... ...
13                      NOALTER
14                  END
15                  ... ...
22                  k=k+1
23          UNTIL k==3
24          flag=TRUE
25          PCEND
. END

. PROGRAM Assembly
1           flag=FALSE
2           dx=0
3           dy=0
4           dz=0
5           rx=0
6           ry=0
7           rz=0
8           except=0
9           DO
10                  CALL wadac
11                  ... ...
35                  rx=0.052*(d2-d4)
36                  ry=0.052*(d3-d1)
37                  ALTOUT except, {0},{0},{0},{rx*TOANG},{ry*TOANG},{0}
38                  ... ...
```

```
45          UNTIL flag==TRUE
46          RETURN
. END
```

## Appendix 6.2 VAL-2 Program for Calculating Displacements

```
. PROGRAM wadac
1       v1=ADC(1)/204.7
2       v2=ADC(2)/204.7
3       v3=ADC(3)/204.7
4       v3=ADC(4)/204.7
5       d1 = -1.9575*v1 + 0.0008
6       d2 = -1.9508*v2 + 0.3197
7       d3 = -1.9825*v3 + 0.1603
8       d4 = -1.9867*v4 + 0.0189
9       IF d1>7.5 THEN
10        HALT
11      END
12      IF d2>7.5 THEN
13        HALT
14      END
15      IF d3>7.5 THEN
16        HALT
17      END
18      IF d4>7.5 THEN
19        HALT
20      END
21      RETURN
. END
```

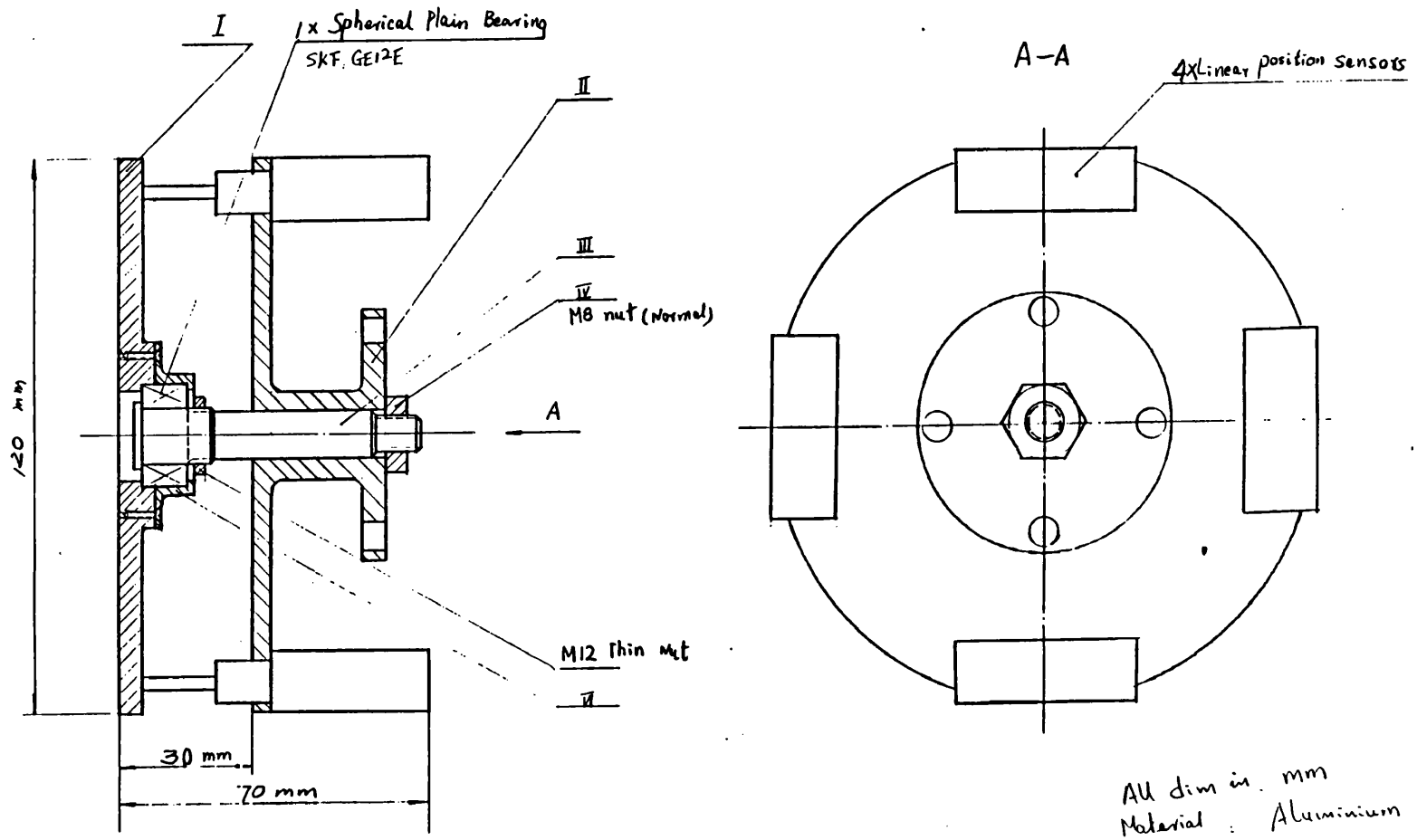
## Appendix 6.3 Description of a MODULA-2 Program Package

The program package XINGYUAN written in MODULA-2 is simply described in this appendix. There are 11 sub-packages which contain procedures. Their functions are listed below:

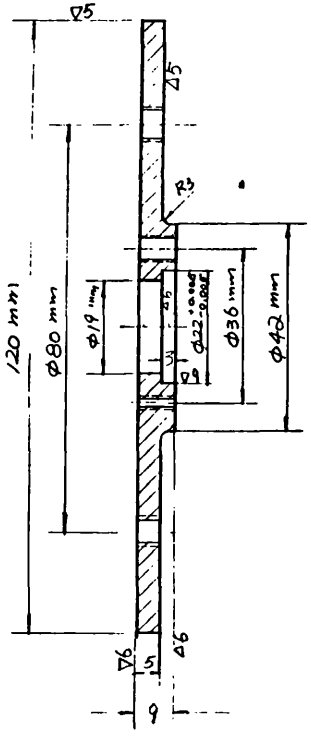
| *Modules* | *Descriptions* |
|---|---|
| XINGYUAN | Main module driving the window menu system to carry out man-machine dialogue. This main module includes two second level modules **Simulati** and **RealCntr** for simulation and real-time supervisory control as following described. |

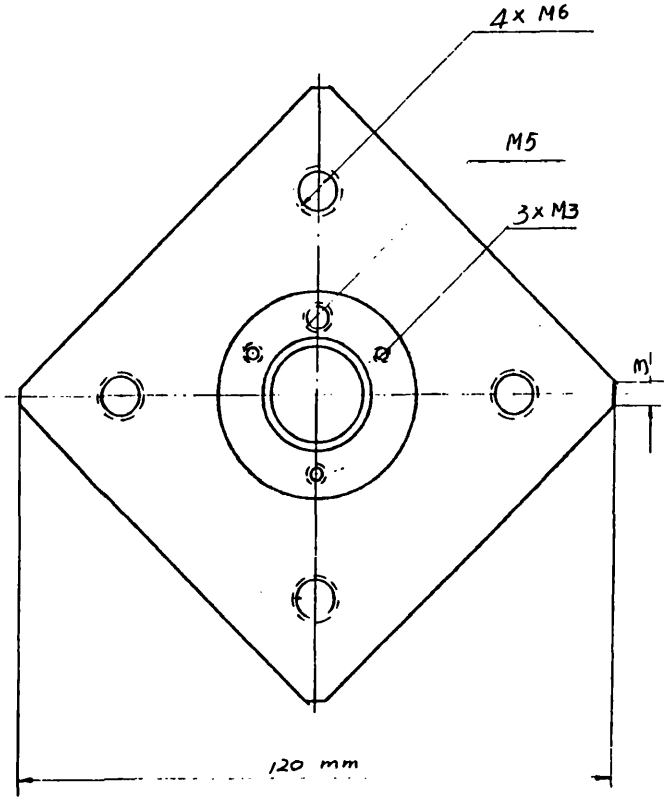
| | |
|---|---|
| **Simulati** | Module to do simulation of PUMA560 gross motion. The dynamic simulation with graphic displaying can be included in this part of the program. |
| **RealCntr** | Module to do PUMA560 gross motion control from the IBM supervisory computer. |
| **MotionCt** | Trajectory planning based on the discussion of section 6.3. Both simulation and real-time motion control need this module to carry out motion control. |
| **PumaDraw** | Module for graphics on IBM PC to draw a skeleton manipulator and its working environment. |
| **PumaKine** | Module for PUMA560 kinematics analysis. This module has been largely changed from the original VAX PASCAL code mainly because which has different co-ordinate system from the PUMA560 co-ordinate system. And also the definition of orientations is different from VAL-2's Euler angles. |
| **ForceDis** | Module to display graphic force/torque information. Graphically displaying these information in windows to make it straight forward for operator to supervise and interesting. |
| **WdHandle** | Module to define the window system. The whole computer screen is divided into five small windows, each of which stands for a different process. |
| **VecLab** | Module for mathematics. MODULA-2 programs for vector and matric manipulations etc. |
| **PumaITF** | Module for PUMA560 interfacing and will be detailed in Chapter 7. |
| **DDCMP** | DDCMP communication protocol in MODULA-2 and will be detailed in Chapter 7. |

## Appendix 6.4 Engineering Drawings of the Compliant Device



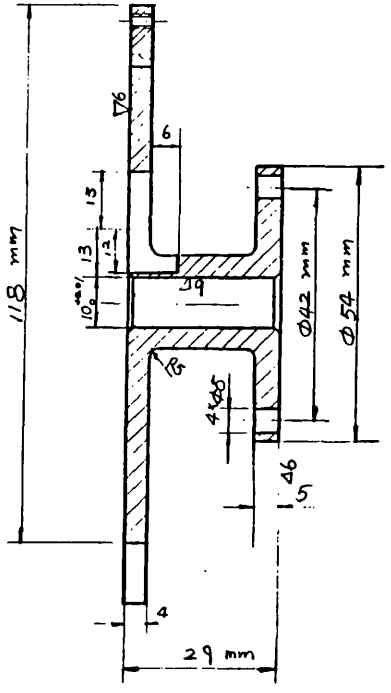
I

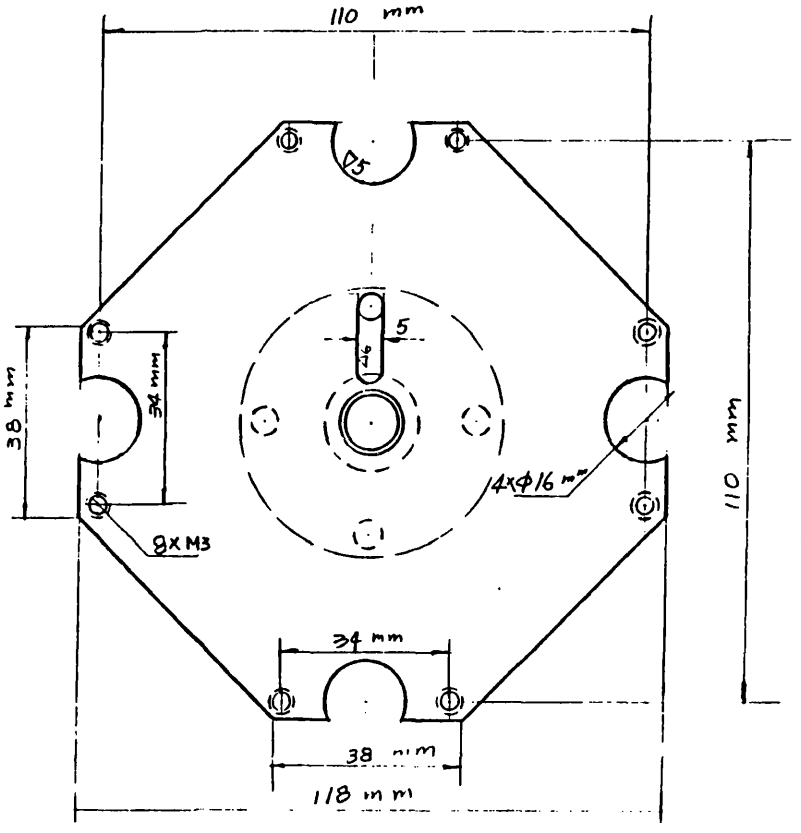



Designer : Q WANG
Data : 24/1/90
Supervisor:

II

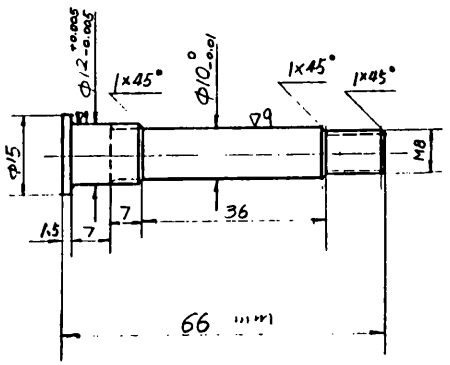



ALL DIM : mm
Material: Aluminium
Designer: Q WANG
Data : 24/1/90
Supervisor:

III

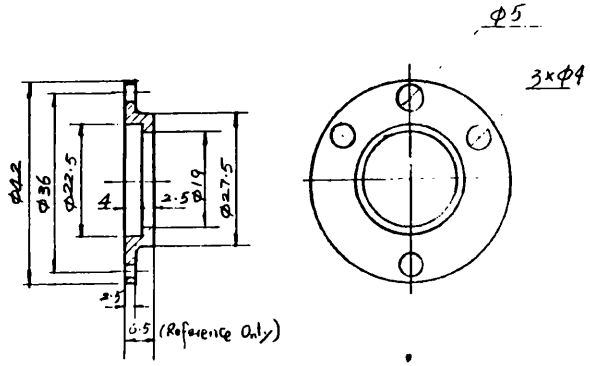
Remains ▽6




DIM: mm
Material: Brass
Designer: Q WANG
Data: 24/1/90
Supervisor:

V

all: ▽5




DIM. mm
Material: Aluminium
Designer: Q WANG
Data: 24/1/90
Supervisor:

# CHAPTER 7
# USE OF FORCE CONTROL FOR ROBOTIC INSPECTION

## 7.1 INTRODUCTION

During inspection work the probe is required to move around the test surface steadily and smoothly. It is highly desirable if a constant contact can be maintained to the surface to make as little noise as possible. This is equally true in deburring operations, otherwise the finished surface will not be accurate. In Chapter 5, the methods for robotic compliant motion control were discussed in detail, and a new approach — self-tuning adaptive control of robotic contact, was also applied for the first time. Adaptive compliant motion control is of special importance when the environment and end-effector are both not well known. It is particularly suitable for the case of sub-sea inspection where the environment is not well known, the structure is complex and the robot arm has nonlinear dynamic behaviour.

It is usual practice for divers to make oil rig inspection but this is both dangerous and, at greater depths where saturation diving is required, very expensive. The Automatic Control Group, University College London, has being involved for the last decade, in automating sub-sea inspection using robotic manipulators. Various sensor based control systems for the robotic manipulators have been developed, which include:

1). Vision processing to form a basic knowledge of the operation environment, and to provide trajectory information for force or tactile control.
2). Robotic tactile (touch) sensing to identify location and geometrical shape of objects.
3). Real-time compliant motion control to follow complex geometrical shapes and identification of stiffness of objects.
4). Various devices for crake detections.

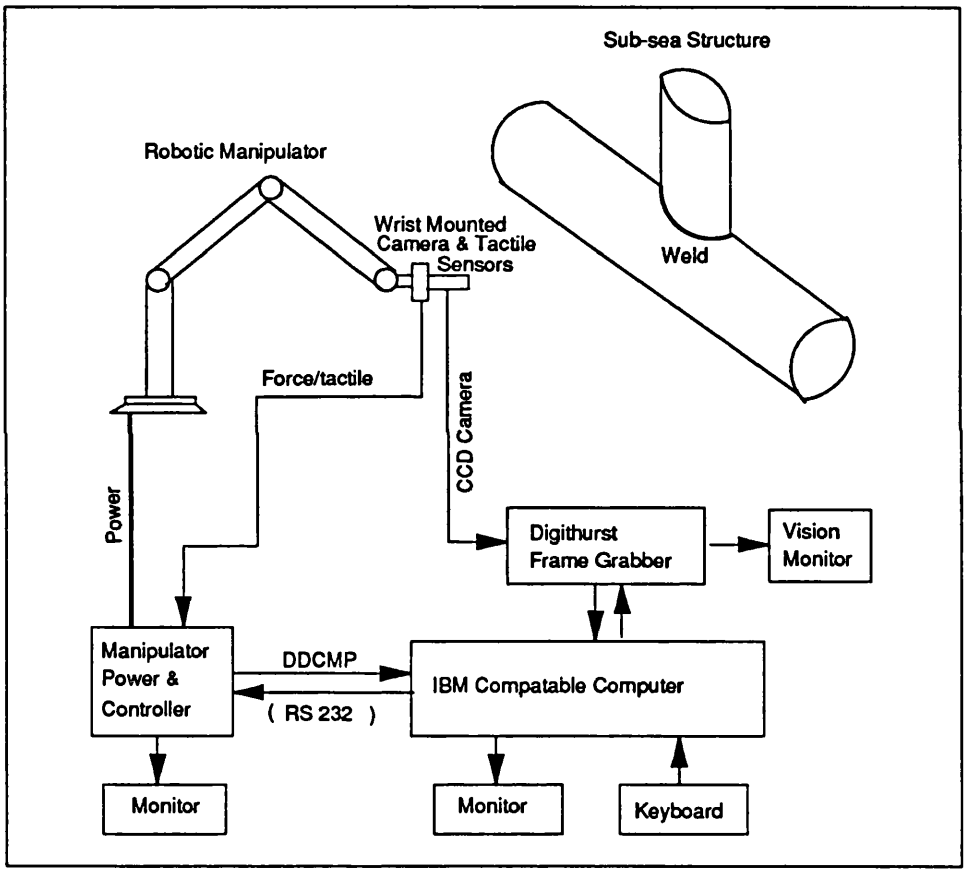



**Fig. 7.1 Hardware arrangement of sub-sea inspection systems**

Welded intersections of sub-sea structural members require regular inspection of fatigue cracks formed by wave loading. As off-shore platforms grow in age and number, divers risks and costs increase and the search for mineral reserves moves into ever deeper waters, the ability to undertake such tasks remotely becomes not only desirable, but essential. In the work described here, it is

supposed that the modelling of the sub-sea environment has been completed. The trajectory for robotic manipulator end-effector to perform inspection tasks has been defined by computing the intersections of the structural members. The subsequent work is to move the probe along the welds to carry out the inspection operation.

Fig. 7.1 shows a sub-sea inspection system. Various sensors are used to identify the sub-sea environment. Usually, there are four stages in the sub-sea inspection (Greig [1991]):

- First pass weld location: Identify the area which requires to be cleared of marine growth to reveal the weld itself.
- Weld area clearing: Using the estimated weld from the first pass weld location the weld area is grit blasted.
- Second pass weld location: Clearly identifiable weld trajectory can be accurately mapped and the weld roots identified.
- Deploying inspection probe on to the weld roots.

The experimental work described here contributes to the last of the above list. After the trajectory of the weld has been found out using various sensors, a robotic manipulator with deployed probe is to be moved around the weld, which can be approximated by a cubic curve, to detect if there are cracks in the weld.

This chapter is going to discuss robotic inspection using force feedback control. The following section will have a discussion of the task formalism for using force control for sub-sea inspections. Section 7.3 describes a MODULA-2 program package for PUMA560 supervisory control with a windows based menu system and multi-process support, capable of incorporating real-time compliant motion control. The correspond VAL-2 system is described in section 7.4, and the communication between MODULA-2 and VAL-2 is

established using DDCMP communication protocol as used in DECnet. Section 7.5 discusses the real-time experimental implementation. The last section is for discussion and conclusion.

## 7.2 TASK FORMALISM FOR ROBOTIC INSPECTION

The functional specification of a compliant motion task plays a role similar to the trajectory planning module in a pure position-control situation as already discussed in Chapter 6. For a given task, it produces desired end-effector position trajectories and desired contact-force trajectories. These trajectories are expressed in a suitable reference frame (task frame) and are passed on to the compliant motion controller.

The term "compliant motion" refers to manipulation tasks which involve continuous contact between manipulator and its environment, and during the execution of which the end-effector trajectory is modified by the occurring contact forces. Examples are peg-into-hole assembly, following a contour or a surface as in sub-sea inspection, etc. In order for robots to improve their adaptability, it is necessary for the underlying control strategies to become more sophisticated. This is especially true in contact tasks such as grinding or deburring and inspection where it is desired not only for robots to apply a constant force, but also to reject undesirable high-frequency disturbances.

Chapter 6 had a discussion about control formalism of the peg-into-hole problem. In this chapter, inspection task formalism is to be addressed. Schutter and Koivo [1988] gives a detailed description about force control formalism, which further extends Mason's theory (Mason [1981]). According to him, every compliant task configuration can be divided into position- (or velocity-) controlled directions and force-controlled directions. Both sets of directions are perpendicular and complementary, configuring the so-called task frame

(or compliance frame). The position- and force-controlled directions may vary with respect to the world reference frame during the execution of the motion; ie., the task frame is a dynamic frame with its own trajectory.

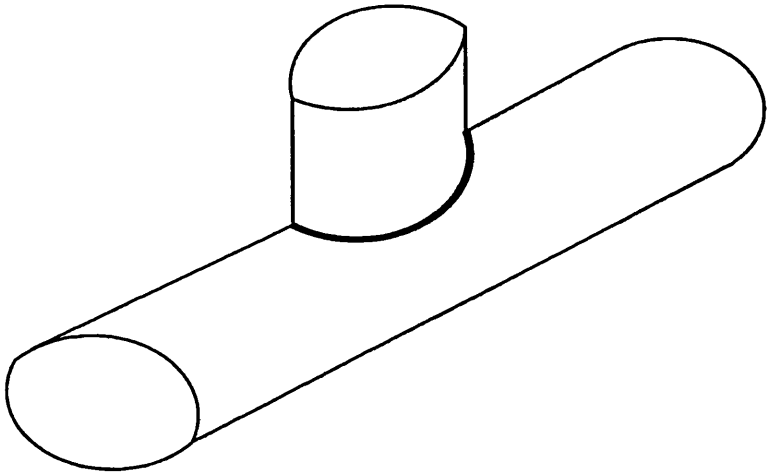


**Fig. 7.2 Typical sub-sea structure — Y-joint**

Fig. 7.2 shows a typical sub-sea structure — a Y-joint. The task presented here is assigned as sub-sea inspection after the joint node of an off-shore structure has been located using laser or tactile sensing etc., which has been carried out at UCL for the last decade (Broome [1984], Broome [1987], Savut [1989], Greig [1990] and Broome [1988]). A topside supervisory control computer will drive the robotic manipulator to the vicinity of the joint node (welds). The inspection work would be carried out afterwards by the robot itself. This kind of tasks can be modelled as Fig. 7.3.

The task can well be described as 'track an unknown and arbitrary (but connected) two-dimensional contour in the xy plane while moving at constant tangential speed and while applying a constant normal force'. Fig. 7.3 shows three relevant frames: 1). The global reference frame, which is the world co-ordinate system (usually is defined as PUMA BASE co-ordinates); 2). The task frame, in which the task is expressed in terms of position and force directions; 3). The robot end-effector frame (PUMA TOOL co-ordinates). There is one more relative frame, which is fixed to the environment, object

frame (not shown in Fig. 7.3). In this example it seems obvious to define task frame fixed with respect to the object to be manipulated (object frame). However, as there is friction between contact surfaces and compliance in the contact point, task frame will not be the same as the object frame.

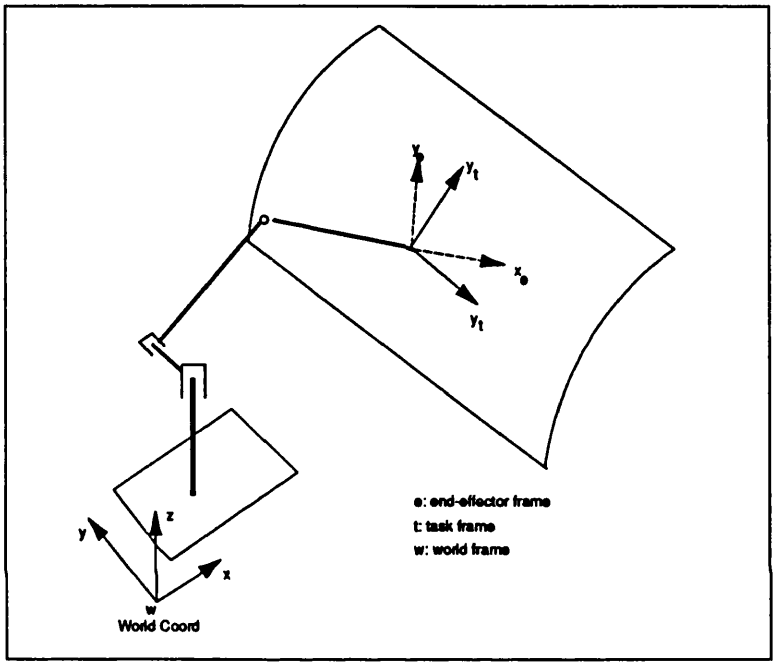



**Fig. 7.3 Inspection task formalism**

From the manipulation aspect of the PUMA560, it is convenient to set task frame fixed to the robotic end-effector. However, from the point of view of task operation, it is highly desirable to specify the task frame associated with the object to be manipulated. This problem is overcome by adjusting the robot end-effector to fit the environment continually. That is to make **z** direction of robotic end-effector frame (TOOL co-ordinates) continually coincide with **z** direction of the task frame. In PUMA560 case, task frame is continually specified in TOOL co-ordinates. The task frame is continually updated using the HERE command in VAL-2. This makes sure that TOOL co-ordinates of the robotic manipulator always coincide with the task frame, which will subsequently make the contact force easier to maintain.

The task frame is defined fixed relative to the end-effector, with the origin being at the centre of contact, with task-frame directions:

x: velocity $v_x$ mm/s
y: velocity $v_y$ mm/s
z: force $F_d$ (N)
$r_x$: torque 0 Nm
$r_y$: torque 0 Nm
$r_z$: angular velocity 0 rad/s

From this specification, there are three force-controlled directions and three position-controlled directions.

## 7.3 MODULA-2 CONTROL PROGRAMS

It was decided to use MODULA-2 for the supervisory control software. MODULA-2 provides many good features, which other compilers do not have. With MODULA-2 windows, menu-drive software is developed for easy man-machine interface and better display facilities. With multi-processes support, a single processor can do many tasks at the same time. This concurrence is especially important in the control of a robotic manipulator. A simple example can explain this: in the sub-sea operation, the supervisory computer is usually needed to process multi-sensor data, e.g. vision, force sensor, sonar and tactile etc. At the same time the supervisory computer is needed to provide information of how the robotic manipulator reacts to the changing environment. These tasks are simultaneous.

In this section, a MODULA-2 supervisory programme, capable of incorporating real-time compliant motion control, is described in the next few sub-sections. Program structures and layouts have been drawn, and examples are given in the appendixes.

### 7.3.1 Supervisory Control System Structure

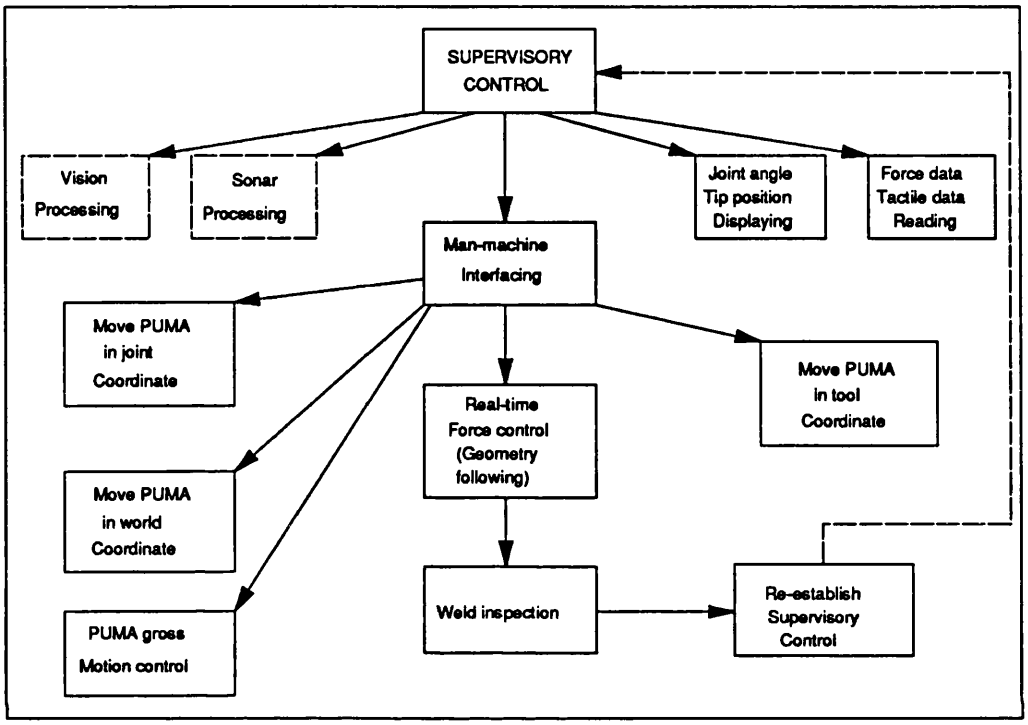



**Fig. 7.4 Structure of the supervisory control programs**

Fig. 7.4 shows the MODULA-2 supervisory control program structures. There are five processes running concurrently in the IBM computer. Two levels are distinguishing: planning and reaction. The dialogue process is reaction, which ensure that the PUMA motion is under human control in deciding which mode of reaction to be taken. The planning level incorporates all the sensor data processing and decisions to be made on where to move the PUMA robot arm in what trajectory.

Fig. 7.5 shows the structure layouts of the

supervisory control, where the bottom is the RS module, which is provided by the MODULA-2 compiler to drive the RS-232 serial port (COM1). The module DDCMP is written according to the format of the DDCMP protocol for sending and receiving message from the serial port buffer. DDCMP module only uses RS module. The PUMA560 interfacing module, PumaITF, is then written based on DDCMP module, according to different tasks. The last one is the XingYuan module for man-machine dialogue, the top layout.

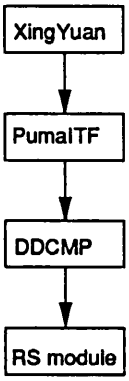



**Fig. 7.5 Supervisory control program layout**

In the man-machine interfacing process (Fig. 7.4), five different modes of PUMA control are specified, which are driving PUMA in joint, world and tool co-ordinates, PUMA arm gross motion control and real-time compliant motion control. The last two will be discussed in detail in a later section. The first three simulate PUMA teach pendant. When a suitable plan of how to move the arm has been worked out using vision, sonar and other sensors, various PUMA arm control modes can be invoked to drive the PUMA arm to its destination. Then real-time compliant motion control can be carried out to do the inspection task by deploying an end-effector mounted probe over the welds. After the supervisory control has been re-established, new planning would continue (Wang [1991b]).

### 7.3.2 Window Menu System

The whole computer screen is divided into several sub-windows, and each presents a different process. As shown in the Fig. 7.6, there is a menu window at the top of the screen, which functions as a dialogue window to make man-machine dialogue. There is a main window, which displays on-line the angles of the six joints and tip-position -orientation to see if they are within

the suitable range for an on-line human supervising. One small window on the upper right displays the relevant message, as an assistant menu window, for better man-machine dialogue. One lower right small window displays force (or tactile, sonar etc) information to see if contact or other interactions have occurred. There is an one line hint window at the bottom of the screen, which is used to give some hints of how to key in commands.

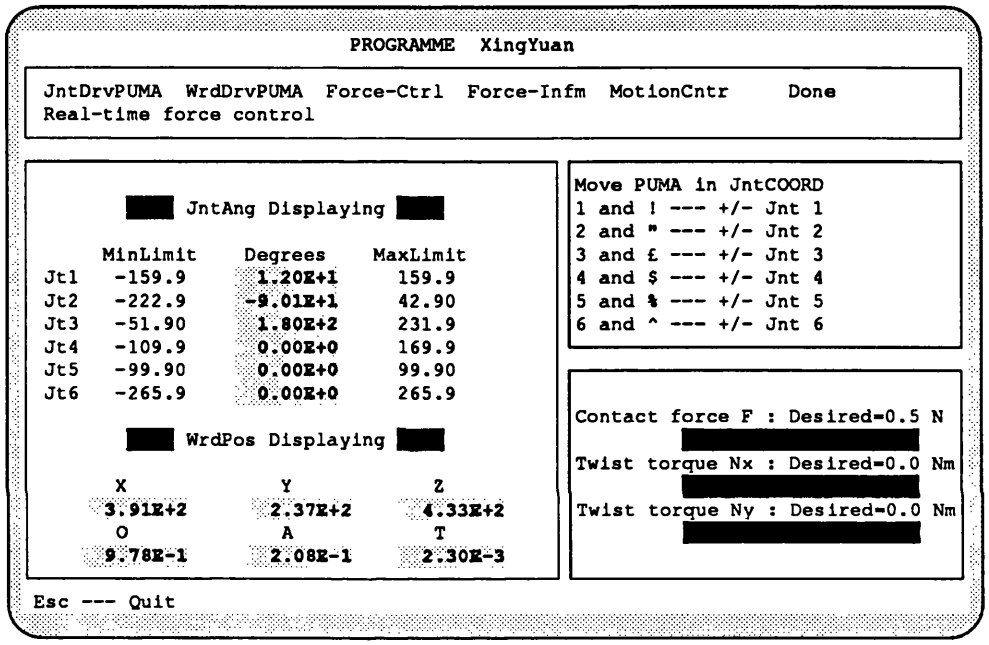



**Fig. 7.6 Man-machine interface**

By moving the cursor keys, request commands can be chosen and there is an explanation line in the menu window to explain the meaning of each command.

It is usually the case that in sub-sea operation, the manipulator has to try many directions to make contact with the structure. So it is important that the human operator or the computer knows whether contact has been made with the environment. The force display screen should be able to display this situation. Therefore, it is highly desirable that the force display process runs all the time without stopping. At the same time the human operator must be able to supervise the manipulator's motion.

Tests have shown that the operator will give wrong motion commands when the processes do not run concurrently. The force information displayed on the screen is out of date if there is any interval between operator's supervisory inputs. The operator can easily make mistakes if force and other information is not continuously displayed. With concurrence, all the sensors' information will be displayed without stop on the screen for the human operator to make right decision. The next sub-section will discussion this subject more.

### 7.3.3 Multi-process Support

One of the most exciting and difficult features of MODULA-2 is its support of co-routines and concurrent processes. A language rarely defines these concepts because, traditionally, they have been left to the operating system or transputer etc. Wirth (Wirth [1988]) recognized the need to standardize a concurrent programming interface so that the interface could remain fixed and stable in a variety of environments. A process is a task that you can think of as a program or a procedure. If two processes are executing simultaneously on the same computer, they are called concurrent processes. Most computers have only one CPU, thus only quasi-concurrency is possible because the computer is simply switching between processes rapidly, which gives the appearance of concurrent execution. Some computers actually have two or more CPUs, which allow true concurrent execution of processes.

By simple definition, co-routines are separate processes that are part of the same program. The primary difference between a procedure and a co-routine is in the way the control is transferred. When one procedure calls another, the called procedure always executes from the top, as shown in the upper part of Fig. 7.7 However, when one co-routine transfers control to another, execution is assumed at the last point of execution before the previous transfer of control, which can be anywhere in the co-routine, as shown in the lower part of Fig. 7.7

When one co-routine gives control over to another, the first co-routine is suspended, and no further execution occurs until the second co-routine returns control. In MODULA-2, you can let the schedule control what co-routine is actually executing at any given moment, or you can explicitly transfer control to various co-routines by using program control. Co-routines are particularly useful in the construction of operating systems and other programs in which processes must appear to operate concurrently.

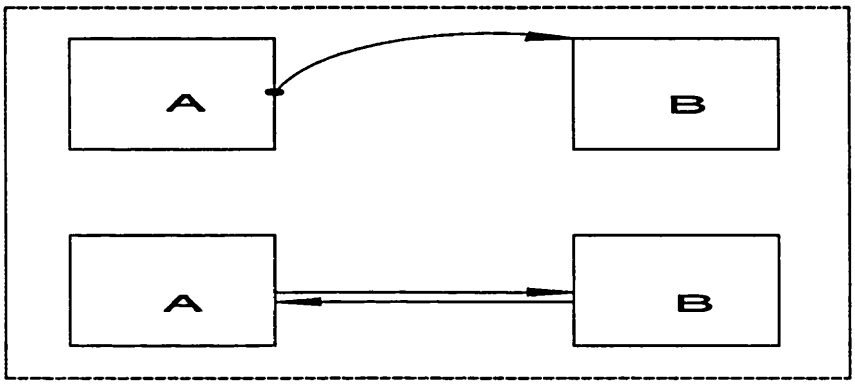



**Fig. 7.7 Control transfer between procedures**

It is highly desirable if a single processor can do many processes simultaneously. Concurrence is especially important in remote operation like sub-sea inspection where data from many sensors are needed to be processed and reaction of the arm is to be specified.

In many cases, some of the processes, like vision processing, force (tactile) and sonar data, have to be kept running for the reason given at the last sub-section. Other processes, like response of PUMA arm, should be able to be under control of human operator. The display of joint angles, tip-positions and force sensor readings, should be kept running. However, the operator should be able to, at the same time, make dialogue with the computer and even stop the processes running if anything goes wrong. The computer can be stopped by re-booting it. But in many situations, re-booting the machine is not the right choice as communication between the manipulator controller and the supervisory computer has to be re-established.

### 7.3.4 DDCMP Communication Protocol and Program

DDCMP stands for Digital Data Communication Message Protocol. It is DEC company protocol used for computer net work. UNIMATION also use DEC's protocol to enable the PUMA arm and its VAL-2 controller to communicate with a supervisory computer to establish supervisory control. Supervisory control is very important in remote operation situation where data are to be collected and processed from many sensors. A supervisory computer can be engaged in most of these tasks.

VAL-2 uses DDCMP as its second bottom layer communication protocol to send and receive messages. This is a rigourous, byte-count oriented protocol which automatically detects transmission errors and re-transmits messages if necessary. It can be used on synchrous or asynchrous, half or full-duplex, serial or parallel and point-to-point or multi-point systems. The message format contains two parts: a header including control information and data string.

A MODULA-2 program was written to substitute for the original ASSEMBLY code written for PASCAL support (Savut [1988]). The compiler provides a implementation module called *rs.MOD*, to drive the RS-232 and compatible serial ports, which is also written in MODULA-2 instead of ASSEMBLER. The DDCMP protocol was written based on this bottom layer. The whole program is compact, concise and easy to understand as there is no ASSEMBLER involved.

In Fig. 7.8, the RS module provides the essential bottom level of BYTE sending and receiving. And every time when a BYTE has been sent or received, the Cyclic Redundancy Code (CRC) has to be calculated to check information error. The purposes of the top level of the DDCMP are sending message, sending acknowledgement and receiving message from the serial port RS-232.

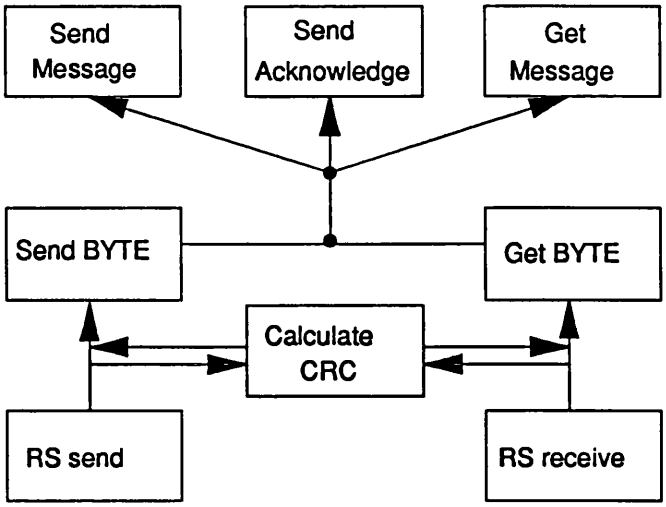



**Fig. 7.8 DDCMP program frame**

DDCMP is a reliable communication protocol particularly suitable for long distance communication between computers. It's disadvantage is also its over (sometimes) reliability resulting in a slow communication speed. It takes about 100 milliseconds to send/receive a set of six-joint motion commands. This makes real-time control from the supervisory computer very difficult. Another physical line has to be established to make real-time path modification using an external computer. Fortunately real-time path modification from the external computer is not crucial, as it can be incorporated into the supervisory control as will be described in the next sub-section.

### 7.3.5 Incorporating Real-time Compliant Motion Control

This sub-section describes how the supervisory control program incorporates the new PUMA560 real-time path modification (geometry following), and how the external computer to get large deal of force/torque information from the PUMA560 terminal. PUMA interface programs are based on the DDCMP lower level communication programs after basic communication has been established. Driving PUMA in joint, world and tool co-ordinates is not described here in detail.

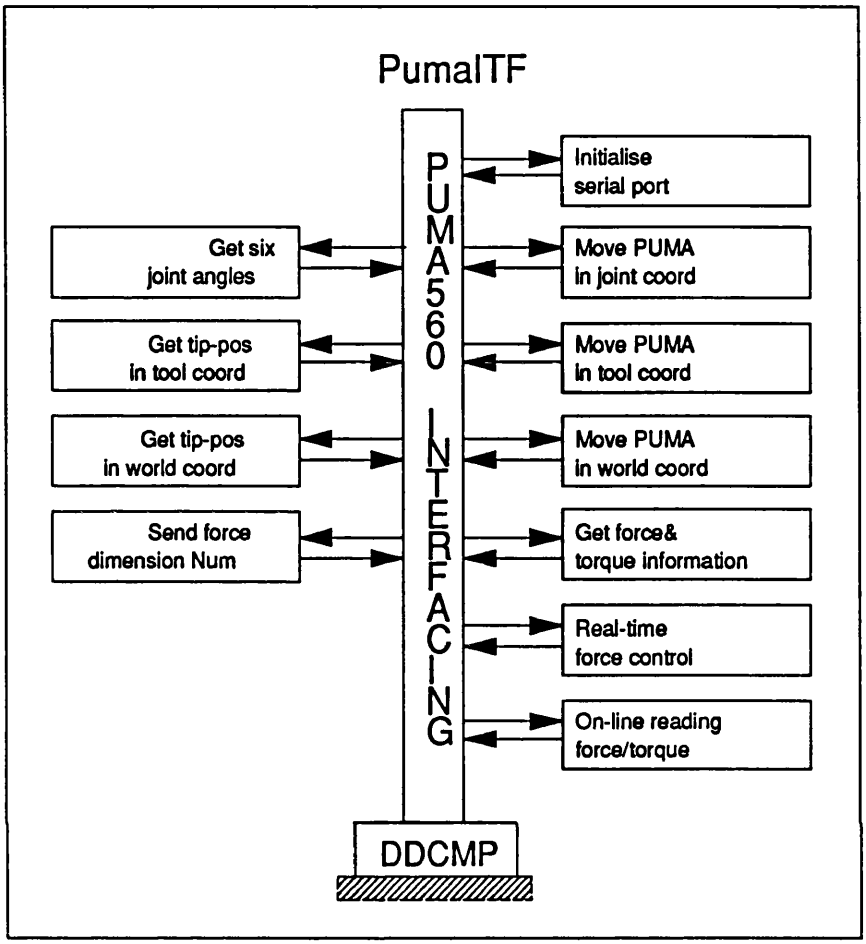



**Fig. 7.9 PumaITF structure frame**

The program actually making dialogue with PUMA560 computer is called *PumaITF*, which means PUMA interfacing. The whole program frame is shown in Fig. 7.9. Before invoking any of the *PumaITF* routines, initialising the serial port has to be done. So this routine is placed at the top. The other ten interfacing modes are simply introduced below. There are three co-ordinate systems, i.e. JOINT, TOOL and WORLD. The data is interpreted in JOINT mode if joint co-ordinate is chosen; similarly for TOOL and WORLD co-ordinates. More detail can be found in Savut [1988]. There are two routines for force/torque information transmission. Real-time compliant motion

control cuts out the supervisory control computer and stores a large amount of force/torque information temporarily in the PUMA terminal. The actual transmission takes place as follows:

The supervisory computer first sends the dimension number by calling the routine *Send_number* to VAL-2, asking which set force/torque information is requested. VAL-2 stores this number, and sends the corresponding dimension of force/torque data back to the supervisory computer when it demands so by calling the routine *Get_Force*. Every time when *Get_force* is called, *Send_number* has to be invoked before it, otherwise the old dimension of force/torque information will be received.

Real-time compliant motion control, with the MODULA-2 routine being called *RTForceCtrl*, is just for sending the command to tell VAL-2 to invoke real-time path modification using internal ALTER. When internal ALTER ends (which is explained in the next section), supervisory control can be re-established.

The last PUMA interface module is for on-line displaying of force/torque information in supervisory control. A more detailed description of the PUMA interface routine is given in Appendix 7.2.

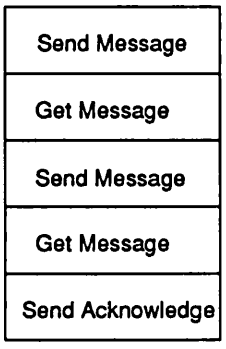



Fig. 7.10 The five step format

Every routine has the same *'five steps format'* (Fig. 7.10). An example in Appendix 7.2 will explain this further. The first *'Send_Message'* sends the command message to VAL-2. The first *'Get_Message'* receives either command message or real data from VAL-2. The second *'Send_Message'* sends either command message or real data to VAL-2.

The second *'Get_Message'* receives real data from VAL-2. Finally an acknowledgement is sent to VAL-2 to indicate that a set of information has been successfully exchanged.

Real-time compliant motion control is incorporated as following: The first *Send_Message* will tell VAL-2 to invoke the path modification process. The supervisory communication is then temporarily cut out until VAL-2 finishes the real-time path modification and sends back a message, so supervisory control is afterwards re-established.

During the execution of the real-time process, the supervisory computer can not get any message from the PUMA560 terminal. A large amount of force/torque information is temporarily stored in the PUMA terminal. There are two routines in the *PumaITF* module to read in the information as described before.

## 7.4 VAL-2 PROGRAM FOR REAL-TIME PATH MODIFICATION

When a robot system controlled by VAL-2 is interfaced to an intelligent (computer-based) sensor, which computes non-time-critical information, the VAL-2 supervisory interface can be utilized to convey the information to the VAL-2 system. However, if the sensor generates data to modify the path of the robotic manipulator while it is moving, the interfacing must be done differently, using the VAL-2 features for 'real-time path control.'

Robotic manipulators are typically directed by user-written programs executing in the VAL-2 system. These programs prescribe the positions and orientations in the work-space the robotic manipulator tool is to move to, the paths to be followed during the motions. This method of control works very well for applications that are totally predictable. However many applications involve some degree of uncertainty in the robotic manipulator locations and motions. Sensory input can be used for such applications to provide information to the robotic manipulator so it can match the changing conditions.

The data for path modification can be obtained from two sources. The data can come from an external computer, in which case 'External Alter mode' is used. The data is transmitted to the VAL-2 system using the serial data protocol described in section 7.4.2. Alternatively, the path modification data can be computed by a second user-written VAL-2 program while the arm is moving. The second program executes in the VAL-2 system in parallel with the robot-control program. This latter method of obtaining the path modification data is called 'internal alter model' and is described in section 7.4.3.

One of the advantages of VAL-2 is its interaction capability with the outside world, which provides an ideal environment to introduce new sensors for the PUMA. Another important feature of VAL-2 is that it allows communication with a host computer via a serial line. The PUMA560 robotic manipulator uses VAL-2 robot programming language. The task that the robot is to perform is completely defined by user written programs. The VAL-2 programming is clear, concise and generally self-explanatory. As well as performing robot motions, VAL-2 can be used to interact with the outside world through A-to-D convertors and binary signal channels, which provide the robot with the ability to respond to sensory information.

### 7.4.1 Program Structure

Fig. 7.11 shows the structure of the VAL-MOD2 communication programs. Before VAL-2 programs start to run, two switches have to be enabled to establish supervisory control by typing in commands EN NET (enable network) and EN RE (enable remote mode). As shown in the figure, moving the PUMA arm in WORLD, JOINT and TOOL co-ordinates only commands VAL-2 to interpret data received from the IBM as WORLD, JOINT and TOOL co-ordinates respectively. When the command is coming from the IBM, requesting real-time compliant motion control, VAL-2 will react as reading force signal and make path modifications according to the contact

force on its own. The supervisory link with the IBM is temporarily cut until the real-time path modification is finished when a NOALTER command is executed. Then the supervisory control is re-established.

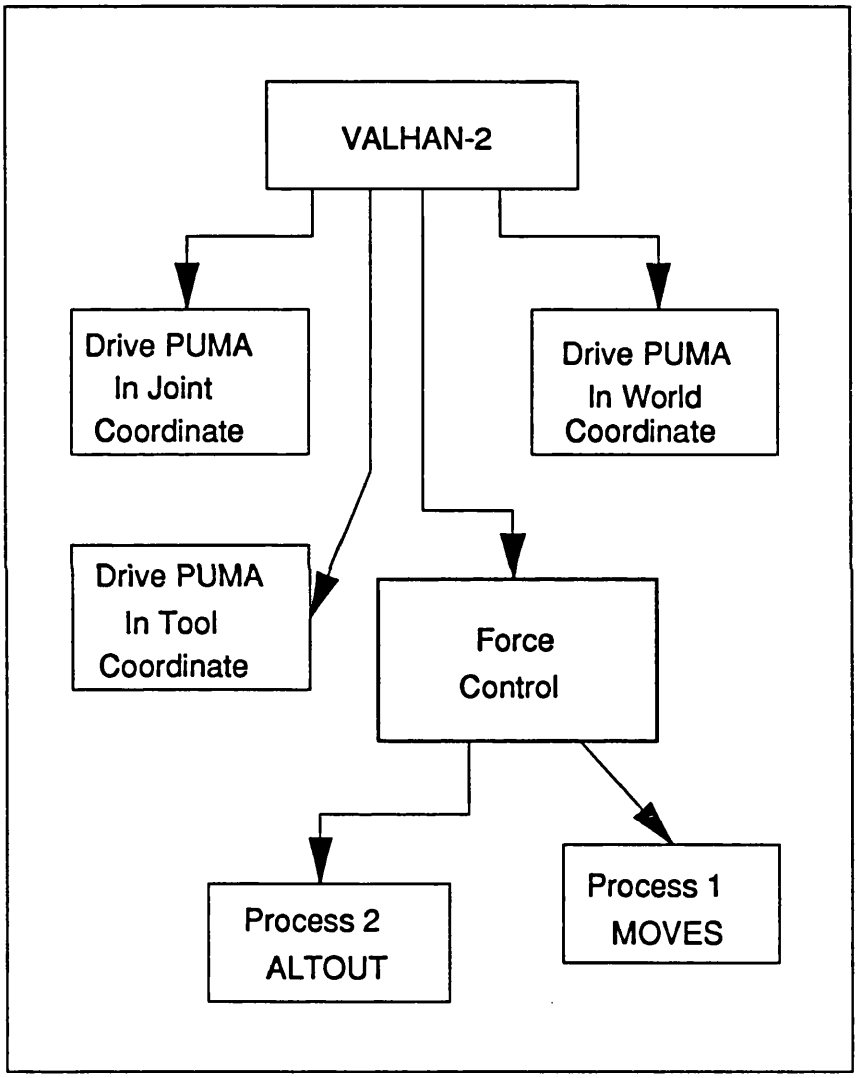



**Fig. 7.11 VAL-2 program structure**

When doing real-time compliant motion control, two processes run concurrently: one is force control directions, the other is position control

directions (as stated before, force control can be divided into position and force control directions). These two group are complementary and perpendicular to each other, which means they can be treated separately.

In Fig. 7.12, the bottom layer for communication is also DDCMP, but it has been hidden from the program by enabling two switches NETWORK and REMOTE. The VAL-2 program to make dialogue with the supervisory computer is called VAL-MOD2. Before executing VAL-MOD2, the operator has to key in 'EN NET' and 'EN RE'.

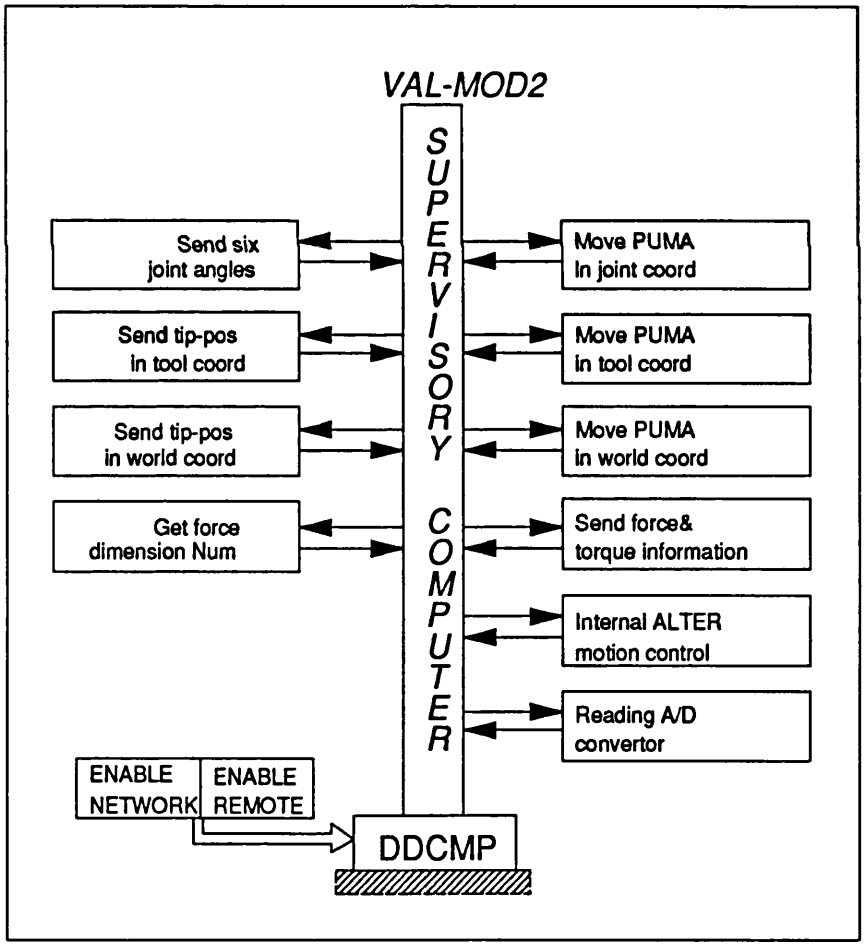



Fig. 7.12 VAL-MOD2 structure frame

This new VAL-2 communication program VAL-MOD2 can incorporate the real-time path modification programs and send large amount of force/torque information to the supervisory computer to carry out analysis using MATLAB. Self-tuning adaptive control is used to control the contact force/torque in the real-time and this is to be discussed in the next section in more detail.

### 7.4.2 External Alter

While in external alter mode, VAL-2 sends a message to the external computer about 36 times every second requesting path modification information. The external computer must respond by sending data that determines how the nominal robot tool trajectory is to be modified. External alter is useful when vision or other sensor based controls are to be incorporated into the control of a robotic manipulator. When real-time path modification is in process, the supervisory computer is temporarily cut out of the connection. This might detune the real-time application when incorporating vision and other sensors (refer to former sections).

The intent of the serial I/O protocol is to provide a high-speed communication link with only enough error checking to detect hardware errors (including noise). This protocol does not provide for any automatic re-transmissions of data if an error occurs. Any such re-transmissions would actually be detrimental to the performance of real-time path control because of the requirement for immediacy of the data being transmitted. Therefore, users should design their systems to be noise free at the level of the physical interface and not rely on error detection to be regarded as a fail-safe mechanism. So, this protocol is high in speed, but is poor in error detection in contrast to DDCMP's slow transmission speed but error free.

### 7.4.3 Internal Alter

Unlike external alter, internal alter causes no messages to be transmitted to an external computer. Instead, VAL-2 expects an ALTOUT program instruction to be executed every 28 milliseconds to pass control data to the robot motion controller.

There are two path modification modes: WORLD or TOOL co-ordinates. When WORLD mode is selected, any data received is interpreted in the WORLD co-ordinate system, which is fixed to the base, of the robot. When TOOL mode is selected, any data received is interpreted in a co-ordinate system attached to the robotic manipulator tool, which is initially aligned with the end flange of the robotic manipulator. This tool system can be positioned and rotated by specifying an appropriate transformation in a TOOL command or instruction, as demanded.

Two other modes of path modification should be distinguished: CUMULATIVE or NON-CUMULATIVE alter. In the first mode, the effect of any data received is accumulated and the robot location is modified by the sum of all past alter data. On the contrary, in NON-CUMULATIVE mode, the robotic manipulator location is modified only by the most recent data. This is also true for the external alter case.

The syntax of the ALTOUT instruction is:

ALTOUT <exception>, {<dx>},{<dy>},{<dz>},{<rx>},{<ry>},{<rz>}

where the three displacement components dx, dy, dz and three angular displacements rx, ry, rz have to be scaled by TODIS and TOANG respectively before being input to the command ALTOUT.

As path modification is a real-time control problem, the ALTOUT instruction has to be put into a separate VAL-2 routine, which is running as a process by calling in the main program using the command PCEXECUTE.

### 7.4.4 Real-time Compliant Motion Control Program

Appendix 7.1 lists VAL-2 programs for real-time path modification. The program flow chart is shown in Fig. 7.13. This program is a sub-module being called from the main VAL-2 program VAL-MOD2, described in sub-section 7.4.1, to do real-time compliant motion control, i.e. path modification.

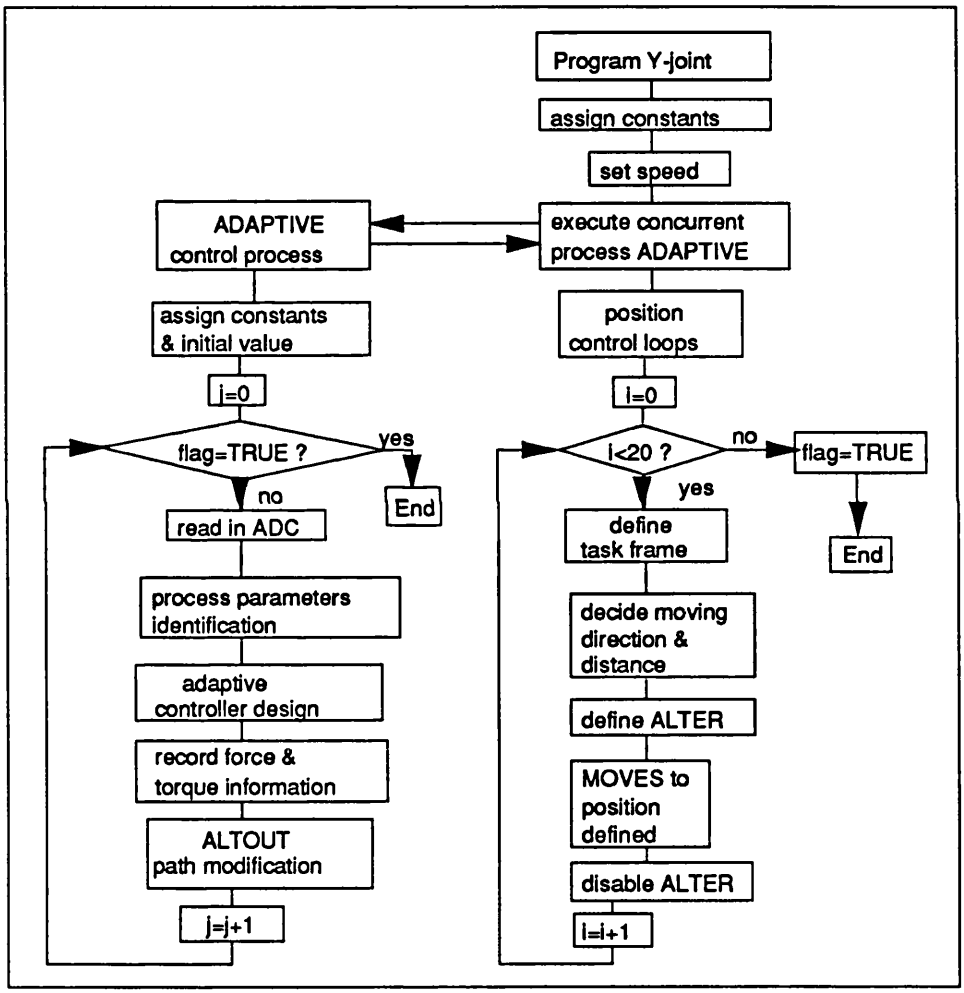



**Fig. 7.13 VAL-2 program for real-time path modification**

There are two processes, one being the position control loop using the VAL-2 command MOVES. The other one is for force control using ALTOUT command. The program *Y-joint* (Fig. 7.13) has 28 lines and *Adaptive* has 84 lines. There are 40 lines other relative programs for reading A-to-D convertors

and error reaction if an error arises. These are not listed in Appendix 7.1. The process parameters identification and the adaptive controller design is the same as described in Chapter 3.

## 7.5 IMPLEMENTATION EXPERIMENTATION AND RESULTS

This section mainly concerns the real controller design based on the analysis described in Chapter 5. Experimental results are presented and analysed. The whole section can be taken as a numerical calculation example of the first four sections in Chapter 5. Some simplification is made in this chapter. The environment is thought to be infinity stiff, which makes the contact dynamics much easier.

### 7.5.1 Introduction

As shown in Fig. 7.14, there is a much less stiff (stiffness k) compliant device assembled between robotic manipulator end-effector and environment. Because of this, it is possible to concentrate on the dynamics of contact itself and ignore the manipulator dynamics as stated in Chapter 5. The function of the manipulator is only to provide the right position/orientation inputs to the end-effector on request. The PUMA560 controller and hardware have been fixed and it is difficult to interfere with the hardware.

In Fig. 7.14, the tracking direction is supposed to be **x** in the tool co-ordinates of the robotic manipulator. The task frame is designed so that the desired tracking force/torques can be met, i.e. **Fz** = 1.2 Newton, **Tx** = 0 and **Ty** = 0. As stated before the robotic manipulator controller (VAL-2) updates its co-ordinates to the task frame by the command HERE to make the tool co-ordinates always coincide with the task frame. The task frame can be stored for use next time. Whole the motion variables are relative to the task frame.

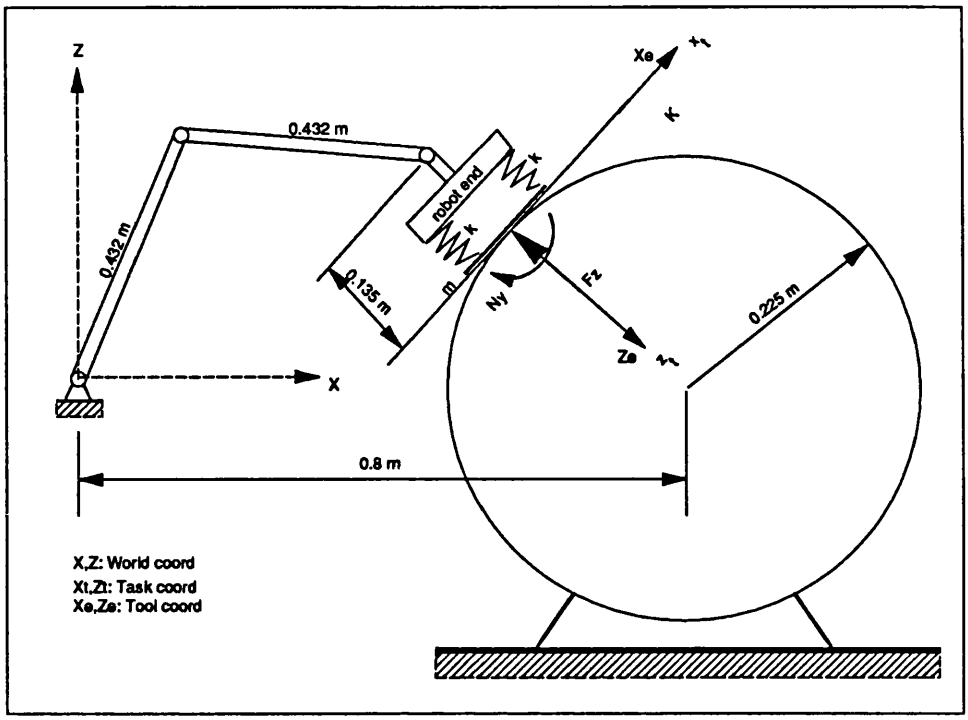



**Fig. 7.14 Tracking direction**

Fig. 7.14 shows a three DOF planery manipulator. It can be seen as the projection of Fig. 7.3 in the X-Z plane. So the three joints represent joint 2, 3 and 5 of the PUMA560 robotic manipulator supposing other joints have been frozen. The figure is only for illustration, and it is not drawn to scale. The environment is the intersection of a sub-sea pipe. To make things easy only two dimensions are taken into account. The contact between the probe, which is attached to the robot end-effector, and the environment can then be simplified as a line contact (instead of a surface contact) as described in Section 5.2. The probe is demanded to always make normal contact with the environment surface and move around the surface (in the $\mathbf{x}_e$ direction as shown in Fig. 7.14) smoothly applying a constant force.

As there is friction between the two surfaces, the $\mathbf{z}_t$ direction of the task frame is not perpendicular to the contact surface. Also the $\mathbf{x}_t$ task direction will be slightly away from the tangent line of the circle. When the friction and compliance in the environment is neglected, the task frame can be defined as

fixed to the environment object (so called object frame). The task frame (or, object frame if no friction and compliance present) changes as the end-effector proceeds along the contour, and since the shape of the contour is also unknown a priori, the object frame or task frame is also unknown to the controller. The problem to be solved is to define the task frame, and to specify its trajectory. The solution adopted is to record the tool co-ordinates of the robotic manipulator to get the task frame.

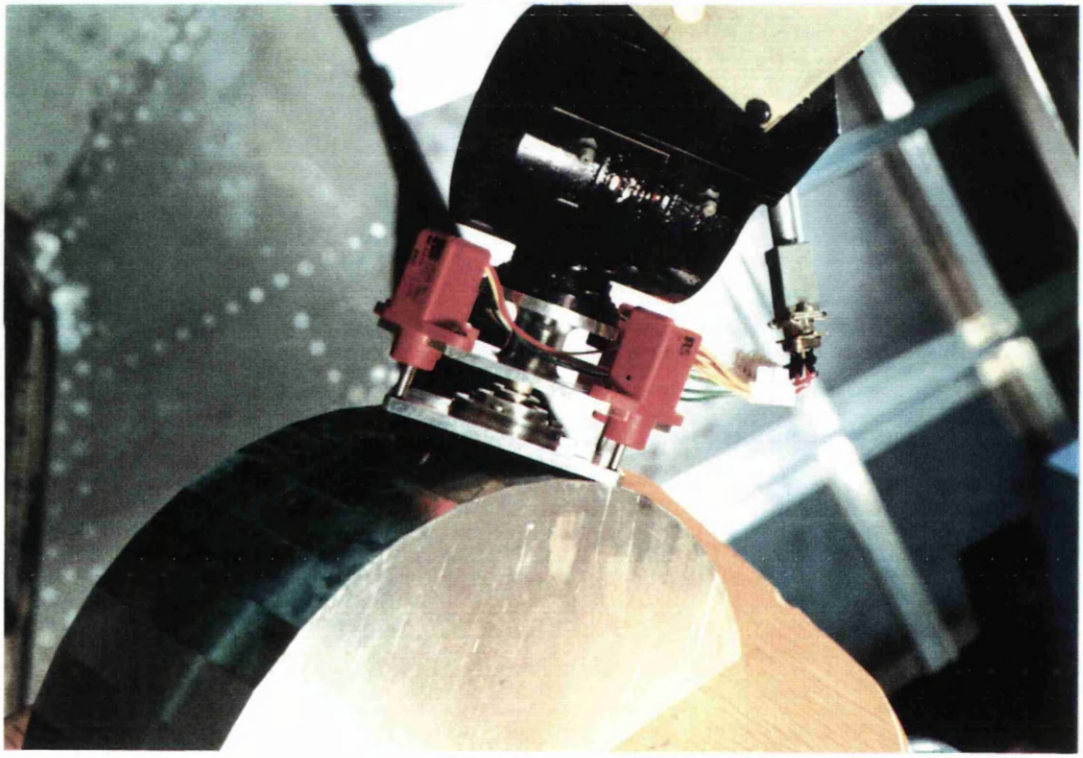


**Fig. 7.15 Picture to show experiment**

Fig. 7.15 shows the experiment carried out at University College London. The picture was taken when the probe hit the surface and at the same time adjusting itself to make full contact with environment. A video was recorded to show the whole successful tracking process.

### 7.5.2 Discretize Model of Contact

As described in the last chapter, there are four positional sensors at each corner of the compliant device. The amount of contact is judged by the average of the four position sensors' reading: $F_z = (d_1+d_2+d_3+d_4)/4$. The four position sensors are so arranged that position 1 and 3 are on the axis x of the manipulator tool co-ordinate, and position sensor 2 and 4 are align with y axis of the tool co-ordinate. So, $N_x = 0.52(d_2-d_4)$ and $N_y = 0.52(d_1-d_3)$. As there is no force sensor presented and only position/orientation are used to judge the amount of contact. Force/torque are transferred from position displacement by multiplying by the stiffness. So the contact and the inputs, which are displacements of the robotic manipulator end-effector, has one-to-one relationship.

As discussed in Chapter 5, compliant contact is of second order (refer to equation (5-3), (5-5) and (5-6)). However, if the environment can be taken as relatively infinitely stiff, the system will be reduced to first order. In the surface contact case:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \cdot \begin{bmatrix} F_z \\ N_x \\ N_y \end{bmatrix} = \begin{bmatrix} d_z \\ r_x \\ r_y \end{bmatrix} \qquad (7\text{-}1)$$

As discussed in Chapter 5, if there is no friction in the ball joint and the end-effector does not move around the unknown surface, the off-diagonal elements in equation (7-1) would be zeros, i.e. the contact is de-coupled. However, in the real world friction does exist and also the probe has to move around. So the contact of surfaces is presented by three coupled equations although the off diagonal elements may be small.

### 7.5.3 The VAL-2 Digital Controller

The controller design is the same as used in many industrial applications and supposes that the contact is de-coupled, though it is a coupled problem as equation (7-1). Each direction is controlled by a single digital PID controller. They have the form:

$$\begin{aligned} d_z &= d_z + s_0(F_d(k) - F_z(k)) \\ &\quad + s_1(F_d(k-1) - F_z(k-1)) + s_2(F_d(k-2) - F_z(k-2)) \\ r_x &= r_x + s_0(N_d(k) - N_x(k)) \\ &\quad + s_1(N_d(k-1) - N_x(k-1)) + s_2(N_d(k-2) - N_x(k-2)) \\ r_y &= r_y + s_0(N_d(k) - N_y(k)) \\ &\quad + s_1(N_d(k-1) - N_y(k-1)) + s_2(N_d(k-2) - N_y(k-2)) \end{aligned} \tag{7-2}$$

Combining (7-1) and (7-2) will result in a closed-loop system of third order denominator: $T(z^{-1}) = 1 + t_1 z^{-1} + t_2 z^{-2} + t_3 z^{-3}$. If $t_1$, $t_2$ and $t_3$ are carefully chosen to ensure all poles to be within the unit circle, the system will be stable. One example of such a stable system is $t_1$=-0.9, $t_2$=$t_3$=0.

The final relationship between the process parameters and the controller has the following form:

$$\begin{aligned} s_0 &= \frac{1 + t_1}{c_{ii}} \\ s_1 &= t_2 / c_{ii} \\ s_3 &= t_3 / c_{ii} \end{aligned} \tag{7-3}$$

For the poles chosen before, $s_0$=0.1/$c_{ii}$ and $s_1$=$s_2$=0 from equation (7-3). Where $c_{ii}$ are diagonal elements of equation (7-1) and are to be identified.

### 7.5.4 Experimental Results and Analysis

Fig. 7.16 shows fixed gains PID, supposing $c_{ii}$ equals 1 in equation (7-3), control of the contact force/torques. In the figure, the first one is the contact force compared with the desired one, which is 1.2 Newton. Number two is the twist torque around x axis of the tool co-ordinate and number three is for y axis (both have desired values 0 Nm). The desired force (1.2 N) and torques (0 Nm) are set as the same in the whole project. The fourth plot in Fig. 7.13 is of cumulative errors, where the solid line is contact force cumulative error, the dashed line is x direction twist torque and the dotted line, twist torque cumulative error in y direction. The cumulative errors are calculated by: $e = \sum_{i=1}^{500} (f(i) - f_d)^2$ where f stands for contact force/torques and $f_d$ the desired ones. The sampling period is 28 ms, which is designed by Unimate company. Five hundred of samples are chosen to analyse.

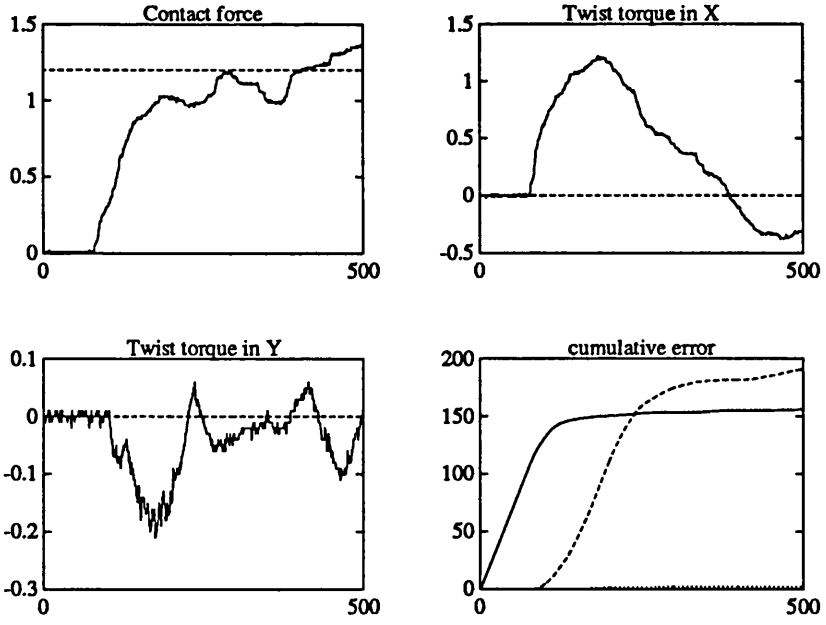



Fig. 7.16 Fixed gains force control results

What has been done in Fig. 7.16 is supposed that whole the off-diagonal elements are zero and all the diagonal elements have unit values, i.e. $c_{11}=c_{22}=c_{33}=1$. In the real situation, this is not true. First of all, the off-diagonal elements in equation (7-1) will not be zero as there is friction in the ball joint and also the probe has to move around the environment surface. However, these off-diagonal elements are difficult to estimated, adaptive control is applied by assuming that the diagonal elements are changing and are to be identified. And so the coupled effect is thought of being included in the diagonal element $c_{ii}$.

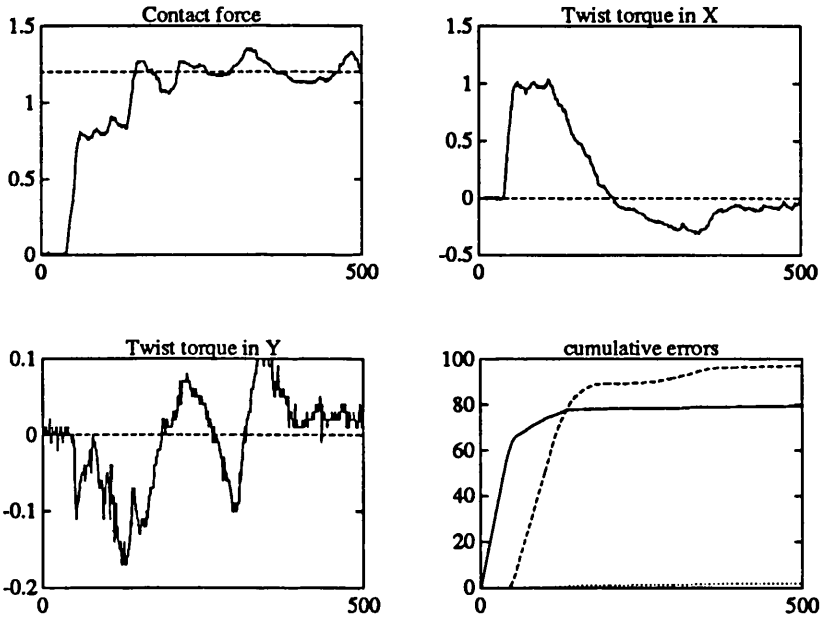



**Fig. 7.17 Adaptive force control results**

Fig. 7.17 shows adaptive control (self-tuning PID) of the contact force/torques. In the figure, the first one is the contact force compared with the desired which is 1.2 Newton. Number two is the twist torque around x axis of the tool co-ordinate and number three is for y axis (both has desired values 0 Nm). The desired force (1.2 N) and torques (0 Nm) are set the same as before. The fourth plot is of cumulative errors, where the solid line is contact force cumulative error, the dashed line is x direction twist torque and the dotted

line, twist torque cumulative error in y direction. Much better tracking has been shown. From the cumulative errors, it can be seen that (comparing with Fig. 7.16) 50 percent improvement has been achieved by using adaptive control.

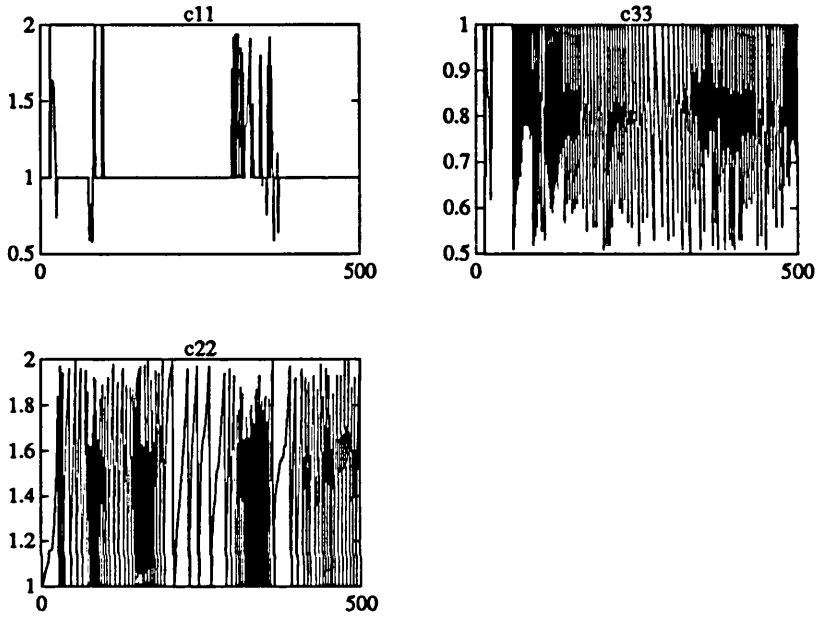



**Fig. 7.18 Parameters identified**

Attention has to be drawn to the identification of the process parameters, which are the spring stiffnesses. There are two ALTER modes in VAL-2, one being *cumulative* mode, and the other *non-cumulative* mode as stated in sub-section 7.4.3. It is experienced that *cumulative* mode ALTER should be used for the real-time path modification. In a *non-cumulative* ALTER mode, it is easy for the arm to bounce against the environment. Fig. 7.18 shows the parameters identified. These parameters are re-set around 1 if they are less than 0.5 or greater than 2. As can see from Fig. 7.18, $c_{22}$ is greater than 1. This means that the rotational change around x axis is less as the controller gain in equation (7-3) shows. However, $c_{33}$ is less than 1, resulting in larger control

gains in the y direction rotation. Different gains are required according to the contour of environment to achieve better tracking. Larger shape changing needs larger gain.

## 7.6 SUMMARY

The whole PUMA supervisory control program package is written in MODULA-2. The most important feature of MODULA-2 lies in modules which enables a program to be organized as a group of semi-autonomous units (black boxes, King [1989]). As a newly created language designed by Niklaus Wirth (Wirth [1985]) of PASCAL fame, MODULA-2 combines the best ideas that are found in its predecessors (FORTRAN, PASCAL, C, ADA etc.).

The hybrid impedance force/position controller is implemented on the PUMA560 industrial robotic manipulator using VAL-2. The communication between VAL-2 and MODULA-2 is accomplished using DEC protocol DDCMP (Digital Data Communication Message Protocol). The real-time path modification is accomplished with a sampling period of 28 ms. After the formalisation of task specification as discussed in section 7.2 and force controller design as in Chapter 5, force controlled directions are controlled using the VAL-2 command ALTOUT and position controlled directions are controlled by MOVES commands. VAL-2 distinguishes these two by putting them into two different groups, which run concurrently as described in the former section.

A control strategy is defined which offers an entirely automatic control solution for a compliant task specification. This strategy is based on external force loops closed around the robot positioning system. The VAL-2 language of the PUMA560 offers a simple solution to the problem.

As the results of this research, it is understood that successful force control requires a passive compliance in order to overcome the limited position resolution and to obtain an acceptable execution speed and disturbance

rejection. Multidimensional force control and control of general compliant motion is performed by using a set of independent one-dimensional control laws, provided that the position loops achieve a sufficient de-coupling of the robot manipulator kinematics.

The initial inspiration for the work was to test self tuning adaptive controller design. The aim is to use an existing robotic manipulator to do as many tasks as possible. In the case of an infinitely stiff environment, the dynamics of contact become much easier. The work described in this thesis can be extended to applications in robotic deburring, flexible manufacturing, automatic assembly (peg-into-hole problem) and general tasks requiring contact or surface following.

The PUMA560 robotic manipulator uses the conventional PID controller with each joint being treated separately. It may appear odd to use self-tuning adaptive control for the end-effector and environment contact force control. The opposite argument is that self-tuning control is used for adapting the unknown environment.

In this control, position controlled directions and force-controlled directions are separated into two processes which run concurrently. The command ALTOUT is used for force control while the MOVES command is used for position control. Equipped with these commands, VAL-2 is particularly suitable for the hybrid force/position control problem. No assumption is made about the approaching phase. The robot manipulator does not need to reduce speed in the transition phase between motion in free space and motion in contact with the environment.

The effect of the contact force on the position loop may also be neglected in many industrial circumstances because (Schutter [1988]):

1). The stiffness k is usually small compared to the servo stiffness.

2). The ir-reversibility of most joint drive systems prevents the passing of end-effector force back to previous links in the kinematic chain.

The following conclusions can be drawn:

- As there is no change made to the existing robotic manipulator hardware, the active force control scheme presented here is fairly easy and cheap to implement.
- The work can easily be extended to other uses like deburring etc., where it is highly desirable to maintain a constant force normal to the grinding surface. Deburring tasks have many thing in common with inspection tasks.
- One of the most popular, general purpose made, industrial robotic manipulators — Unimation's PUMA560, with VAL-2, is versatile and flexible in many tasks providing with simple sensors.
- Existing robotic manipulators, which are controlled using constant gains PID controller with each joint being treated individually, can achieve many tasks within their ability, that is, at reduced speed and with slow configuration changes.
- With self-tuning adaptive control, many traditional controller designing burdens can be avoided. The only thing needs to know is the order of the dynamic system to be controlled. If PID controller forms the basis of classic control theory, self-tuning PID is the most promise hybrid of dynamics and control.
- With adaptive compliant motion control, much better tracking can be accomplished, quality of contact can largely be improved.

## Appendix 7.1 Adaptive Force Controller in VAL-2

```
. PROGRAM Y-joint
1           ATTACH
2           mode=17
3           fd=1.2
4           PCEXECUTE adaptive, -1, 0
5           SPEED 30 MMPS ALWAYS
6           x=0
7           y=0
8           FOR j=0 TO 19 STEP 1
9                 y=6
10                HERE task-frame
11                ALTER (-1, mode, sub, 1)
12                .....
```

```
16              END
17              ......
26              flag=TRUE
27              PCEND
28              RETURN
. END


. PROGRAM adaptive
1               flag=FALSE
2               c11=1
3               c22=1
4               c33=1
5               Kc=0.1
6               f1=0
7               tx1=0
8               ty1=0
9               f=0
10              tx=0
11              ty=0
12              Px=100
13              Py=100
14              Pz=100
15              dx=0
16              dy=0
17              dz=0
18              rx=0
19              ry=0
20              rz=0
21              except=0
22              i=0
23              DO
24                      CASE PENDANT(1) OF
25                       VALUE ? 1:
26                          except=-1
27                       VALUE ? 10000:
28                          rz=-0.2
29                       VALUE ? 20000:
30                          rz=0.2
31                       ANY
32                          CALL wadac
33                          βx=0.98+Px*rx*rx
34                          γx=Px*rx/βx
35                          e_x = tx - tx1 - c22*rx
36                          c22=c22+γx*e_x
37                          Px=(1-γx*rx)*Px/0.98
38                          tx1=tx
39                          tx=0.52*(d4-d2)
40                          rx=Kc/c22*(0-tx)
41                          ......
58                          ty1=ty
59                          ty=0.52*(d3-d1)
60                          ry=Kc/c33*(ty-0)
61                          ......
72                          f1=f
73                          f=(d1+d2+d3+d4)/4
74                          dz=Kc/c11*(1.2-f)
75                          ......
80                      END
```

```
81                  i=i+1
82                  ALTOUT except,(0),(0),(dz*TODIS)
              ,(rx*TOANG),(ry*TOANG),(rz*TOANG)
83            UNTIL flag==TRUE
84            RETURN
. END
```

## Appecdix 7.2 PUMA560 Interface MODULA-2 Programs

### IMPLEMENTATION MODULE PumaITF;

```
(* Modules for talking between MODULA-2 and VAL-2 by Q Wang Feb., 1990. And revised
in June 1991. Original created by M Savut in PASCAL *)
IMPORT IO, FIO, Window, WdHandler;
(* ========================================================= *)
PROCEDURE PumaInit(VAR Status : CARDINAL ) ;
VAR
      LUN,FnCode,FnQual: CARDINAL ;
      PumaMsg: ARRAY[0..80] OF CHAR ;
BEGIN
 rs.Install(1);
 rs.Init(9600, 8, rs.None, TRUE, FALSE);
 REPEAT
      DDCMP.GetMessage( LUN, FnCode, FnQual , PumaMsg, Status ) ;
      DDCMP.SendAck();
 UNTIL (LUN = 4) AND (FnCode = 4) AND  Str.Match( PumaMsg, '0*') ;
 (* LUN =4 means program terminal input and output
   FnCode=4 read data after writing data 8/4/90*)
END PumaInit;
(* ========================================================= *)

... ...

(* ========================================================= *)
(* Procedure to receive large amount of force/torque information from PUMA terminal after
real-time path modification ended.  Q. Wang 3/7/91 *)
PROCEDURE GetForceTorq ;
VAR
      i,LUN,FnCode,FnQual, Status,
      MsgLength        : CARDINAL ;
      Message          : ARRAY[0..80] OF CHAR ;
      OK               : BOOLEAN ;
      TempString       : ARRAY[0..10] OF CHAR ;
      F                : ARRAY[1..3] OF REAL;
BEGIN
 LUN:= 4 ;
 FnCode:= 132 ;
 FnQual:= 1 ;
 Message := '6' ;(*  Command for setting XYZs in VAL-II  *)
 DDCMP.SendMessage( LUN, FnCode, FnQual, Message, Status ) ;
 DDCMP.GetMessage( LUN, FnCode, FnQual, Message, Status ) ;
 IF ( LUN = 4 ) AND ( FnCode = 3 ) THEN
  FOR i:= 18 TO 80 DO
    Message[i]:= '';
```

```
  END;
  FOR i := 1 TO 3 DO
   TempString := '' ;
   Str.Item( TempString, Message, Str.CHARSET{' '}, (i-1) ) ;
   F[i] := REAL(Str.StrToReal( TempString , OK )) ;
  END;
  ForceZ:= F[1];
  TwistX:= F[2];
  TwistY:= F[3];
  FnCode:= 132 ;
  FnQual:= 1 ;
  Message:= '';
  DDCMP.SendMessage( LUN, FnCode, FnQual , Message , Status ) ;
  DDCMP.GetMessage( LUN, FnCode, FnQual, Message, Status ) ;
  DDCMP.SendAck ;
  Status:= CARDINAL(Message[0]) - 48 ;
  (* include letters as well, i.e. 'A'=10 and so on.. *)
  IF Status > 9 THEN
   Status:= Status - 7 ;
  END;
  Window.Use(WdHandler.FreeWind);
  IO.WrStr('Status='); IO.WrCard(Status,2);IO.WrLn();
 END;
END GetForceTorq ;
(* ========================================================== *)

... ...

END PumaITF.
```

# CHAPTER 8
# CONCLUSIONS AND SUGGESTION FOR FUTURE WORK

## 8.1 SUMMARY OF WORK CARRIED OUT AND CONCLUSIONS

The work described in this thesis consists of the research carried out during the last three full-time years at University College London. There are four main aspects of robotic research: 1). Mechanical Manipulation and Control; 2). Vision; 3). Locomotion; and 4). Artificial Intelligence. This work belongs to the first group and three main topics are covered.

The first one is the dynamics model, which forms the basis of dynamic simulation, of a robotic manipulator. In dealing with dynamics, kinematics has to be referred to as they are closely related. In Chapter 2, a new dynamic model of the PUMA560 was established using Horak's idea of partitioning a robotic manipulator into two parts, an arm and a hand. As usually practiced in robot kinematics, this partitioning reduces mathematical calculation a great deal. This model was not actually used for simulation in this research, but rather a simple two DOF manipulator is used extensively. However, this work provides good understanding of manipulator dynamics and serves as basis for further research.

The second area is dynamic control of robotic manipulators, which is presented in Chapter 3 and Chapter 4.

The last topic is compliant motion control of robotic manipulators making contact with the environment. This topic is covered in Chapter 5, Chapter 6 and Chapter 7. Two practical applications of robotic compliant motion control were presented in Chapter 6 for robotic assembly and Chapter 7 for robotic inspection respectively.

The main work lies in the second and the third topic. It is worth giving each separate chapter a description, and some main conclusions are also summarised in the following.

Chapter 3 mainly deals with self-tuning adaptive control theory based on SISO systems. An approximate recursive least square (ARLS) identification method was introduced. This method can considerably reduce calculation and comparison was made with normal recursive least square (RLS). Pole-assignment self-tuning adaptive control was then described. A new self-tuning PID controller based on Tustin's bi-linear transform was acquired. Section 3.4 shows a new control scheme for non-linear systems. This scheme can identify the typical parameters of a highly non-linear term. Craig [1986] uses Lyapunov strategy for non-linear parameter identification. The recursive non-linear identification is based on conventional least square method, which offers a simple solution.

Chapter 4 is for the extension of what is described in Chapter 3 to the practical application for robotic manipulators, ie the MIMO systems. It has been found that the direct use of self-tuning adaptive control for such highly non-linear systems as robotic manipulators will result in highly chattering control inputs. A solution to this phenomena was attempted, and it was found that slightly increase the forgetting factor will reduce control input chattering. A new simulation scheme, backward simulation, was introduced in section 4.5. This scheme can make direct use of Newton-Euler dynamic equations, which are traditionally thought of as inconvenient for simulation of robotic manipulator control systems.

Because of this research, it is understood that the choosing of PID controller's gains can be automated according to different payload and manipulation speed. This is much more advanced than the commercial robot without adaptation in the control gains. And for dedicated work conditions, ie specific payloads, manipulation speeds and trajectory, a fixed gain PID controller can be good enough as the control system. With self-tuning adaptive control, many traditional controller design burdens can be avoided. The only thing that needs to be known is the order of the dynamic system to be controlled. If PID controller forms the basis of classic control theory, self-tuning PID is the most promising hybrid of *dynamics and control.*

The topic of robotic compliant motion control has been very active in recent years. One reason for using robots in the place of human operators is to remove men from potential hazards such as in undersea, nuclear or defence applications. Such tasks are often conducted remotely and tele-operation using cctv visual feedback is limited due to lack of depth perception. Also the task environment is often disorganised unlike in advance automation where work is laid out and simple pick and place or assembly can be performed by a sequence of simple operations. These tasks, such as inspection, cleaning or deburring, are of contact type, which can be achieved by robotic compliant control. Chapter 5, 6 and 7 are devoted to robotic compliant motion control.

In Chapter 5, robotic contact dynamics are detailed with the emphasis being placed on surface contact which is very common in inspection manipulation and robotic deburring. The concept of coupled concept was introduced for the first time for robotic contact between two surfaces. For an unknown environment with unspecified geometrical configuration, adaptive compliant motion control proved to be necessary and much better tracking ability can be achieved by adaptive control strategy. For a flexible environment, a second order feature is typical for contact dynamics. For a stiff environment the contact

reduced to first order; adaptive control is still necessary as the coupled contact feature and changing geometry as the robotic end effector moves around the environment surface to carry out inspection, or deburring operations.

In section 5.5 of Chapter 5, manipulator dynamics together with contact dynamics are jointly taken into consideration. Simulation was carried out and some conclusions were presented.

Chapter 6 deals with a few ancillary topics. Simulation packages for robot gross motion and fine motion control both on a VAX work station and on IBM PCs are described. Joint space trajectory planning is detailed. Real-time experiment for robotic gross motion control and fine motion control (assembly) was carried out and detailed. This was made possible with a designed compliant device, whose mechanical structure and measurement of compliances are detailed in section 6.6.

Robotic compliant motion control has been attracting interests of many researchers recently. Chapter 7 is mainly concerned with adaptive robotic compliant motion control for sub-sea inspection, which can be thought as an example of the use of what is described in Chapter 5. A supervisory control system with window menu and multi-process support, capable of incorporating real-time compliant motion control was detailed in this chapter. Implementation of self-tuning adaptive control on the PUMA560 system in VAL-2 was described and some programmes were listed for easy reference.

As a result of this research, it is understood that successful force control requires a passive compliance in order to overcome the limited position resolution of the existing robot manipulator and to obtain an acceptable execution speed and disturbance rejection. The implementation of the system is inexpensive as there is no hardware change made to the manipulator. The experimental work can be easily extended to assess robotic deburring.

The compliant end-effector has the potential to 'short circuit' the process of object modelling, trajectory calculation, arm motion planning and execution. For a wide range of geometries and tasks it will be possible to simply 'point' the force compliant probe in the required direction, let it follow the surface contours and redirect it occasionally to complete a tracking operation.

As an example let us consider deploying an inspection probe around the intersection weld between two cylindrical members. The methods based on component modelling, either tactile or vision, require a definition of the location and size of each cylinder, a calculation of the intersection path and deployment of the probe around it. Errors occur in the object sizing and location procedures due to calculation of intersection, differences between 'real' welds and cylinders, arm accuracy etc. The compliant end-effector method relies on maintaining physical contact between probe and cylinder, and the tracking round the toe of a weld, remaining in contact can be performed 'blind' ie without any absolute knowledge of the path to be followed.

The results of this thesis have been used as the basis of a £2.6m, 22 man year application to the recent ESPRIT computer integrated manufacture/mechanics initiative called TRACK, for Tracking Robot with Adaptive Contact Kinematics.

## 8.2 SUGGESTION FOR FUTURE RESEARCH

In section 3.4, a new adaptive control scheme using non-linear system identification was presented. This scheme has shown good control performances for SISO systems, as shown. How to extend this scheme for MIMO systems deserves further research.

Chapter 2 presents a dynamic model for PUMA560 robotic manipulator. Simulation of the whole manipulator dynamics was not actually carried out. An obvious extension of this research is to simulate the whole manipulator dynamics, possible with a graphics display to view the dynamic motion of the

robotic manipulator. This work would combine the PUMA kinematics package, which already exists in the Automatic Control Group, University College London.

There is a tendency to make cheap and reliable robotic manipulators. Tactile sensing provides robots with a cheap and reliable means to make contact with the environment. In sub-sea robotic inspection, vision system can be easily detuned by muddy, water due to tides or thruster activity on the ROV. However, these situations can be avoided by using tactile sensing.

In this thesis, the use of compliant motion control for sub-sea inspection has be described, and it has been concluded that adaptive control can further improve contact quality and provide better tracking ability in the inspection task. It may be assumed that the trajectory of the inspection has been found by various means including vision, sonar etc., or the compliant sensor can 'feel' along artificial or real surface boundaries, such as weld toes. As described in Chapter 5 and Chapter 7, the work is actually surface following in a two dimensional sense.

The inspection is always carried out at the welded part of sub-sea structures, which is usually a cubic trajectory. How to follow a cubic random trajectory using compliant motion control needs further research. If this can be done, vision would be less important than before.

In the two dimensional surface following three DOF compliances are needed as described in Chapter 7. However, in cubic random trajectory following at least four degree of freedom are needed to carry out cubic tracking operation. So a new compliant device, which is of four or five DOF, has to be designed.

As stated in Chapter 6, using the compliant device for robotic assembly will reduce the overall system stiffness, the manipulation speed will subsequently be reduced. In order to overcome this problem, it is to suggested that a higher stiffness force sensor is used to do assembly tasks.

As computers are becoming more and more powerful and cheap, hybrid and impedance force control can be improved and high bandwidth force control can be achieved. The force control combining both manipulator and contact dynamics described in section 5.5 of Chapter 5 deserves further research.

In this thesis, a simple self-tuning PID controller and its use in the outer-loop force control of a industrial robotic manipulator were presented in detail. This control scheme, because of its simplicity and cheapness, is well suited to industrial practice. Implementation of this adaptive PID control is recommended and remains an objective of the authors further research.

# LIST OF REFERENCES


An C. H. and Hollerbach J. M., [1989], "The Role of Dynamic Models in Cartesian Force Control of Manipulators", *The International Journal of Robotics Research*, vol. 8, No 4, pp.51-72

Armstrong B. etc, [1986], "The explicit dynamic model and inertia parameters of the PUMA560 arm", *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.510-518.

Asada H. and Slotine J. J. E. [1986], *Robot Analysis and Control*, New York: Wiley, 1986.

Astrom K. J. and Wittenmark B., [1989] *Adaptive Control*, Reading: Addison-Wesley, 1989.

Besant C. [1986], *Computer Aided Design and Manufacture*, Chichester: Horwood, 1986, Ellis Horwood Series in Engineering Science.

Bone G.M. and Elbestawi M.A., [1989], "Robotic Force Control for Deburring Using an Active End Effector", *Robotica*, vol. 7, pp.303-308

Broome D. R., Ihnatowicz E., Wray A. M. and Savut M. K., [1984], "Automated inspection of sub-sea structures", *Proc. Sub-sea 84*, Rotterdam, November.

Broome D. R., Greenshields M. C., Hughes G. and Savut M. K., [1987], "Computer Control of GE Underwater Manipulator System", *International Underwater System Design, 9*, No. 5.

Broome D. R. and Hughes G., [1988], "Automated Development of Inspection Systems for Tubular Nodes", *INTERVENTION 88*, Bergen.

Broome D. R. and Wang Q., [1991], "Adaptive Control of Underwater Robotic Manipulators," *Fifth Int. Conf. on Advanced Robotics ('91 ICAR)*, Pisa, Italy, June, 1991, p.1321-1326.

Broome D. R. and Wang Q., [1992], "Persistent Excitation and Adaptive Control of Robotic Manipulators," *Paper submitted to IEEE, Int. Conf. on Robotics and Automation*, Nice, France, May 1992.

Chung C. H. and Leininger G. G., [1990], "Task-level Adaptive Hybrid Manipulator Control", *The International Journal of Robotics Research*, vol. 8, No 4, pp.51-72

Clarke D. W. and Gawthrop P. J., [1975], "Self-Tuning Controller," *Proc. IEE*, vol. 122(9), pp.929-934, 1975.

Craig J. J., [1988], *Adaptive control of mechanical manipulators*, Reading: Addison-Wealey Publishing Company, 1988.

Daniel R.W., et al, [1989], "Inner Loop Sensors for Outer Loop Force Control of Robots", *IEE Computing and Control Colloquium on "Controllers for Robotic Applications---Concepts and Implementations"*, Nov., 1989, London, 7/1-7/4

Dubosky S. and DesForges D. T., [1979], "The application of model-referenced adaptive control to robotic manipulators," *J. Dyn. Sys. Meas. Control*, vol. 101, pp.193-200, 1979.

Eppinger S. D. and Seering W.P., [1986], "On Dynamic Models of Robot Force Control", *Proceeding of the IEEE International Conference on Robotics and Automation*, pp.29-34

Eppinger S. D. and Seering W. P., [1987], "Understanding Bandwidth Limitations in Robot Force Control", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.904-909

Franklin G. F., Powell J. D. and Workman M. L., [1990], *Digital Control of Dynamic Systems*, Addison-Wesley Publishing Company, Chapter5, 1990

Fu K. S., Gonzalez R. C. and Lee C. S. G., [1987], *Robotics: Control, Sensing, Vision and Intelligence*, McGraw-Hill, 1987.

Greenshields M. and Broome D. R., [1989], *Digital Control Systems*, MSc lecturing Manuscript, UCL, Spring 1989.

Greig A. R. and Broome D. R., [1990], "Underwater Applications of Touch Sensors", *Proc. ROV90*, Vancouver, June.

Greig A. R. and Broome D. R., [1991], "Automatic Inspection of Complex Geometry Welds", *IEEE-91 5th Int. Conf. on Advanced Robotics*, Pisa, Italy, June, 1991, pp.1788-1791.

Greig A. R., [1992], "Automatic Inspection of Complex Geometry Welds", *PhD Thesis*, in preparing UCL.

Hogan N., [1985], "Impedance Control: An Approach to manipulation", *ASME J. Dyna. Sys., Measurement and Control*, Vol 107, pp.1-24

Horak D., [1984], "A Simplified Modelling and Computational Scheme for Manipulator Dynamics", *Trans. ASME J. Dyna. Syst., Meas. Control*, Vol. 106, pp.350-353.

Hsia T. C., [1977], *System Identification*, Lexington Books, D. C. Heath and Company, 1977, pp.143

Jebb A. and Wynn H., [1991], "Quality Through Robust Design", *European Journal of Engineering Education*, Vol. 16, No. 2, 1991, pp143-151.

Karam K. Z. and Warwick K., [1989], "A microprocessor based adaptive controller for robotic manipulators," *IEE Colloquium London*, pp.4/1-4/4, Nov., 1989.

Kazerooni M., [1989], "Compliant Motion Control for Robot Manipulators" *International Journal of Control*, vol. 49, No. 3, pp.745-760

King K. N., [1989], *Modula-2 for IBM and Compatibles, Language Tutorial*, Jensen & Parternal International.

Koivo A. J. and Guo T. H., [1983], "Adaptive linear controller for robotic Manipulators," *IEEE Trans. Automat. Control*, vol. AC-28(2), pp.162-171, 1983.

Kreyszis E., [1979], *Advanced Engineering Mathematics*, John Wiley & Sons, Inc., 1979.

Leininger G., [1983], "Adaptive control of manipulators using self-tuning methods," *International Symposium of Robotic Research*, 1983 (August-September, Bretton Woods, New Hampshire).

Liu M. H., Chang W. S. and Zhang L. Q., [1988], "Dynamic and Adaptive Force Controllers for Robotic Manipulators", *Proceeding of the IEEE International Conference on Robotics and Automation*, pp.29-34

Liu M. and Lin W., [1987], "Pole assignment self-tuning adaptive control for robotic manipulators," *Int. J. Of Control*, vol. 46, pp.1307-1317, 1987.

Ljung L. and Soderstrom T. [1983], *Theory and Practice of Recursive Identification*, The MIT Press, Cambridge, Mass., 1983, pp.12-23.

Lovell S., [1990], "Automatic Air Craft Detection", *PhD Thesis*, UCL.

Lu W. S. and Meng Q. H., [1991], "Impedance Control with Adaptation for Robotic Manipulators", *IEEE T. Robotics and Automation*, Vol. 7, No. 3. June 1991, pp.408.

Mason M. T., [1981], "Compliance and Force Control of Robot Manipulators", *IEEE Transaction on System, Man and Cybernetics*, vol. SMC-11, No. 6, 418-432

Mon D., [1988], "Computer Control of Robotic Manipulator Dynamics", *PhD Thesis*, UCL.

Northcott J. and Brown C., [1986], *Robots in British Industrial*, 1986, Policy Study Institute.

Ogada K., [1987], *Discrete-Time Control System*, Prentice-Hall International, Inc., New York 1987, pp.213-222.

Paul R. P., [1982], *Robot Manipulators: mathematics, Programming and Control*, The MIT Press, Cambridge, Mass., 1982.

Paul R.P., [1987], "Problem and Research Areas Associated with the Hybrid Control of Force and Displacement", *IEEE Proceedings of the International Conference on Robotics and Automation*, pp.1966-1971

Raibert M. H. and Craig J. J., [1981], "Hybrid position/force control of manipulators" *ASME J. Dyn. Syst., Meas., Control*, vol 102, pp.126-133.

Redford A. and Lo E., [1986], *Robots in Assembly*, Open University Press, 1986, Milton Keynes.

Savut M., [1988], "A Supervisory Manipulator Control System Using Visual Feedback", *PhD Thesis*, UCL.

Savut M. K., Broome D. R. and Greig A. R., [1989], "Automated Inspection System Using Visual Feedback", *Proc. ROV89*, San Diego, March

Schutter J. De. and Brussel H. Van, [1988], "Compliant Robot Motion I. A Formalism for Specifying Compliant Motion Tasks; II. A Control Approach Based on External Control Loops", *The International Journal of Robotics Research*, vol. 7, No. 4, pp.3-33

Seraji H., [1989], "Decentralized adaptive control of manipulators: theory, simulation and experimentation," *IEEE T. on Robotics Automation*, vol. 5, pp.183-201, 1989.

Slotine J. J. E. and Li W. P., [1991], *Applied Nonlinear Control*, Prentice-Hall International, Inc., 1991, pp.291-293.

Stepien T.M. etc, [1987], "Control of Tool/Workpiece Contact Force with Application to Robotic Deburring", *IEEE Journal of Robotics and Automation*, vol. RA-3, No. 1.

Unimate [1985], *Programming Manual — User's Guide to VAL II*, Unimation Incorporated.

Wang Q., [1989], "The Mathematical Model of the PUMA Type Robot Manipulators", *Internal Report*, UCL.

Wang Q., [1990], "A New Explicit Self-tuning Adaptive Digital Controller", *UCL internal report.*

Wang Q. and Broome D. R., [1991a], "A new simulation scheme for self-tuning adaptive control of robot manipulators," *ROBOTICA*, Vol. 9, 1991, pp.335-339.

Wang Q., Broome D. R. and Daycock I., [1991b], "Adaptive force control of the PUMA560 industrial robotic manipulator," *Proc. of CONTROL'91, IEE*, Edinburgh, U.K., March 1991, pp.358-363.

Wang Q., [1991c], "Assessment of an approximate recursive least square identification," *Unpublished internal report*, UCL, April, 1991.

Wang Q., Broome D. R. and Greig A. R., [1992], "Adaptive compliant motion control for robotic sub-sea inspection," *Paper submitted to IEEE, Int. Conf. on Robotics and Automation*, Nice, France, May 1992.

Warwick K., [1984], "A new approach to reduced-order modelling", *IEE Proceedings, Vol. 131, Pt. D, No. 2*, March 1984.

Warwick K., [1988], "Simplified algorithms for self-tuning control", IEE Control *Engineering Series 35: Implementation of self-tuning controller*, Peter Peregrinus Ltd. on behalf of IEE 1988, pp96-124.

Wellstead P. E., Prager D. and Zanker P., [1979], "Pole assignment self-tuning regulator," *Proc. IEE*, vol. 126, pp.781-787, 1979.

Wellstead P. E. and Zarrop M. B., [1991], "*Self-tuning Systems — Control and Signal Processing*", John Wiley & Sons, 1991

Whitney D. E., [1976], "Force Feedback Control of Manipulator Fine Motions", *Proceedings of the Joint Automatic Control Conference*, pp.687-693

Whitney D. E., [1985], "Historical Perspective and State of the Art in Robot Force Control", *IEEE Proceedings of the International Conference on Robotics and Automation*, pp.262-268

Wirth N., [1988], *Programming in Modula-2*, Springer-Verlag, 1988.

Youcef-Toumi K. and Li D., [1987], "Force Control of Direct Drive Manipulators for Surface Following", *IEEE International Conference on Robotics and Automation*, pp.2055-2060

Zheng Y. F. and Fan Y., [1991], "Robot Force Sensor Interacting with Environments", *IEEE T. Robotics & Automation*, Vol. 7, No. 1, pp.156-164.