TREX: Taxonomy-based Robot-control EXpert-system

by

DEJAN Milan MILJANOVIC

B.Sc. (M. Eng.), University of Belgrade, Yugoslavia, 1996

A THESIS SUBMITED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF MECHANICAL ENGINEERING

We accept this thesis as conforming
to the required standards

THE UNIVERSITY OF BRITISH COLUMBIA

August 1999

Abstract

This work presents the basis for an expert system design for robot control. The system is based on a new *taxonomy for robot control.* This taxonomy identifies the main issues in the design and selection of robot controllers and categorizes and relates these issues. A clear distinction is drawn between the selection of robot control architectures and the controller algorithms that are utilized within these architectures. In both cases a wide variety of approaches are examined and evaluated based on characteristics such as disturbance rejection, effective stiffness, allowable planing task error, dynamic motion characteristics, sensor information, hardware requirements and model accuracy. Based on these characteristics the controller architectures and controller algorithms are categorized and evaluated using these major criteria: reliability, complexity and robustness. This review and categorization gives rise to a taxonomy based on desired task, imposed criteria, and the distinctive characteristics of various controller architectures and controllers described in the taxonomy. Based on this taxonomy an open expert system for selecting suitable control architectures and algorithms is developed. The appropriateness of the rules embedded in expert system's inference engine as well as the correctness of the decisions made by the expert system is verified via simulations and experiments. Simulation results are obtained by applying selected control architectures and controllers on a two-link manipulator both for free robot motion and constrained robot motion. Experiments are performed both for constrained and unconstrained motion on a five-axis industrial robot. This system is designed for implementation on a new real-time open architecture controller system. This system has potential applications for industrial robots that are currently limited to either very restrictive proprietary controllers, or complex and specialized controller designs.

Contents

List of Tables

List of Figures

Acknowledgements

- I would like to express my gratitude and appreciation to my supervisor, Dr. Elizabeth A. Croft, for her guidance, encouragement, patience and tremendous support during these two years of my research.

- I would also like to thank Dr. Miomir K. Vukobratovic from the Robotics department of Mihailo Pupin Institute, Belgrade, for making himself always available to discuss any matter and to answer questions that I had.

- My thanks also go to all of my colleagues in the Industrial Automation Laboratory at UBC, for promoting a professional yet enjoyable work environment.

- Special thanks go to my parents and my sister's family for their love and ceaseless support, and my closest friends who have always been there to cheer me up and give me constant encouragement.

Chapter I

"So far as the laws of mathematics refer to reality, they are not certain. And so far they are certain, they do not refer to reality"
-Albert Einstein

Introduction

Robotics has been defined as the science studying the intelligent connection of perception to action [32]. The goal of this introductory chapter is to point out the problems related to the use and control of robot manipulators in industrial applications, with the reference to the general framework of robotics and hierarchical structure of robot controllers.

1.1 Background

In the last decades, robots have become integral components of flexible manufacturing cells, and intelligent technological systems. Intelligent robotics usually refers to the robots with characteristics of autonomy and whose applications are conceived to solve problems of operation in a variety of environments where there is a level of a priori uncertainty about the environment. This uncertainty requires a level of adaptability that is generally higher than achievable at the robot controller level.

The manipulator's action is provided by a mechanical system. The realization of such a system refers to a framework concerning the design of mechanical system, choice of materials, and type of actuators. The manipulator perception is provided by a sensory data. The realization of such a system refers to a scientific framework concerning material science, signal conditioning, data processing, etc. The connection action to perception in an intelligent fashion is provided by a control system. This system selects the appropriate action given the constraints imposed by the mechanical system and the environment. The realization of such a system refers to the framework of cybernetics, concerning artificial intelligence and expert systems, motion control, etc. [32].

In view of these three identifiable frameworks, it is useful to organize a robot control based on principle of control hierarchy. In the following we will consider mainly automatic control regimes (Table 1.1), although these comments can be applied to some extent to biotechnical or interactive control regimes as well.

For the automatic and automated control regimes shown in Table 1.1, various types of hierarchical controllers are generally utilized. Hierarchical controllers can be classified into four basic levels [23]. The highest level of hierarchical control comprises the elements of artificial intelligence. These elements handle the sensor information (visual, tactile, etc.), recognize the situation (environment) where end-effector acts, and provides environmental modeling in some convenient form. This information is compared with the system's database and decisions about the system behavior are adopted.

The next, lower level is the "strategic level", where the planning operation takes place (defining the manipulator position in the workspace, trajectory planning, etc.). At strategic level, trajectory planning is accomplished in the robot's work-space coordinates. At the lower hierarchical level, the so-called "tactical level", coordinates transformation from workspace into joint-space takes place. Finally, at the lowest "executive level", the realization of planned robot task is accomplished.

TYPE	VARIANT
1. AUTOMATIC	- Programmable (First generation) - Adaptable (Second generation) - Intelligent (Robot with artificial intelligence, third generation)
2. BIOTECHNICAL	- Controlled (Control of certain degrees of freedom) - Copied - Semiautomatic
3. INTERACTIVE	- Automated (Combination of automatic and biotechnical regimes) - Dialogical (Different types of communications with operator) - Supervised

Table 1.1. Classification of robot manipulator.

At the "strategic" level, the robotic system is typically constrained by the system controller that is generally specified at the "executive" level. Allowing the "strategic" level to select between controllers utilized at the "executive" level would allow a much greater flexibility at the planning level. That is, the planning level could specify the type of controller system best-suited to the task being planned.

A great deal of work has been done in the area of planning for robot motion [21, 25, 26, 27]. Even more work has been done in the area of robot control, from simple independent joint control to complex robot control architectures. However, without the ability for the planner to specify the controller, much of the advantages of schemes on both sides of the problem, are lost.

In the present work, a taxonomy for robot control is developed. This taxonomy is the basis for the development of an expert system that can appropriately select a robot controller based on identified defining characteristics of a planned task. The objective is to develop an expert system tool implemented on an open-architecture controller platform that provides a flexible link between the "strategic" and "executive" levels of open hierarchical robot controller system.

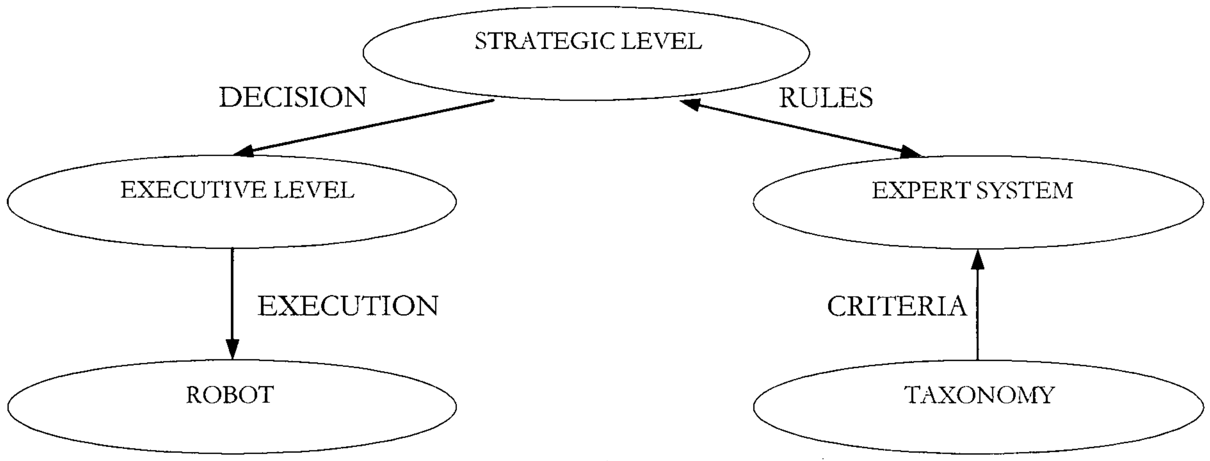


Figure 1.1. Hierarchical structure of intelligent control.

1.2 Industrial Robot Controllers

Since the introduction of modern industrial robotic systems, there has been a substantial increase in the number of robot control architectures and algorithms. However, until recently, the selection of an appropriate architecture/technique was an issue only for non-standard research based robot controllers, or a one-time decision for the designers of industrial robot controllers. In the latter case, a decision for a very robust and simple controller, that did not fully exploit the capabilities of the robot, would be made.

Traditionally, a control system for industrial manipulators is designed as a decentralized system, consisting of a set of local constant-gain PID or PD controllers with feed-forward control signal that are calculated on the basis of a nominal robot model dynamics. In its simplest form such control schemes consist of local servo systems closed around individual joints of the robot. This work is used in the majority of contemporary commercial robots, but it appears satisfactory only for certain class of robot tasks where only position control is required with slow trajectory tracking.

On the other hand, if accurate tracking of high-speed trajectories is necessary such simple control laws do not work well, and various model-based control schemes have been proposed in order to compensate for the effects of the dynamic coupling between joint axes of the robot.

However, until the recent development of open architecture controllers hardware [55], a such schemes have not been widely utilized in industry. The development of these open controllers provides a widely new scope in the implementation of this work in industry. Due to the complexity nature of these controllers, expert knowledge or, at least, encapsulation of such knowledge in a usable form is required. This, again, leads to the need for a robot controller expert system based on a well-developed taxonomy.

1.3 Project Scope and Research Objectives

This thesis focuses on the development of a taxonomy for industrial robot control. The objective is the development of an expert system that gathers information about various parameters provided by user and makes certain decisions about how to accomplish the particular task. Openness of the system is provided so the potential user can adjust the rules of particular modules and add its own control algorithms.

The key components of the system are the rulebases of different expert system's modules that acts on database (control toolbox). The developed database comprises a number of different control algorithms for both free and constrained robot motion, and is designed for using with the open real-time system.

The specific research objectives of this thesis are outlined below:

- To review and understand in depth, the difference between various robot industrial tasks, robot motion characteristics as well as control algorithms commonly used for industrial robot control.

- Based on this review, to develop a taxonomy for robot control that classifies different robot industrial tasks, manipulator characteristic motions, control architectures and controllers. This taxonomy is essential for defining the criteria for selection of robot control architectures and controllers within an expert system database.

- To develop an expert system for intelligent control by establishing a set of criteria and associated rules by which the control architecture and controller are selected for particular task.

- To investigate the correctness and appropriateness of implemented rules through the simulation results and experiments made on industrial robot.

1.4 Thesis Outline

The structure of this thesis is summarized in the outline below:

Chapter 1: Introduction: This introductory chapter.

Chapter 2: Literature Review: A review of other industrial applications of expert systems for robot control and open architecture system.

Chapter 3: Issues in Robot Control: The issues around the robot controller algorithm design, namely task identification modeling, data validity, control architecture and stability are discussed. The control architectures and controllers are classified based on their major characteristics.

Chapter 4: TREX: Taxonomy-based Robot-control EXpert System: design implementation of an expert system based on fuzzy reasoning for selection of appropriate control algorithm for a particular robot task. The major task based characteristics important for robot control scheme selection are identified.

Chapter 5: Evaluation: simulation results for contact and non-contact task using different control schemes based on decisions made by expert system.

Chapter 6: Evaluation: experimental results for non-contact tasks using different control schemes based on decisions made by expert system.

Chapter 7: Conclusions and Recommendation: A summary of the results detailed in this work and suggestion for future improvements.

Chapter II

Literature Review

Advances in robotics over the past several years have been largely incremental, with few technical breakthroughs. Vendors have focused on more subtle changes, such as improving robots' overall accuracy and repeatability, designing better teach pendants, and refining computer controllers and software. However, the current trend toward open-architecture, PC-based controllers, newer robot vision systems, incorporating different levels of intelligence into system, and the integration of the latest in CAD/CAM and robotics simulation packages, represent more substantial changes. This chapter considers literature review in development of open architecture controllers and use of expert systems for intelligent control of robotic systems with application in industrial processes.

2.1 Open Architecture Controllers

Since the introduction of the Open Architecture Control concept by Greenfeld et al.[56] in 1989, several projects have been initiated. Most open-system researchers adopt the concept of a modular approach: a master controller oversees multiple independent controller boards on a common bus. Specification for an Open System Architecture Standard (SOSAS) is described by Anderson et al. [57], in 1993 that aims to facilitate the development of interoperable, portable, scaleable and interchangeable components for a broad family of industrials controllers.

Interoperability means that the system components are able to function together in a cooperative manner. *Portability* describes the ability to operate the same system components on different controllers or hardware platforms. *Scaleability* is the ability to increase or decrease the functionality of the system, without incurring the cost of repurchasing the entire system, through upgrading specific system components. *Interchangeability* involves the ability to substitute one component with another.

Yellowley and Pottier [58], have focussed on integrating process monitoring and optimization with interpolation and axis control functions. They propose that conventional hardware and a relatively simple architecture can achieve the necessary levels of integration. Lo and Koren [59], and Pasek et al. [60], have focused on the lower level loops of Open Architecture Controllers, especially various types of servo-control algorithms for machine tool.

Teltz, Urbasik, Shawky and Elbestawi [61] have developed a comprehensive multiple pass control strategy in the context of an open architecture control environment. The strategy is implemented experimentally for a sensor-based planning and control system for rough tuning. Lundholm [62] has designed a new adaptive control system based on open architecture system design. Tung and Tomizuka [63], have built a PC based controller using the ISA bus, for advanced tracking control and adaptive force control during milling.

Recently, researchers in the Manufacturing Automation Laboratory have developed a non-proprietary, PC-based, open architecture controller that can be applied to multiaxis machinery such as CNC machine tools and robotic arms [16]. The introduction of such controllers allows the technical user to select and implement any one of a multitude of control architectures and algorithms. Given the variety of architectures/algorithms available, the selection of these becomes an issue of importance to industrial users, as well as researchers.

2.2 Intelligent Control of Robotic Systems

Prior to the introduction of open architecture controllers, the implementation of systems to select robot control strategies or algorithms was limited to research based system. For example, expert systems have been utilized in robot control applications to utilize the system designer's knowledge in the mechanism's real-time control.

Chande and Newcomb [8] proposed the integration of numerical spacecraft operations software, such as trajectory planners, with an expert system for higher-level control. Foulloy [9] has prototyped a real-time expert production system for laser cutting. Anderson [10] has developed a robot expert controller, which provides features necessary for operation in a very dynamic environment, namely robot ping-pong.

de Silva [24] investigated some approaches, prospects, and benefits of intelligent control in robotic applications. Several types of "approximation" that are used in representing and processing knowledge are outlined in this paper. Intelligence is applied on the workcell of a fish-processing system. The hierarchical nature of the workcell control system is depicted in Figure 2.1.

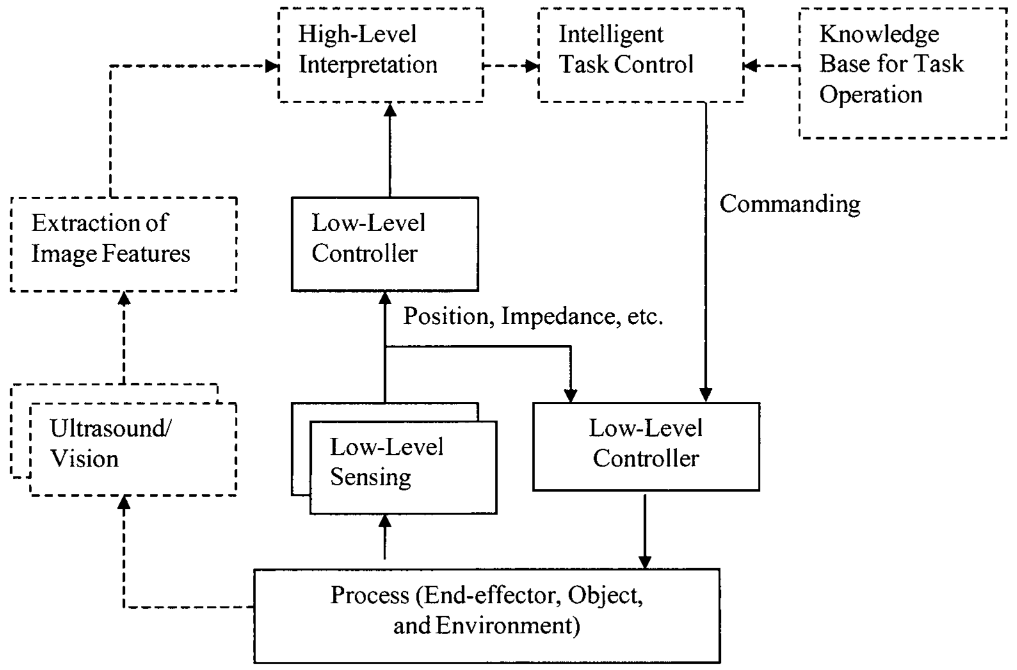

High-Level Interpretation
Intelligent Task Control
Knowledge Base for Task Operation
Extraction of Image Features
Low-Level Controller
Commanding
Position, Impedance, etc.
Ultrasound/ Vision
Low-Level Sensing
Low-Level Controller
Process (End-effector, Object, and Environment)

Figure 2.1. The hierarchical control structure of a robotic workcell.

Minbashian and Warwick [64], proposed parallel digital control scheme where the feedback path consists of four components, namely a filter bank, a control bank, an assessment/weighting block and a signal conditioning element. Selectivity in the controller mixing procedure means that the overall controller obtained is dependent on the advantages and disadvantages of different control methods, at a certain time, on a particular process. This can involve enhancement of controller tuning or adaptation capabilities, an increase in flexibility, or, in particular, robustness of real-time controllers.

The decision block is the central processing section of the intelligent parallel controller structure, and the extent and level of artificial intelligence employed is dependent on the amount of signal processing carried out, the number of controllers and the type of the plant. The intricacy of signal processing also depends on the number of mathematical functions needed to circumvent undesirable events.

The multiple tasks of checking a large number of errors and taking the appropriate actions for each individual response, requires a large rule base and this is linked directly into a recursively modifying data base for the analysis of signal trends.

The Intelligent Parallel Controller approach has some advantages and the most obvious one is processing time which is only dependent on the maximum value of the individual parallel limbs. There also exists the capability to switch a particular controller off, i.e. give it a zero weighting. Further more, the method allows for confidence building in a new controller before this is fully merged into the control process, i.e. the new controller is given a very low weighting at first and this is increased as confidence increases.

Moving from one controller to another can be carried out very smoothly providing bumpless transfer, by a gradual phased changeover.

2.3 Summary

In all of the above cases, the expert systems have been developed for very specific process/robot tasks, and low-level controllers. However, more general-purpose expert systems for open architecture control would be of benefit for designing and using industrial robot systems. In order to design such a system, a **taxonomy for robot control** is a useful and necessary first step.

In the following chapters, some of the important issues in the design and selection of control architectures and robot controllers are reviewed. The surveyed work is a subset of a much larger collection of important work on robot control issues. Based on this review, we propose an initiatory taxonomy of industrially appropriate robot controller architectures and algorithms that could be achieved using an open architecture controller system. Using this taxonomy, an expert system design for selection and implementation of task-appropriate robot controllers is designed and presented.

Chapter III

Issues in Robot Control

In this chapter the issues around robot controller algorithm design, namely task identification modeling, data validity, control architecture and stability, are discussed, and as a result of this discussion, robot controllers and control architectures are classified based on their internal characteristics (linear/non-linear), robustness, sensitivity, reliability, task appropriateness, as well as environmental interaction characteristics. This chapter provides the basis on which a task oriented taxonomy of robot controllers is presented in Chapter IV.

3.1 Robot Control

The robot control problem involves the computation of the actuator inputs necessary to track a desired trajectory within certain specification. The dynamics of a manipulator are typically described by a set of highly nonlinear and coupled differential equations. Consequently the design specification and tuning of controllers is a difficult task even for simple robot operations like point-to-point motion. Complex operations like continuous path motion with applied force can be very difficult to control; for example, in processes like robotic assembly.

Robot control can be widened to include the integration of task goals, trajectory planning/generation, force and position feedback, and modification of trajectories. It also requires a good physical (dynamic and geometric) understanding of the task so that effective control strategies can be implemented.

The selection of the control architecture to be used for a particular task in order to give the best possible performance depends on variety of issues such as the type of task, the speed of the sampling interval, the type of robot, knowledge of the robot dynamics, the characteristics of the environment, availability and validity of sensor information, knowledge of unmodeled (external) disturbances, hardware availability etc. In the following sub-sections each of these issues will be considered.

3.2 Type of Robot Task

In general, industrial robot tasks can be divided in two groups: non-contact tasks and contact tasks. The non-contact tasks are further subdivided into point-to-point (PTP) motion and continuous path (CP) motion. In both cases a large number of approaches have been developed for controlling such motion (e.g., PTP [11]; CP [12]). Specialized controllers have been developed to achieve specific objectives and optimizations such as obstacle avoidance, time optimality and minimum energy consumption.

On the other hand, many issues, some still unresolved, arise for tasks when the robot is in contact with environment (friction forces, environment dynamics, stability problems, etc.). These issues must be addressed in order to achieve satisfactory results. Over the last decade there have been attempts to make the application of some control procedures suitable for the industrial, contact-type tasks. Among these, the most notable are: hybrid position/force control [6] and impedance control [7].

3.3 Models

In order to design a control system, the model of the robot and the environment, either explicit or implicit, is required. Conversely, the characteristics and uncertainties associated with the available models drive the selection of the controller architecture/algorithm. The use of non-adaptive model-based algorithms, where feedback loops are intended to compensate for nonlinear changes in cross-coupling dynamic forces, is still common practice.

One of the methods available for obtaining the values of the dynamic forces involves the use of an internal model of the robot dynamics. This leads to difficulties when the dynamic model is formulated as a complex system of nonlinear differential equations.

Thus, an important question in control design is: to what extent it is necessary to model the robot dynamics. A more complex model allows better performance characteristics, but at the same time it requires faster, and more expensive control hardware. The second problem follows from the fact that the model used in robot control is always more or less an approximation of the behavior of the real robot. There are always unmodeled effects whose incorporation would significantly increase the time needed for a precise calculation of the inverse dynamics [17].

In the following subsections the models typically used for both, the robot and the environment are considered, and the control issues related to these models are identified.

3.3.1 Robot and Environment Model

Robot manipulators are highly nonlinear systems. Conventional robot manipulators operate under simple PID regulators with feed-forward control signals that are calculated on the basis of a nominal robot model dynamics. Thus, the analysis and design of model-based motion control procedures require the development of efficient, closed form dynamic equations.

An accurate analysis of the characteristic of the mechanical structure, sensors, and actuators contribute to well-behaved end-effector control. Some approaches have been developed for handling the mathematical complexities involved in the dynamic model of robot manipulator with varying levels of success [19]. A brief consideration of the models used to describe robot motion follows.

The dynamic model of the robot interacting with the environment with second-order actuator dynamics is described by the vector differential equation in the form

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \tau_q + \mathbf{J}^T(\mathbf{q})\mathbf{F}, \tag{3.1}$$

where $\mathbf{q} = \mathbf{q}(t)$ is an n – dimensional vector of robot generalized coordinates; $\mathbf{H}(\mathbf{q})$ is an n x n positive definite matrix of inertia moments of the manipulator mechanism; $\mathbf{h}(\mathbf{q}, \mathbf{q})$ is an n – dimensional non-linear function of centrifugal, Coriolis and gravitational moments; $\tau_q = \tau_q\ (t)$ is an n – dimensional vector of input control (joint torques); $\mathbf{J}(\mathbf{q})$ is an n x m Jacobian matrix relating joint space velocity to task space velocity; and $\mathbf{F} = \mathbf{F}(t)$ is an m – dimensional vector of generalized forces and moments acting on the end-effector.

In the case when the environment does not exhibit displacements (DOF) that are independent of robot motion, the mathematical model of the environment dynamics in the frame of robot coordinates can be described by non-linear differential equation [34]

$$\mathbf{M(q)\ddot{q} + L(q, \dot{q}) = - S^T(q)F}, \tag{3.2}$$

where $\mathbf{M(q)}$ is a non-singular n x n matrix of environmental inertia, $\mathbf{L(q, \dot{q})}$ is a non-linear n – dimensional vector function that takes into account the equivalent elasticity and damping of the dynamic environment, $\mathbf{S^T(q)}$ is an n x n matrix with rank equal to n, i.e. rank($\mathbf{S}$) = n. Then the system (3.1), (3.2) describes the dynamics of robot interacting with dynamic environment.

For controller design purposes, it is customary to utilize a linearized model of the manipulator and environment. The applicability of linearized model in constrained motion control design, especially in industrial robotics systems, was demonstrated in [35, 36].

Neglecting the non-linear Coriolis and centrifugal effects due to relatively low operating velocities during contact and assuming the gravitational effect to be ideally compensated, the linearized model around a nominal trajectory in Cartesian space x_0 is obtained from (3.1) and (3.2) in the form

$$\mathbf{\Lambda(x_0)\ddot{x} + B(x_0)\dot{x} = \tau(x_0) + F},$$
$$\mathbf{\Lambda(x_0) = J^{-T}(q_0)H(q_0)J^{-1}(q_0),\ B(x_0) = J^{-T}(q_0)bJ^{-1}(q_0),\ \tau(x_0) = J^{-T}(q_0)\tau_q}, \tag{3.3}$$

where $\mathbf{b}$ is viscous friction coefficient for the robot joints.

In passive environments, it is convenient to adopt the relationship between forces and motion around the contact point in linear form

$$-\mathbf{F} = \mathbf{M}_e\Delta\ddot{\mathbf{x}} + \mathbf{B}_e\Delta\dot{\mathbf{x}} + \mathbf{K}_e\Delta\mathbf{x}, \tag{3.4}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_e$, $\mathbf{x}_e$ is contact point location, $\mathbf{M}_e$, $\mathbf{B}_e$, $\mathbf{K}_e$, are inertial, damping and stiffness matrices, respectively.

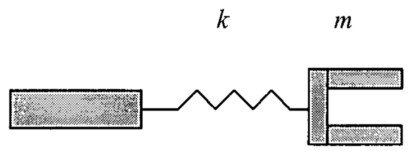
3.3.1.1 Classification of Environment

For contact task operations, the description and characteristics of the environment (that is, the contact surface) are essential for good control. The environment can be classified in different groupings: e.g. inertial (pushing), resistive (sliding, polishing, drilling), compliant (spring-like, compliant wall). For each group different control architectures are applicable, Figure 3.1.

For example, in a stiff environment, a **damping control scheme** suffers sluggish behavior but a **stiffness control scheme** with imposed low gain of controller achieves very good results. If the environment is highly compliant, a **position control architecture** works well but in a stiff or resistive environment this approach is not recommended [1].

One of the main problems in **hybrid control**, which combines both position and force control, is that the system model breaks down in non-orthogonal directions (i.e., directions that are either purely force constrained or purely position constrained) [37].

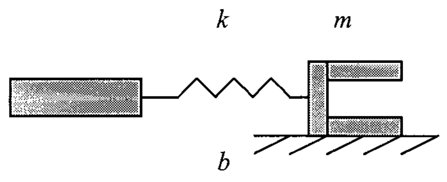
The orthogonality between the constraint force and the direction of unconstrained motion has been assumed and used in the majority of works.



environment impedance

$Z_e = mks/(ms^2+k)$

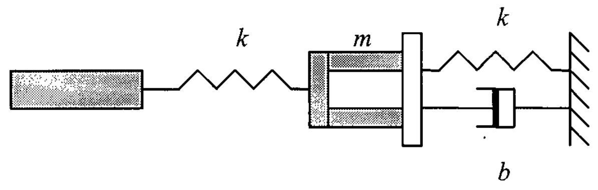
a) inertial environment requires a position controlled manipulator



environment impedance

$Z_e = b + mks/(ms^2+k)$

b) resistive environment allows either position or force/torque controlled manipulator



environment impedance

$Z_e = ms + b + k/s$

c) capacitive environment requires force/torque controlled manipulator

Figure 3.1. Classification of dynamic environments.

Block diagrams of environment interaction for position and force control can be represented in general form as shown in Figure 3.2 (a) and (b).

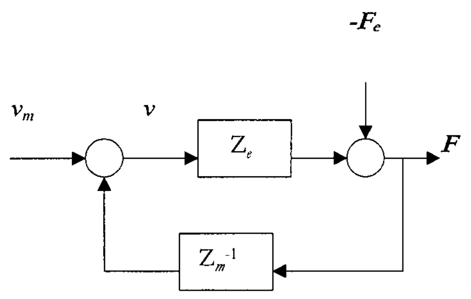


Figure 3.2 (a). Position-control block diagram.

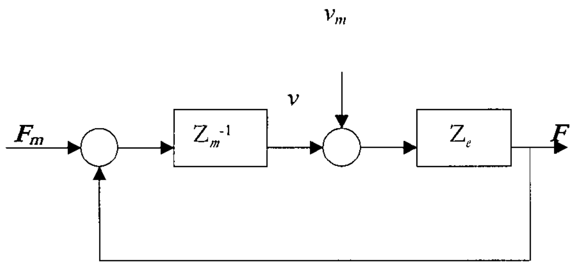


Figure 3.2 (b). Force-control block diagram.

A relatively new approach for robot control interacting with a dynamic environment is to represent the environment as a mechanical system with two deformations and two dynamic modes. A benefit of this model is in its more accurate representation of real environmental dynamics, however it introduces additional complexity into system (Figure 3.3). The system dynamics is described by

$$F = m_1(\ddot{q}_1+\ddot{q}_2)+b_1\dot{q}_1+k_1q_1,\quad b_1\dot{q}_1+k_1q_1 = m_2\ddot{q}_2+b_2\dot{q}_2+k_2q_2. \quad (3.5)$$

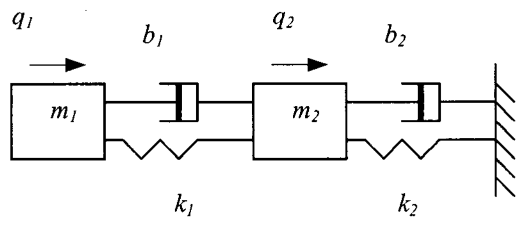


Figure 3.3. Dynamic environment with two deformations and two dynamic modes.

3.3.1.2 Model Uncertainties

Uncertainties associated with either the robot model or the environment model can degrade task performance [13]. One of the main problems in the synthesis of control laws in a contact tasks is the representation of uncertainties in the dynamic model of the environment. These uncertainties are generally due to difficulties in identification and prediction of model parameters and the behaviors of the environment. Therefore it is important that the synthesized control law is robust to these model uncertainties. Reference [14] presents a method for an accurate and effective analysis of the influences of these uncertainties on system performances. Knowledge of unmodeled (external) disturbances is an important issue in robot operations where precision is crucial issue. For example, the **hybrid impedance control architecture** is very good for force tracking under external disturbances [1], whereas an **impedance control scheme** [2], would not be suitable for such a task.

3.4 Controller Data Issues

Typical robot control architectures are characterized by a tight coupling between sensors and actuators, minimal computation, and task-achieving "behavior" problem decomposition. Within these architectures, the controller requires a reference signal that is provided at regular intervals, and must receive all necessary data within some certain bounded time intervals to function properly.

3.4.1 Uncertainties and Disturbances

Model uncertainties and internal/external disturbances can significantly degrade the system performance if not properly modeled or compensated. Friction is a common source of disturbances in robots; controllers with integral actions are able to compensate for these disturbances only in a limited number of cases. For example, for the contact task case, if the first order integral action controller is applied, then the constant disturbances (constant force) will be compensated. Disturbances of the ramp type, or some more complex type, can be compensated by second or higher order integrators, respectively. This remark holds for Cartesian robots. However, for non-Cartesian robots, which are the most common, friction force compensation is difficult [13].

3.4.2 Sampling/Controller Rate

The selection of the best sampling rate for a robot control system is a compromise of many factors. Choosing an appropriate sampling rate for robot contact tasks is particularly difficult due to the relationship to the type of the environment, and external disturbances etc. Khosla [4] has compared real-time performances of **computed-torque (CT)** and **independent joint control (IJC)** schemes as a function of the sampling rate. He showed that increasing the sampling period from 2 to 5 ms resulted in the degradation of the performance of the IJC scheme but improved the performance of CT scheme. If the maximum possible gains are selected for certain sampling rates then the performance of both schemes improves with higher sampling intervals. One can also note that, a high sampling rate is important for high stiffness environments due to the pronounced effect of external disturbances on the system. For a more compliant environment lower sampling rates or a larger approach speed can be allowed.

3.4.3 Sensing

The availability and validity of sensor information is one of the crucial issues in choosing appropriate control scheme. Dynamic forces can be obtained utilizing the force sensors, such as joint force transducers. Since such transducers measure total dynamic forces, joint force sensory feedback is robust to parameter variations and model uncertainties [18]. However, the application of joint torque sensors has numerous technical problems. Implementation of force transducers usually requires special design of the robot joints. Further, joint torque sensors reduce the structural stiffness of the arm, so the torque sensor behaves as a compliant member in the system, and therefore, may be a cause of instability for the robot controller [17].

As pointed out in [3], when robot joint velocity and position are available as measurements, different control laws that are fixed or adaptive in nature can be applied for point-to-point and continuous path motion control. For example, applying a **parallel control architecture** always requires fully available sensor information [2]. Such an architecture is suitable when dealing with planning task error, but strongly relies on detailed geometric modeling of the environment, and the type of contact. **Hybrid position/force control architecture** requires position and force sensor measurements.

On the other hand, a **hybrid impedance control architecture** can still achieve relatively good performance if the sensor information is degraded or skipped. In some robotic applications, information about joint velocity may be degraded, for example with actuators using tachometers or differentiation methods for measuring speed. In such cases other control schemes such as **stiffness control** or **resolved acceleration-motion force control** that rely on such information as input become unsuitable.

3.5 Stability Issues

One of the most difficult problems in position/force control of robots interacting with a dynamic environment is the stability of both desired motion and interaction forces [13]. Various control approaches, such as hybrid control, parallel control, explicit/implicit control, stiffness control, impedance control, damping control, etc., indicates that stability of a contact task is a problem that has not yet been satisfactorily solved, either from the theoretical or practical view. In fact, when considering a specific contact task, in almost all approaches simplifications are introduced in the modeling of robot and the environment.

3.5.1 Kinematic and Dynamic Instability

In [15] Hollerbach has shown that a new form of instability (*kinematic instability*) in the force control of the robot manipulator is caused by an inverse kinematic transformation in the feedback path and this form of instability occurs only in multi-joint manipulators. It has been shown that kinematic instability occurs not only at the point of kinematic singularities, where the Jacobian inverses are not defined, but also at the wide range of the manipulator workspace, where the Jacobian inverses are well-defined [13].

In [12] McClamroch and Wang emphasized the important role of the constraints in the dynamic model of manipulator, especially with relation to the stabilization problem. They presented the global condition for tracking based on a modified computed torque controller and local conditions for feedback stabilization using a linear controller. One of the problems in hybrid position/force control refer to some dynamic stability issues in force control, such as high-gain effect of force sensor feedback (caused by high environment stiffness), unmodeled high frequency dynamic effects (due to arm and sensor elasticity), contact with a stiff environment, non-collocated sensing and control, and is identified as the *dynamic stability problem* [19].

3.5.2 Contact Task Stability

A treatment of the contact task stability problem, considering the environment as a nonlinear dynamic system, is given in [5]. It is shown that if impedance control is applied, enabling the robot to be asymptotically stable in free space, then the robot interacting with the environment is a passive system and is stable in isolation. However, the conclusion is valid only if the robot in contact is at rest and for this reason the result cannot be considered complete [13].

3.5.3 Practical Stability against Asymptotic Stability

In all practical cases, the robustness of a synthesized control law must be tested against the environment model uncertainties in order to achieve valid results. For such an analysis, conditions for the practical stability of the robot motion around desired path and interaction forces with the environment, must be established. Taking the model and parameter uncertainties into account, as well as different external disturbances, it may be difficult to achieve asymptotic (exponential) stability of the system. Therefore, it is of interest to consider the so-called practical stability of the system, which is less rigorous, rather then exponential stability. The practical stability of the robot around the desired position and force trajectories is defined by specifying the finite regions around the desired position and force trajectories within which the robot actual position coordinates and velocities and forces have to be during task execution [13].

3.6 Overview of Controllers and Control Architectures for Robot Control

Traditionally, a control architecture for manipulation robots is designed as a decentralized system, consisting of a set of local constant-gain **PID** regulators.

Decentralized control schemes, in their simplest form, consist of local servo systems closed around individual joints of the robot. For an uncoupled system, a decentralized architecture is less complex and gives a faster response. However, robot stabilization around a nominal trajectory is poor with such local controllers, which do not take dynamical coupling between joints into account. If precise positioning is not required and if the trajectory is low-speed in nature then the use of a decentralized controller is sufficient to achieve desired dynamic behavior.

Implementation of **centralized control schemes** is not common in industrial robot systems and their use is only considered if robot is required to overcome a specific problem (e.g. large disturbances or tracking of high-speed trajectories). These architectures perform better both for slow and fast trajectory tracking, but they require complex hardware and specific systems software due to computation of the whole system's dynamics and the more computationally complex control laws.

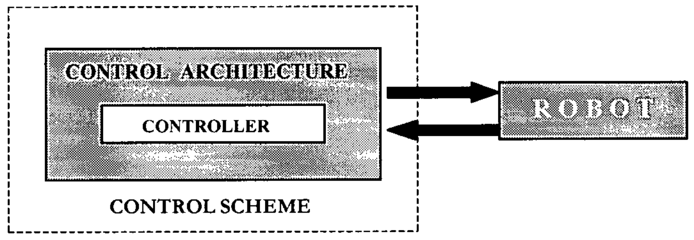


Figure 3.4. General control scheme.

A centralized nominal control architecture will compensate for the dynamic coupling between joints and stabilize the robot for high-speed trajectories. For direct-drive robots and for fast trajectories it is generally necessary to use such a centralized control structure that takes into account the whole system dynamics. A major distinction between the **control architecture** and the **low-level controller** should be emphasized. Low-level controllers are incorporated within the design of control architectures. The architecture routes the input/output signals to and from the controllers and as a result determines which specific variables are controlled (e.g. stiffness, damping, impedance, etc.) in order to take advantage of specific model-based characteristics of the plant. Low-level controller on the other hand considers only the signals as provided, and has a dominant influence on task performance. Together they make a **control scheme/algorithm** (Figure 3.4).

3.6.1 Controllers

Low-level robot controllers can be divided in generally into two groupings: **instantaneous controllers** (P, PD, PD+G etc.), which do not have the ability to store previous state information, and can operate only on the current position and velocity errors, and **dynamic controllers**, which are characteristically robust. Further, one can make a distinction between **linear controllers** (e.g. computed torque controllers) and **non-linear controllers** [33].

Controllers that have the ability of storing and manipulating the previous state information are **dynamic controllers.** Such controllers are classified as **one-degree of freedom controllers** that filter the measured signal through a dynamic system before feeding the signal back to the input, and **two-degree of freedom controllers** that allow simultaneous specification of the desired response to a command input, and guarantee the robustness of the closed-loop system [23]. Introducing the integral feedback (one-degree of freedom) can compensate for the influence of constant gravitational moment. However, this type of controller cannot eliminate the error produced by the time variable disturbance, such as external moment, thus use of dynamic controllers of higher order are necessary. **Non-linear controllers** are very sensitive to parameter uncertainty. In particular, load changes introducing a large-state error. Nonlinear **passive-based controllers (PBC)** are simple to implement but do not provide quantifiable performance measures. High gains are not desirable due to their negative effect on asymptotic stability. With a small damping factor, the response for non-linear controllers is very slow and has a large overshoot. However, these controllers achieve good disturbance attenuation, and hence may be viable candidates for applications where rise time is not of prime concern. Their bandwidth stays fixed (redundant) for varying reference trajectories (e.g. sinusoidal).

Sliding mode controllers (SMC) are also quite robust to source disturbances but are highly sensitive to parameter uncertainties, and usually have very fast response times with no overshoots [33]. **Variable structure controllers (VSC)** encounter 'chattering' as a main problem and perform well when using a robot without excessive torque effort. **Saturation-type controllers (STC)** are most useful when a short transient error can be tolerated whereas **feedback-linearizing controllers (FLC)** have faster response for faster poles, very high peaks and large overshoot. For precise applications FLCs are not recommended.

However, two important advantages of FLCs over other controllers are: they achieve almost same convergence rate with smaller current (less energy consumption) and the state error is systematically smaller. The smallest phase shift can be achieved with these controllers. This also provides the best achievable bandwidth. However, there are still quite sensitive to step disturbances [33].

3.6.1.1 Summary of Controllers

Based on above discussion, robot controllers can be classified as shown in Figure 3.5 and 3.6 according to the following characteristics: overshoot, complexity, robustness, sensitivity, response, disturbance rejection, and reliability. The first diagram is a three-dimensional representation of three controller's characteristics dependency *overshoot/response/disturbance rejection.*

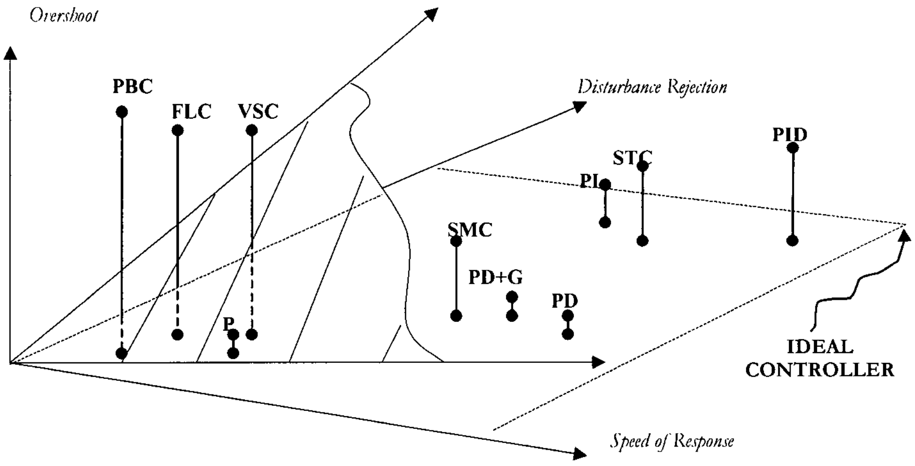


Figure 3.5. Response/overshoot/disturbance rejection diagram.

The second diagram is a three-dimensional representation of three other controller's characteristics dependency *complexity/reliability/robustness.*

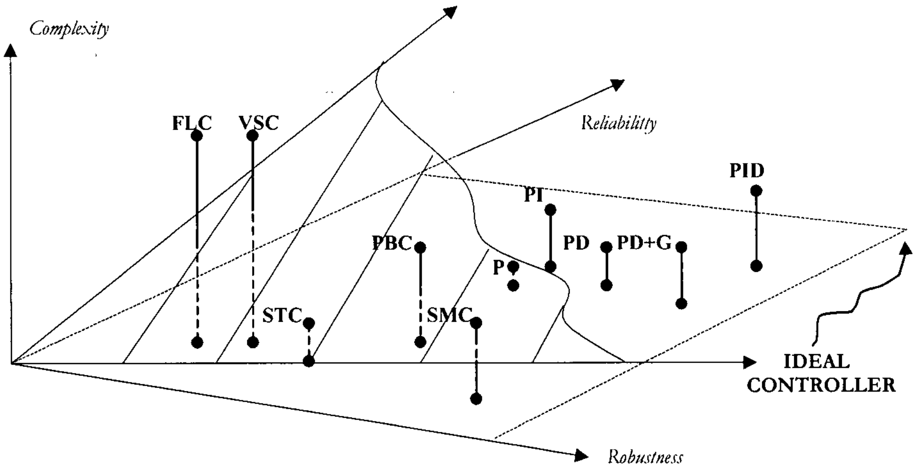


Figure 3.6. Robustness/reliability/complexity diagram.

Reliability – controller's ability to accomplish not only tasks from one certain class of tasks and specific processes , but also the various kind of either less or more complex tasks

Complexity – designing complexity of certain controller, can be expressed through the number of mathematical operation needed to calculate the control law

Robustness – controller's ability to cope successfully with parameter uncertainties, variations of system's parameters, unmodeled system dynamics, external /internal/ disturbances, sensitivity to force perturbation, changes in end – effector position, etc.

From Figures 3.5 and 3.6 one can conclude that generally controller's can be divided into three groupings. Firstly, 'highly' recommended controllers (P, PD, PD+G, PI and PID), secondly, 'occasionally' recommended controllers (SMC and STC), and thirdly, 'rarely' recommended (for special purposes) controllers (VSC and FLC).

As shown in the figures, except for the specific cases, where the ability to model complex plant dynamics is crucial, non-linear types of controllers are generally not a good choice. A tuned PID controller is a standard use in most of industrial applications. Of course, the most appropriate choice is dependent on many of the issues previously discussed, namely trajectory/task rate, uncertainties and disturbances, sensing and stability issues. Taking these controller characteristics and criteria based comparisons together leads towards the taxonomy and rule creation for controller selection in TREX.

However, the controller selection is mainly driven by the choice of the controller architecture, which is discussed next.

3.6.2 Control Architectures

Control architectures are most commonly discussed in the context of control for robot interacting with the environment. For active force control, one can distinguish between: (*i*) **operational space control** techniques where the robot control is taking place in the same frame in which robot actions are specified [38, 39]. This approach requires the construction of a model describing the system dynamic behavior as perceived at the point of an end-effector where the task is specified (operational point, i.e., coordinate frame). The traditional approach for specifying compliant motion uses **task** or **compliance frame** approach [6]. This geometrical approach introduces a Cartesian compliant frame with orthogonal force and position (velocity) controlled direction. In order to overcome the limitations of this approach, new methods were recently proposed [41]. These, approaches, referred to as explicit compliant motion task specification, are based on the model of the constraint topology for every contact configuration and utilize projective geometry metrics in order to define a hybrid contact task.

(*i*) **Joint space control**, whereby control objectives and actions are mapped into joint space [42]. Associated with this control approach are transformations of action attributes, compliance and contact forces from task into joint space.

The increasing demand for advanced robot application has brought about an enormous growth of interest in the development of different concepts and schemes for the robot control. Numerous control procedures are available today for completely different class of tasks from non-contact to contact robot task. Over the past several years, compliant motion control has emerged as one of the most attractive and fruitful research areas in robotics [13].

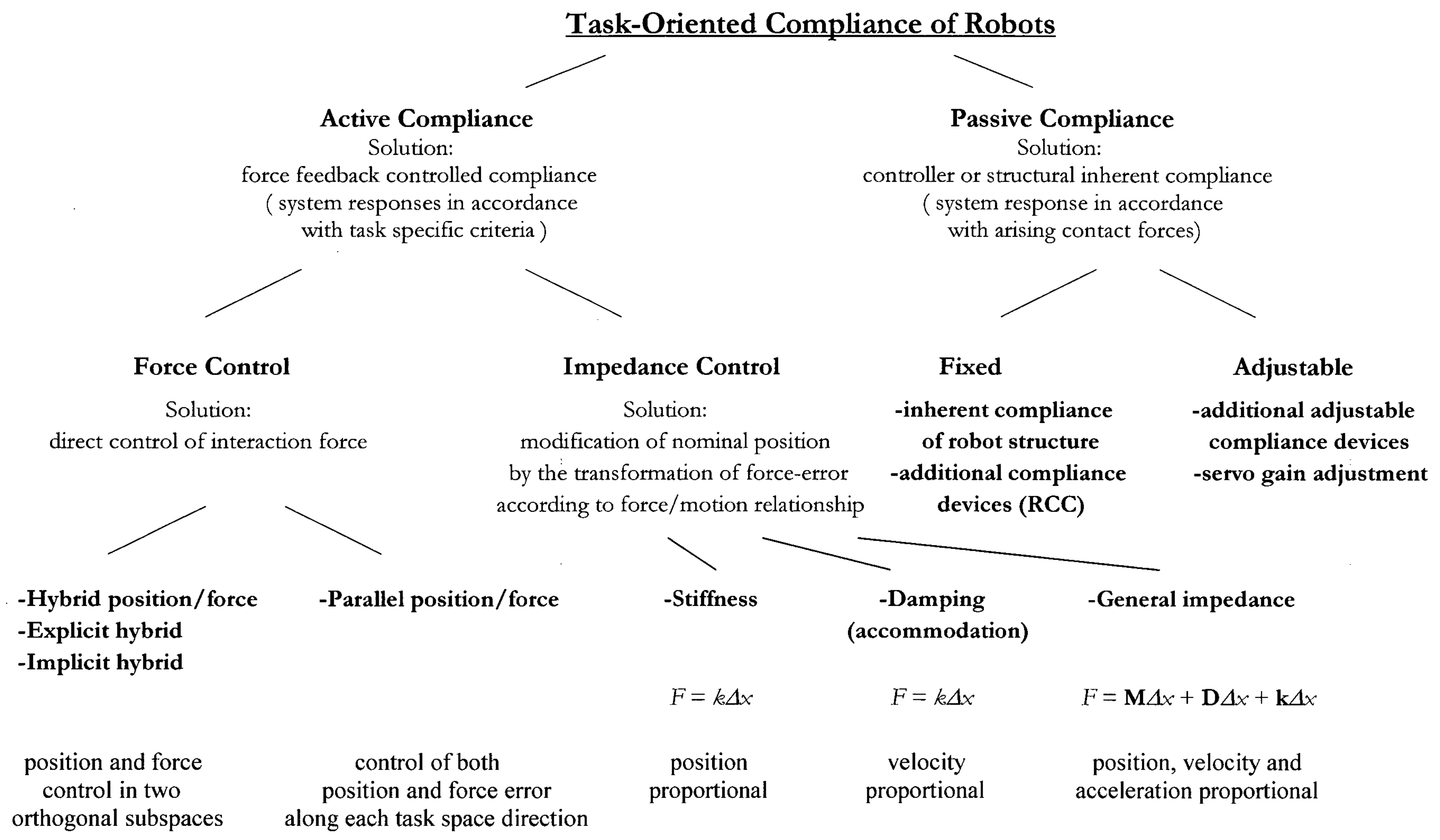


Figure 3.7. Classification of control concepts.

The robot control methods for contact task can be systematized according to different criteria. In terms of robot/environmental compliance, two basic groups of control concepts are distinguishable: **passive compliance**, where position of the end-effector is a result of the contact forces, the environmental compliance, and the passive compliance located in the manipulator structure, and, **active compliance**, where the end-effector compliance is adjusted using force feedback to either control the interaction force or to achieve a task-specific compliance at the robot end-point.

Further classification of control concepts for constrained motion control is shown on Figure 3.7.

Passive compliance is a concept often used in practice to overcome the problems arising from positional and angular misalignments between the manipulator and its working environment.

3.6.2.1 Impedance Control

Active compliant motion control methods utilize the dynamic behavior of the manipulator to achieve a desired level of interaction with the environment. The first implementation of force feedback control of a manipulator was **impedance control**. The objective of this control concept is to achieve a specific mechanical behavior at the manipulator end-effector, referred to as *target impedance*. This objective imposes a desired relationship between position error and force acting at the end-effector. A basic impedance control architecture with internal position control is presented on Figure 3.8.

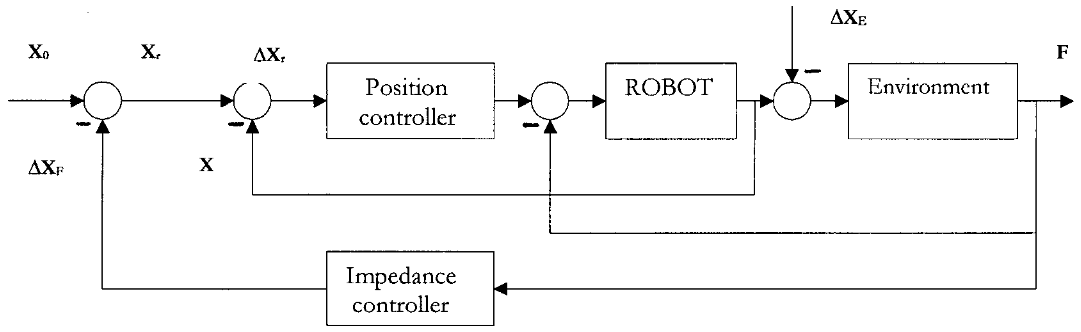


Figure 3.8. Impedance control scheme.

The scheme consists of two control loops: an inner position servo as and an outer loop generating position command modification (Δx_F) based on the end-effector force-torque measurements. The outer loop is closed when the end-effector encounters environment. The role of impedance controller is to realize the target impedance model in Cartesian space (compliance frame). This architecture does not require any significant modification of the position servo controller and it is very suitable for implementation in conventional robotic systems. This control scheme was recently implemented in the new space control system SPARCO [43] that integrates completely impedance control including servo control, sensor integration, motion planning, language support and monitoring functions.

The most common and general impedance control concept was established by Hogan [7]. The focus of this approach is on the characterization and the control of dynamic interaction based on manipulator behavior modification. In this sense the impedance control is an augmentation of position control. The control strategy requires the robot to behave as the inverse to the environment. That is, if the environment is best modeled as an admittance (i.e. it is very compliant), then impedance control should be applied and, conversely, if the environment is best modeled as an impedance (i.e. it is very stiff) then admittance control should be applied.

Almost all proposed impedance control architectures utilize a linear target impedance with, at most, a second order model. The reason for this is that the dynamics of a second-order system is well understood.

3.6.2.1.1 Damping Control

Various control schemes were established and proposed for controlling the relation between robot motion and interaction force. One of the first approaches to impedance control is proposed by Whitney [44]. In this approach, referred as **damping** or **accommodation control** (Figure 3.9), the force feedback is closed around the velocity control loop. The interaction force is converted into velocity command modification by a constant damping coefficient $\mathbf{K}_F$. Conditions for stability imply that if the stiffness of environment is high, to avoid large contact forces, very high sampling rates are required.

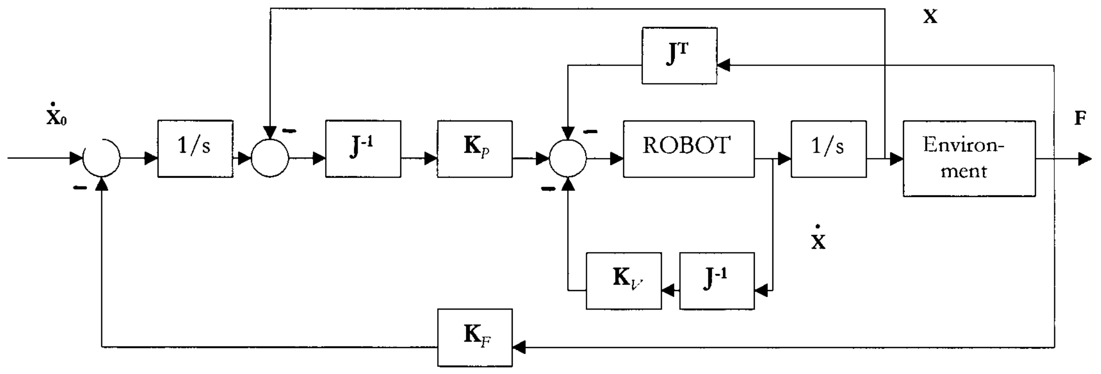


Figure 3.9. Damping control scheme.

3.6.2.1.2 Stiffness Control

Salisbury [45] has proposed a new scheme, namely, **stiffness control**, where the end-effector position is modified in accordance with the interaction force (Figure 3.10). This architecture is based on a generalized stiffness formulation $\mathbf{F} = \mathbf{K}\delta\mathbf{x}$ where $\delta\mathbf{x}$ is a generalized displacement from a nominal commanded end-effector position, and $\mathbf{K}$ is a six-dimensional stiffness matrix.

Based on a difference between desired and the actual end position, a nominal force is computed and converted into joint torques using the transpose of the Jacobian matrix.

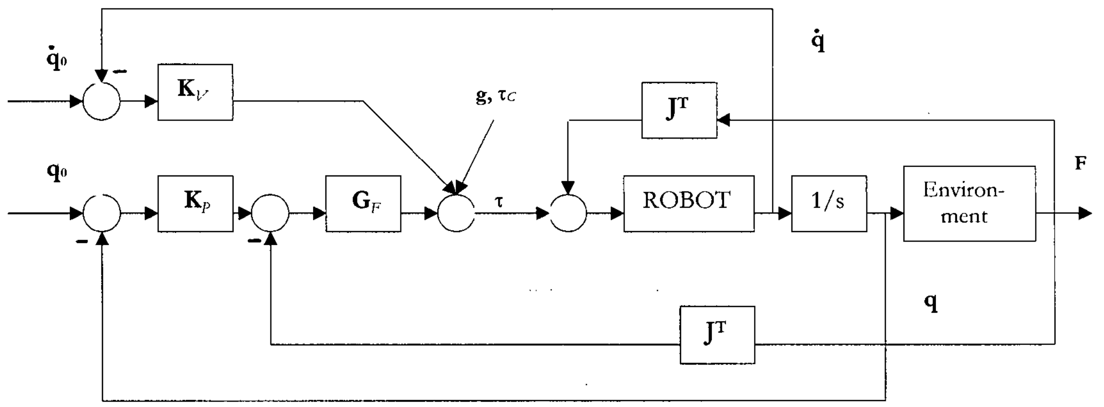


Figure 3.10. Stiffness control scheme.

This force is then used to determine the torque error in each joint, which is further used to correct the applied torque so that the desired force (i.e. stiffness) is maintained at the robot hand.

3.6.2.2 Force Control

Maples and Backer [46] modified the Salisbury's architecture by closing the force control loop around a position controller, namely **internal loop force control**. Unlike the original architecture, all computation including desired force $\mathbf{F}_0 = \mathbf{K}\Delta\mathbf{x}$, force error formation using sensed forces, and force compensation, are made in a task frame. The output of the force compensator is a position modification, which is realized through a standard position controller. A pure integrator is used as a force compensation filter. This architecture achieves a relatively good robustness against perturbation (e.g. friction).

3.6.2.2.1 Hybrid Position/Force Control

Hybrid position/force control approach is based on a theory of compliant force and position control formalized by Mason [6] and concerns a large class of tasks involving partially constrained motion of the robot. In dependence of the specific mechanical and geometrical characteristics of the given contact problem, this approach differs in two sets of constraints upon the robot's motion and the contact forces. A set of constraints which is a natural consequence of the task configuration, i.e., of the nature of the desired contact between an end-effector and constraining surface, is called *natural constraints*. A suitable frame in which the natural constraints are prescribed is referred to as the *task* or *constraint frame* [47]. In order to specify the task, which should be realized by the robot in the compliant frame, so-called *artificial constraints* have to be introduced. Practically, these constraints partition the possible DOFs of the motion into those which must be position controlled and those that should be force controlled in order to perform the given task. It is reasonable to define an artificial constraint with respect to force when there is a natural constraint on the gripper motion in this direction (DOF), and vice versa. For the sake of simplicity, a diagonal Boolean matrix S, called the compliance selection matrix [40] has been introduced. In accordance with the prescribed artificial constraints the *i*th diagonal element of the matrix has the value 1 if the *i*th DOF with respect to the task frame is to be controlled, and the value 0 if it is position controlled.

3.6.2.2.2 Explicit Hybrid Position/Force Control

The most important method within this group is the architecture proposed by Raibert and Craig [40], namely **explicit hybrid position/force control** (Figure 3.11). The control consists of two parallel feedback loops, the upper one for the position and lower one for force feedback loop.

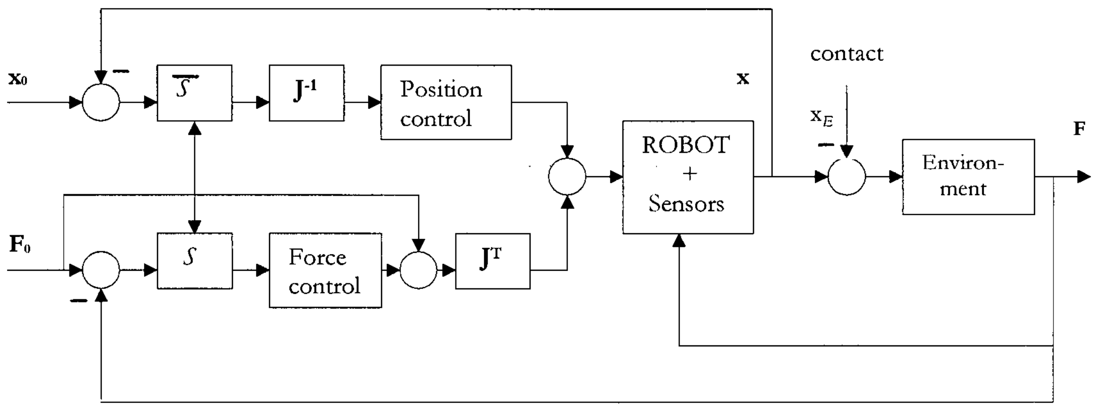


Figure 3.11. Explicit position/force control architecture.

Each of these loops use separate sensor systems. Separate control laws are adopted for each loop. The central idea of this hybrid control method is to apply two independent control loops assigned to each DOF in the task frame. Both control loops cooperate synchronously to control each of the manipulator joints.

Initially this concept appears to be ideally suited to solve the hybrid position/force control problems. However, there are some fundamental difficulties related to this concept. The first problem is related to the opposite requirements of the hybrid control concept concerning position and force subtasks. Position control must be very stiff in order to keep positioning errors in the selected direction as small as possible, while the force control requires a relatively low robot stiffness (corresponding to the desired force) in the force controlled direction in order to ensure that the end-effector behaves compliantly with the environment. As already explained above, the explicit hybrid control tries to solve the problem by control decoupling into two independent parts which are position and force controlled. This would require that the position control does not interfere with the force control loop. However, this is not the case. Since each robot joint contributes to the control of both the position and force, coupling in the manipulator mechanical structure results in control input to the actuator corresponding to the force loop producing additional unwanted forces in position-controlled directions in the task frame and vice versa. Due to this coupling between position and force controlled DOF, setting position errors in the force controlled directions to zero reduce the position feedback gains in all directions, causing the entire system to become highly sensitive to perturbations. As a consequence, the performance of a robot is not unique with this scheme for all configurations and for all force-commanded directions. Moreover, one can find certain configurations for which, depending on selected force and position directions, the systems becomes unstable. To overcome the dynamic problems of hybrid position/force control several investigators have pursued the idea of including the robot dynamic model in the control law. For example, resolved acceleration control, originally formulated for the position control [48], belongs to the group of dynamic position control algorithms. Shin and Lee [49] have extended this approach to the hybrid position/force control. The joint space implementation of the proposed control scheme is sketched in Figure 3.12.

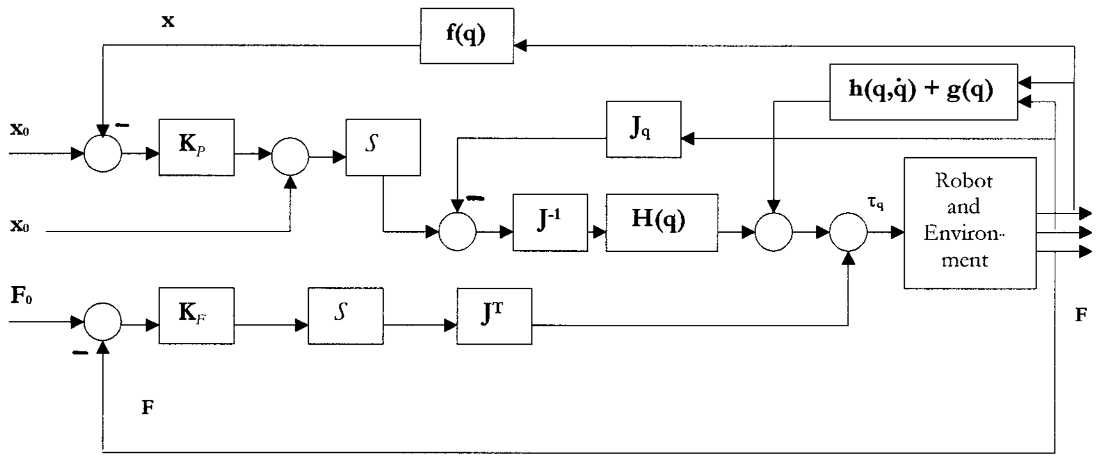


Figure 3.12. Resolved acceleration-motion force control.

In this scheme, the driving torques compensate for the gravitational, centrifugal, and Coriolis effects and feedback gains are adjusted according to the changes in the inertial matrix. An acceleration feedforward term is also included to compensate for changes along nominal motion in position directions. Finally, the control inputs are computed by

$$\tau = \hat{\Lambda}\bar{S}\,\ddot{x}^{*} + \hat{\mu}(x,\dot{x}) + \hat{p}(x) + Sf^{*}, \tag{3.9}$$

where $\mathbf{x}^*$ is the commanded equivalent acceleration

$$\ddot{x}^{*} = \ddot{x}_0 + K_v(\dot{x}_0 - \dot{x}) + K_p(x_0 - x), \tag{3.10}$$

and $\mathbf{f}^*$ is the command vector from the force control parts whose form depends on the applied control law. To minimize the force error, it is convenient to introduce the **PI** force controller of the form

$$f^* = K_{f_p}(F_0 = F) + K_{f_r}\int (F_0 - F)dt. \tag{3.11}$$

3.6.2.2.3 Implicit Hybrid Position/Force Control

In commercially applied robotics systems, the most promising approach is to implement a **implicit** or **position-based control** [46] architecture by closing a force-sensing loop around the position controller (Figure 3.13). The input to the force controller is the difference between the desired and actual contact forces in the task frame. The output is an equivalent in force controlled directions which is used as a reference input to the positional controller.

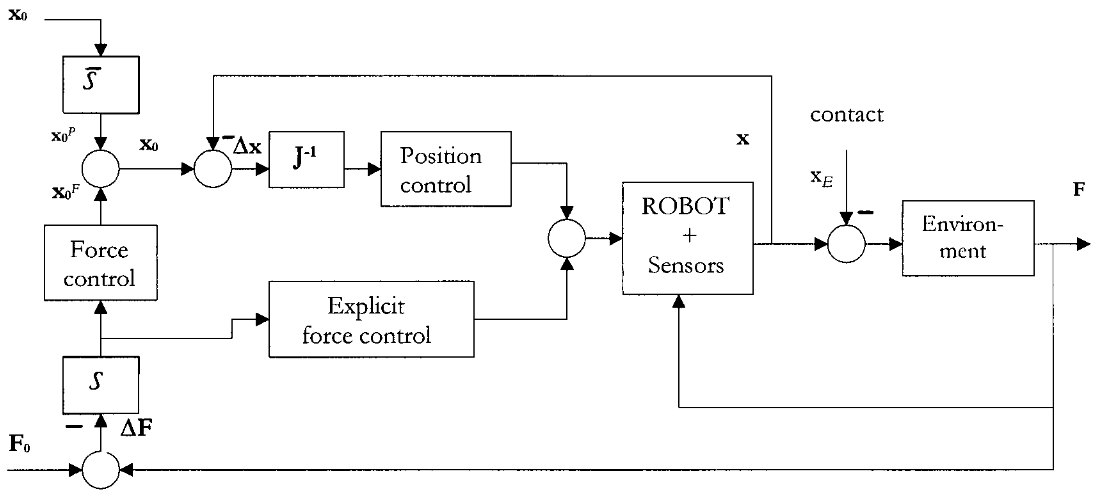


Figure 3.13. Implicit position/force control scheme.

According to the hybrid positional/force control concept, the equivalent position in the force direction $\mathbf{x}_0^F$ is superimposed on the orthogonal vector $\mathbf{x}_0^P$ in the compliance frame, which defines the nominal position in orthogonal position controlled directions. The robot behavior is practically affected only by the acting force. The positional controller remains unchanged, except the additional transformations between the Cartesian and task frames.

3.6.2.3 Parallel Control

A conceptually new approach to position/force control, namely **parallel control** (Figure 3.14), has been proposed in [50]. Contrary to the hybrid control, the key feature of the parallel approach is to have both position and force control along the same task space direction without any selection mechanism. Since in general both position and force cannot be effectively controlled in a uncertain environment, the logical conflict between the position and force actions is removed by imposing the dominance of the force control action over the position one along the constrained task direction where a force interaction is expected.

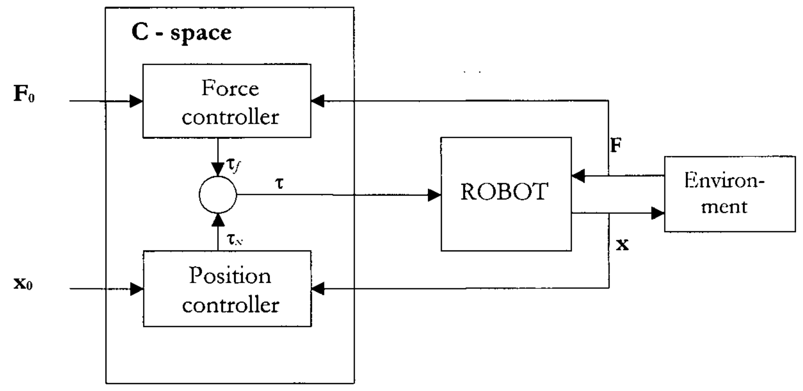


Figure 3.14. Parallel position/force control architecture.

The force control is designed to prevail over the position in constrained motion directions. This means that the force tracking is dominant in directions where an interaction with environment is expected, while the position control loop allows the compliance, i.e., a deviation from the nominal position in order to reach the desired process. For a parallel control architecture, consisting of a **PD** controller on the position loop, and **PI** action in the force loop, together with the gravity compensation and desired force feedforward loop, a set of sufficient local asymptotic stability conditions has been derived in [51], including the stability analysis and simulation results on a industrial robot.

Different from parallel control approach proposed in [50], Vukobratovic and Ekalo [52, 53] have established a unified approach to control simultaneously position and force in a environment with completely dynamic reactions. This highly complex dynamic approach dealing with the control of robots interacting with dynamic environment is unlikely to be feasibly implemented for industrial robots.

3.6.2.4 Summary of Control Architectures

The summary of control architectures is given through the diagram shown on Figure 3.15, expressed over three major characteristics: robustness, complexity, reliability. Comparison of control architectures through these characteristics helps in deriving appropriate rules for architecture selection in TREX.

The use of the characteristics and criteria for selection of a particular controller via an expert system is consider in Chapter IV. Each characteristic is defined as follows.

Reliability – architecture's ability to accomplish not only tasks from one certain class of task, but also the various kind of either contact or non-contact.

Complexity – defines the time needed for a certain architecture to calculate the control law and provide the control input to the robot manipulator

Robustness – architecture's ability to cope successfully with parameter uncertainties, variations of system's parameters, unmodeled system dynamics, external /internal/ disturbances, sensitivity to force perturbation, changes in end – effector position, etc.

The comparison of control architectures is presented in a three-dimensional coordinate system *robustness/complexity/reliability*.

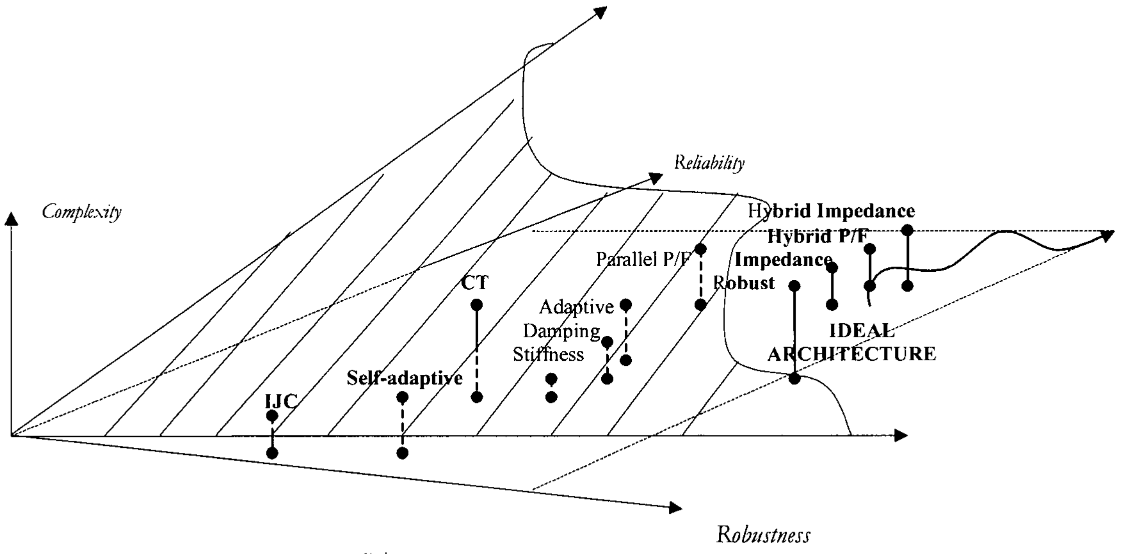


Figure 3.15. Complexity/reliability/robustness diagram for constrained motion.

According to the comparison diagram shown above, one can conclude that 'ideal' control architecture would be the one that is highly robust and reliable and less complex. As shown, IJC, the most commonly used architecture in industrial systems, is the least ideal scheme in terms of robustness and reliability.

Due to today's very powerful hardware systems, the higher weight in these relationships is set to first two characteristics then to a complexity characteristics. Thus, the control architectures that are commonly used in robotics constrained motion tasks, and what is called 'ideal architecture' are: impedance control and hybrid position force control.

This review presents a number of issues which are essential to selection of a robot controller - mainly task type and robot/environment model. Considerations of these issues, along with the criteria shown here, are the basis for the development of TREX.

3.7 Remarks

In this chapter, the survey of the state-of-the art in advance robotics area based on available publications is given. Although several high-level control strategies and schemes have recently been proposed and elaborated, the number of advanced robotics applications for complex contact tasks remains insignificant. The reason for this is that the majority of new concepts are still in laboratory investigation stage, and their implementation in today's low performance control systems is too tedious and inefficient.

This brief review of robot control issues points to three somewhat separate, but related themes, namely: (*i*) robot tasks, (*ii*) robot/environment models and the associated controller architectures developed for such models, and, (*iii*) the low-level controller design that is mainly related to stability issues. Sub-issues such as model uncertainties, disturbances, sampling rates, and sensing provide a clearer view of the problems related to robot control.

A closer evaluation of the low-level controllers point to the following major characteristics that classify such controllers: trajectory rate (low/high speed), driving system of the robot joints, the rate of response, control accuracy, degree of stability, overshoot, and rise time.

However, as discussed, the controller selection must be strongly linked to the robot control architecture. Control architectures, are most generally offered for interaction control. They can be classified by following characteristics: type of motion, model (robot/environment) type, model (robot/environment) confidence, sensor information, computational rate, input/output rate, sampling rate, environmental/manipulator joint stiffness, planning task error, and internal/external disturbances.

Thus, this review give rise for the classification of control procedures, namely, a **robot control taxonomy**, which is presented in the following chapter.

Chapter IV

"...biti isti, biti poseban, biti slobodan,
biti samo svoj..."
-Partibrejekersi

TREX: Taxonomy-based Robot-control EXpert-system

4.1 Taxonomy

In order to develop an expert system for robot control, the issues mentioned in previous chapter, related to robot and environment characteristics, control schemes, and controller, should be addressed. The first step in the development of such a system is a classification of different control approaches related to various industrial robot tasks. By developing such a classification, the system can be applicable in industrial purposes, and used to provide better understanding of robot control problems and solutions.

The proposed taxonomy of robot control architectures is shown in Figure 4.1. In this figure, the main issues for selection of robot control architectures have been divided into four sections: (1) **Industrial Application**, (2) **Motion Characteristics**, (3) **Control Architecture**, and (4) **Low-level/specialized controller**.

The first two categories are tightly coupled, as the selection of a **motion characteristic** is application driven.

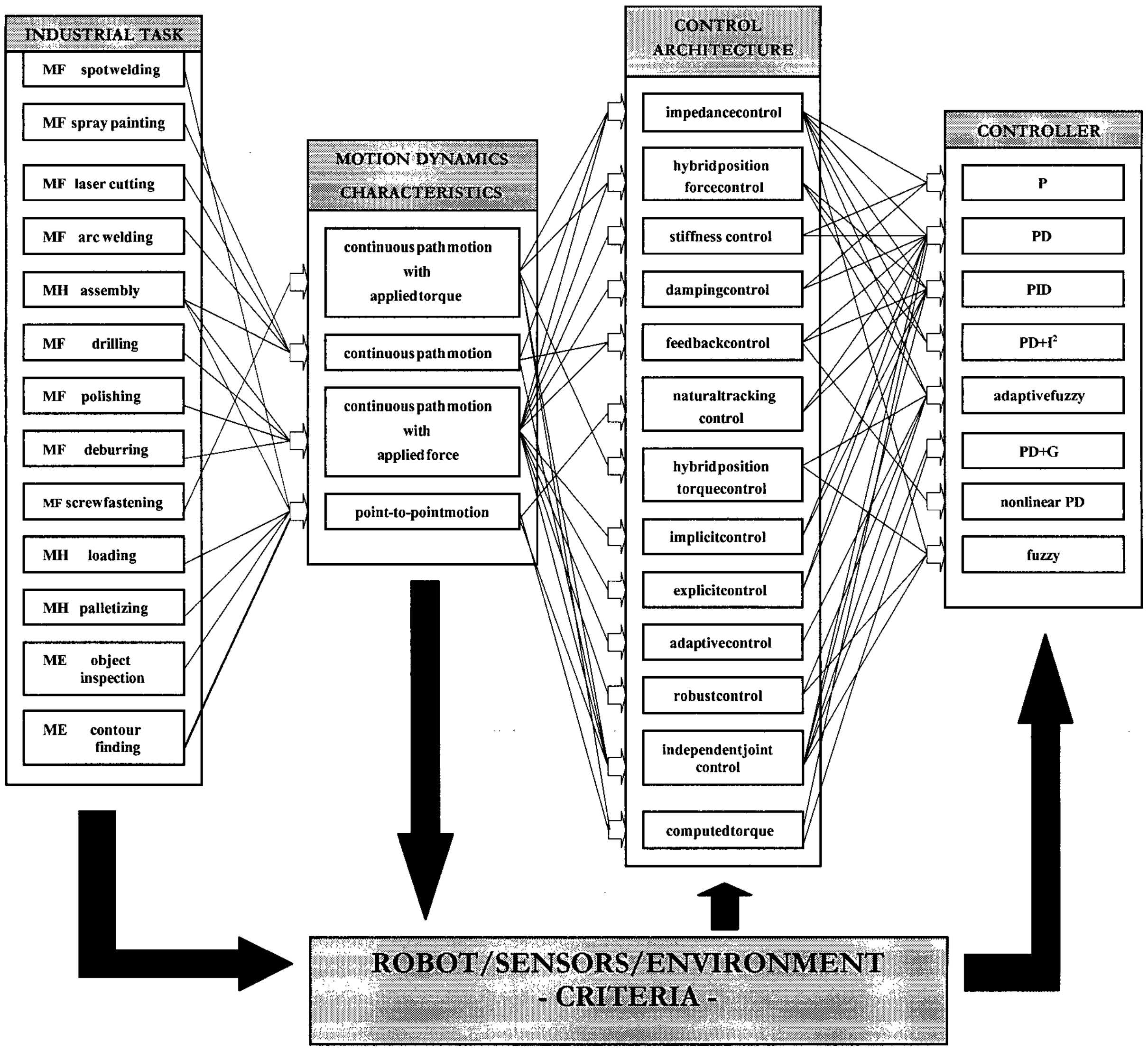


Figure 4.1. Taxonomy scheme.

The remaining categories distinguish between issues relating to the design of **control architectures** and the applied **control algorithms**. As seen in the previous chapter, control architectures resolve issues mainly related to the modeling of the robot and the environment. Control algorithms are mainly designed based on issues of stability, sampling rates, and uncertainties. However, as represented in this taxonomy, these issues must also be considered in selecting control architectures. As a result some particular robot control algorithms and architectures tend to be better suited to each other. These relationships are represented by the crosswise links shown in Figure 4.1.

4.1.1 Industrial Tasks and Motion Characteristics

In these categories, robot industrial tasks are divided into three major groups: **manufacturing (MF)**, **material handling (MH)** and **measuring (ME)**. Each consists of various different industrial tasks each related to one of four possible robot motion dynamics characteristics: **point-to-point motion**, **continuous path motion**, **continuous path motion with applied force**, and **continuous path motion with applied torque**.

Some robot operations like the assembly process may consist of multiple types of robot motions. A description of the type of robot motion is a necessary first step for selection of appropriate control procedure. Some industrial tasks like spray painting, laser cutting or arc welding require a simple unconstrained **continuous path motion** that may be efficiently accomplished, for example, by a computed torque control architecture with a PD controller.

However, it is well known that an independent joint architecture with a PD controller with exact gravity compensation is asymptotically stable at the zero equilibrium state for **point-to-point motion**, although imposing high gains is necessary.

On the other hand, for processes like drilling, deburring or polishing, the problem of modeling the environmental contact arises requiring a control architecture for force compensation and, simultaneously, position control. Thus, for this motion dynamic characteristic, namely, a **continuous path motion with applied force**, different control architectures can be applied: impedance, hybrid impedance, parallel position/force, hybrid position/force each altogether with different types and orders of controllers depending upon user's requirements.

4.1.2 Control Architectures

In this particular system, the main representatives of structurally different control schemes that have been successfully used over the last decade to accomplishing various industrial tasks have been chosen. Each has advantages and disadvantages, and may be selected via a matching of user requirements and task specifications with the characteristics of each control scheme. Based on the review in Chapter III, the following characteristics for the selection of a particular control architecture are identified:

- robot model
- environment model
- system capabilities
- controller objectives and
- system inputs

For example, in applications where the robot is in contact with environment, impedance control and hybrid control architectures are the most suitable depending upon the environmental model and controller objectives. In the case where force sensor information is unavailable, implicit force control can be utilized. However, it is very difficult to achieve robust hybrid position/force control that would, with a unique control law and unique controller gains, be suitable for a wide range of contact problems (i.e. for different natural and artificial constraints). For point-to-point motions, natural tracking control gives good tracking capabilities without the knowledge of the nonlinear system's internal dynamics. In summary, the characteristics listed above can assist in the selection of one control schemes over another.

For continuous path motion with applied force or torque, especially when robot operates in uncertain environment, or when some unmodeled dynamics are introduced into system, robust control approach can cope with these problems. Two attractive features of the robust control architecture are that on-line computation is kept to a minimum and its inherent robustness to additive bounded disturbances. However, disadvantages of this control approach are in its requirement for a priori knowledge of bounds on the uncertainty, and that even in the absence of additive bounded disturbances, it is impossible to guarantee the asymptotic stability of the system.

4.1.3 Controllers

There is no overall mathematical approach that can prove that using one controller over the other will give better performances although some indications exist. For example, disturbances in robot contact tasks due to friction forces can be compensated with **PI** or **PID** only in a limited number of cases. However, both **PD** controllers with partial compensation and with fixed non-linear compensation rely on the complete and exact knowledge of manipulator dynamics, and such knowledge cannot be acquired easily. In the case where a robot picks up some payload or changes operation tool during the operation, it is impossible to accurately calculate varying robot dynamics. In such cases an adaptive control scheme, which does not require exact manipulator dynamics seems to be a promising solution. Model reference adaptive control (MRAC) relies on the assumption of slow motion of the manipulator as well as slowly varying dynamics. However, asymptotic stability cannot be proven. Zero state error and fast parameter accommodation can be achieved with adaptive control via linear parameterization.

Using the fuzzy logic or adaptive fuzzy controller might improve output response a great deal. Herein, the proposed criteria for choosing the appropriate controller are the indicators for quality of the transient response:

- tracking error
- speed of response
- damping ratio
- overshoot
- settling time and
- dominant time constant

4.1.4 Discussion

The proposed taxonomy provides a methodology for selection of a control architecture and related controller suitable for a particular task, robot model, environmental model, sensor availability, and stability criterion. The control architecture is clearly separated from the controller. The control architecture provides a mechanism for getting the persistent excitation of inputs for the particular control law, whereas the controller is incorporated into the internal architecture's loop. Each control architecture obtains and calculates these inputs in different ways depending upon the task objective, available measurements /sensor information/, and desired inputs.

The criteria that also play a role in selection of the control architecture are identified. Interconnections between architectures and particular controllers on Figure 4.1 are proposed, based on the review given in Chapter III. The selection of the architecture along with the identified criteria, namely, overshoot, rate of response and disturbance rejection, drive the controller selection.

This taxonomy leads to the design of an expert system controller that is "open" in form. The potential user could adapt, and expand such as existing system by adding or revising control architectures and controller modules. Using an open architecture controller hardware a flexible controller system can be developed as described in the following section.

4.2 Taxonomy-based Robot-control EXpert-system

This chapter discusses the design of implemented expert system and its sub-systems for selection of appropriate control algorithm for a particular robot task. A brief introduction of fuzzy logic approach as one of the possible methods for intelligent control precedes the discussion of its application to the hierarchical expert system architecture and its modules. In this work fuzzy logic is used as a tool for implementation of taxonomy over an expert system design.

4.2.1 Fuzzy Logic

Fuzzy logic is an extension of conventional set theory. It is a theory for operations on a class of sets known as fuzzy sets. As opposed to conventional sets that have a degree of membership of 0 or 1, fuzzy sets have an additional real number valued parameter that expresses the degree to which an object belongs to a set. Thus, fuzzy logic allows the extension of set theory to problems where there are not crisp determinations of an object membership in a specific class, and further, there are not crisp definitions of the class itself.

This makes fuzzy logic well suited to reasoning about problems often considered by human expert.

4.2.2 Fuzzy Reasoning

Fuzzy inference is the process of formulating the mapping from a given input to an output utilizing fuzzy logic. There are two types of fuzzy inference systems that can be implemented in MATLAB Fuzzy Logic Toolbox [65]: Sugeno-type and Mamadani-type, where latter is most commonly used. Mamdani's method uses min- and max-operators to make inferences with a set of fuzzy rules expressed in an IF-THEN format [22].

$$\textbf{IF } x \textbf{ IS } M_1 \textbf{ AND } y \textbf{ IS } N_1 \textbf{ THEN } z \textbf{ IS } L_1\textbf{,} \text{ Weight } 1.0 \tag{4.1}$$

$$\textbf{IF } x \textbf{ IS } M_2 \textbf{ OR } y \textbf{ IS } N_2 \textbf{ THEN } z \textbf{ IS } L_2\textbf{,} \text{ Weight } 0.2 \tag{4.2}$$

The fuzzy rules (4.1) and (4.2) map the inputs x and y from M and N sets respectively, into output z from set L. The variables x, y, and z referred to as universe of discourse for given sets. Figure 4.2 illustrates Mamdani's direct fuzzy reasoning method over membership functions $\mu_M(x)$, $\mu_N(x)$, and $\mu_L(x)$, which describe the fuzzy sets M, N, and L.

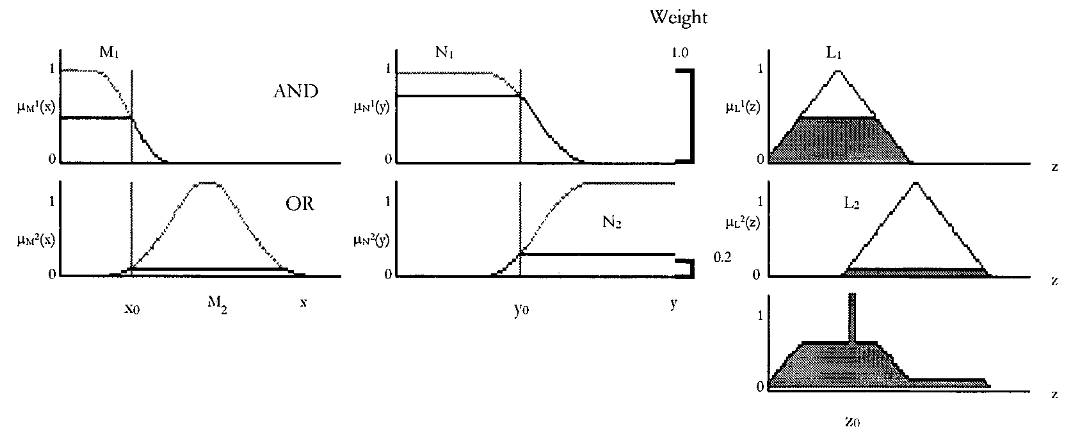


Figure 4.2. A diagram illustrating Mamdani's direct fuzzy reasoning method.

4.2.3 Expert System Design

Intelligent robot control can significantly improve the performance of the manipulator. Conventional AI research often employs rule-based systems. Rule-based systems have a number of advantages when applied to certain problems. The rules separate the knowledge base from the inference engine required to use it, allowing each to be developed independently. The inference engine provides data matching and search algorithms that are task independent. The expert controller must select among multiple strategies and cope with varying results. In this particular application, the developed expert system acts on database, which consists of different control architectures altogether with different controllers (see Figure 4.3)

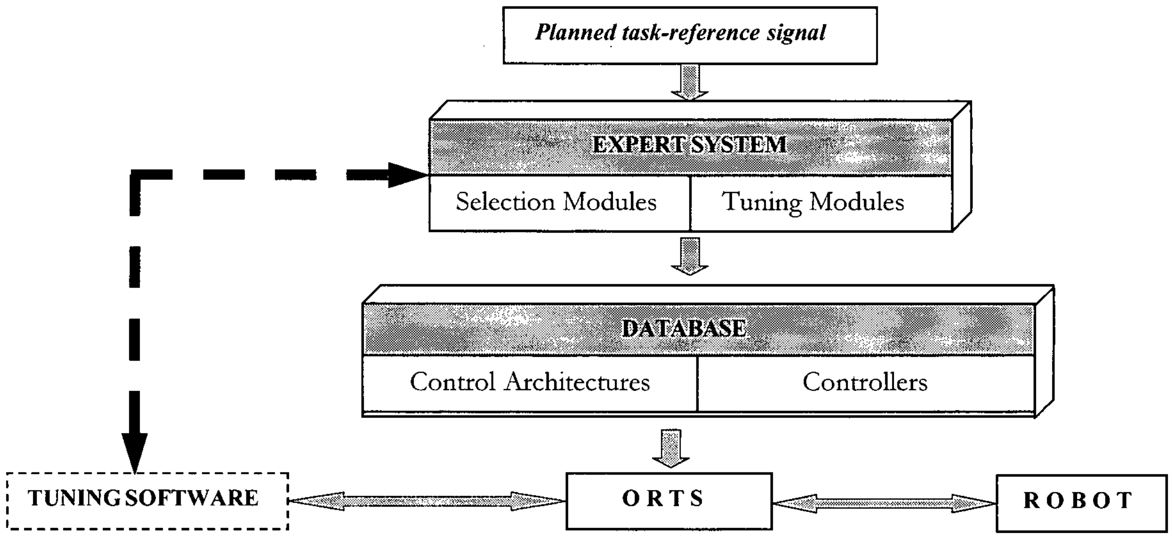


Figure 4.3. Schematic figure of system.

The proposed system is shown on Figure 4.4. The system is developed using a MATLAB Fuzzy Logic Toolbox [65]. Given a planned task motion and user information about the task goals as the inputs to expert system's rule base, the inference engine selects a most appropriate control scheme and controller for particular problem. These are then downloaded to the real-time open architecture system and applied to the robot. The interconnection between the tuning software, expert system, and controller provides online tuning of particular controller. It also allows feedback to the expert system to self-tuning of the rules.

4.2.4 Expert System Architecture

The expert system comprises three main modules namely the **control architecture module (CAM)**, the **controller module (CM)**, and the **tuning module (TM)**. Each module works separately but the architecture output from the *CAM* module is used as an input to the *CM* module. This work focuses on the CA, and C modules. The tuning module is the subject of future research.

4.2.5 Expert System Modules

The expert system's inference engine for choosing an appropriate control architecture consists of eight modules whereas controller selection is made by two expert modules. Each of these modules works separately. Different membership functions and different number of rules are developed for each module.

However, the final decision about controller depends on the final decision about control architecture. The individual modules are discussed in the following sub-sections.

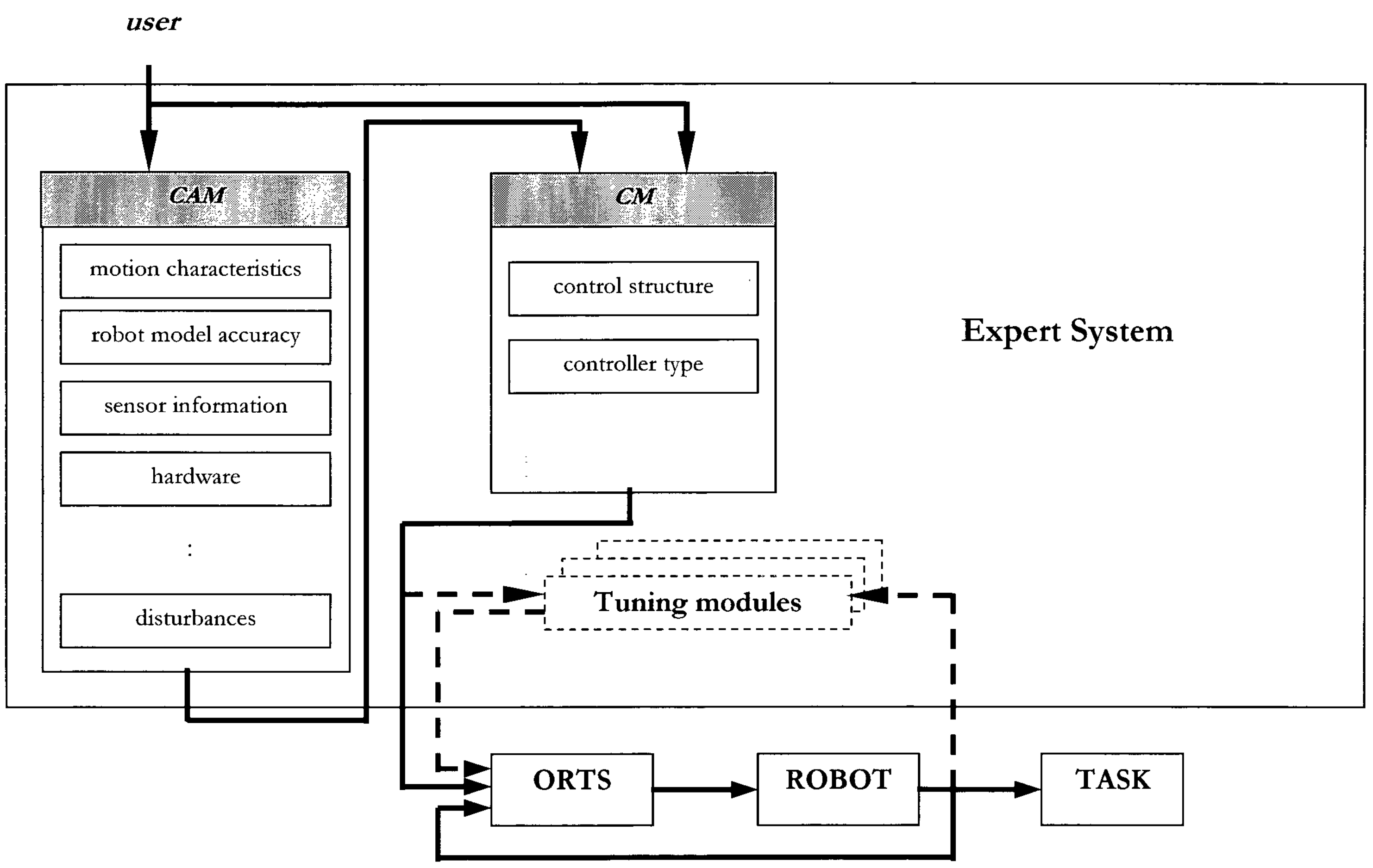


Figure 4.4. Configuration of Expert System with Robot.

Control Architecture Modules	Controller Modules
- robot and environment model type - robot and environment model confidence - disturbances - effective stiffness - allowable planing task error - dynamic motion characteristics - sensor information - hardware	- control structure - controller type

Table 4.1. CA and C modules.

4.2.5.1 Dynamic Motion Characteristics Module

The criteria for selection of appropriate **dynamic motion characteristic** are expressed through the following relationships. These relationships are shown in the user interface through graphically displayed membership functions. The expert system consists of 14 rules.

Type of motion = *f*(contact/non-contact), **T = t/l**, [sec]/[m], (4.3)

Following of desired/specific path (tracking) = *f*(following/non-following),
F = k/l, [%]/[m], (4.4)

Control type during the contact (if exists) = *f*(position-force/position-torque),
C = n/t, [%]/[sec], (4.5)

t – time that robot spent in contact with environment expressed in [sec]
l – path length
k – a number that express end-effector following desired path expressed in [%]
n – a number that expresses position-force control during the contact (if exists) expressed in [%]

Based on the estimated values for T, F, C, their fuzzified memberships, and the fuzzy rules, the inference engine calculates a defuzzified value that express one of the dynamic motion characteristics: **point-to-point, continuous path, continuous path with applied force** or **continuous path with applied torque motions**.

This module comprises 14 rules. The membership functions and related part of fuzzy rules are shown on next figure.

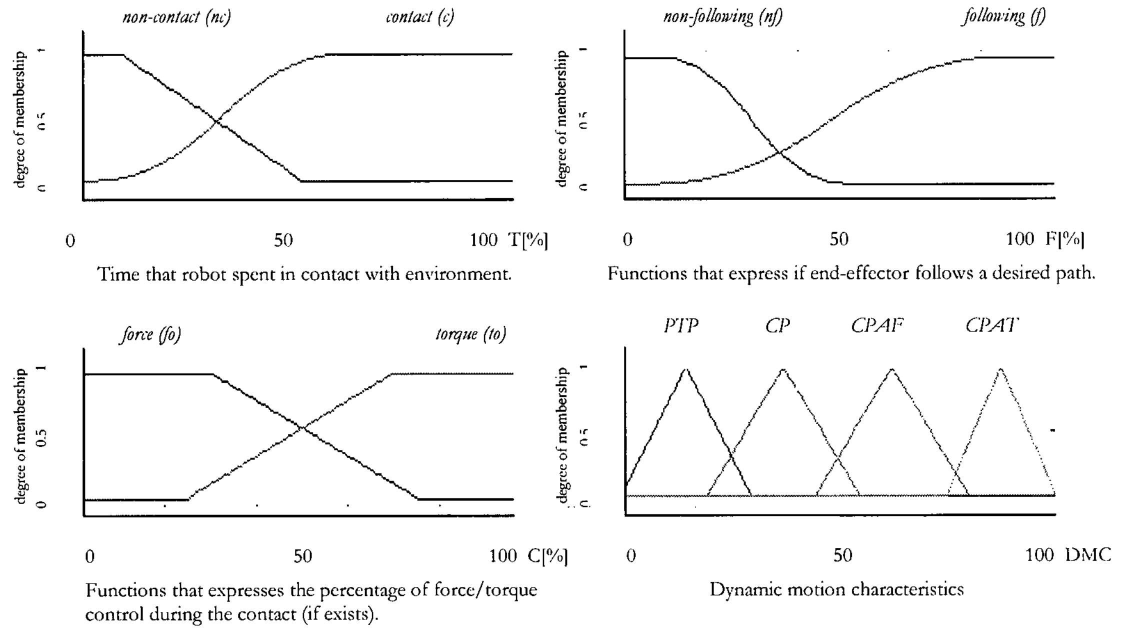


Figure 4.5. Membership functions for DMC module.

```
If T is nc and F is nf and C is fo then DMC is PTP, W=1.0
else if T is c and F is f and C is fo then DMC is CPAF, W=1.0
else if T is c and F is nf and C is to then DMC is CPAT, W=0.7
else if T is c and F is nf and C is fo then DMC is CPAF, W=0.9
                         :
end
```

Table 4.2. A part of rulebase for DMC module.

4.2.5.2 Model Confidence and Model Type Modules

The **Model Confidence** and **Model Type** modules are connected, and work together. If the output from dynamic motion characteristics module described in section 4.2.5.1 is some type of unconstrained motion (e.g. point-to-point, continuos path motion) then only information about the robot model confidence (expressed over fuzzy values *low, medium, high)* and robot model type (expressed over membership functions for *linear-coupled, linear-uncoupled, nonlinear*) are gathered. Otherwise, for robot contact motions, information about the environment model confidence (*low, medium, high)* and environment model type (*linear-kinematical, linear-dynamical, nonlinear-kinematical, nonlinear-dynamical*) have to be obtained as well. The membership functions are presented on Figure 4.6.

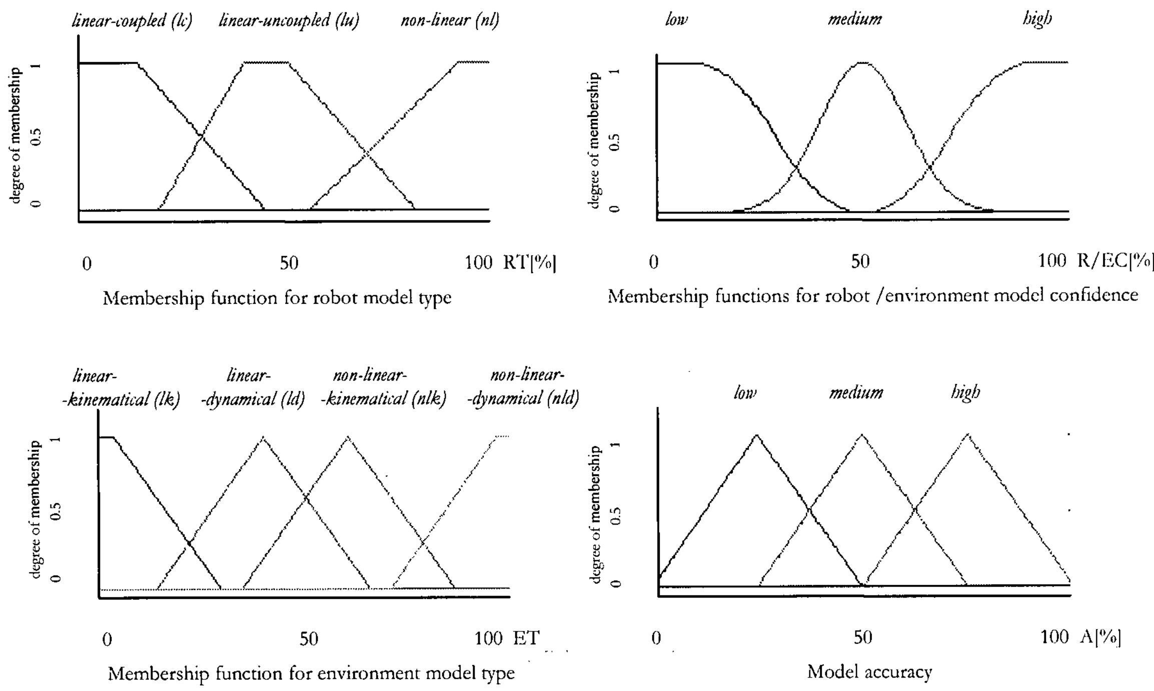


Figure 4.6. Membership functions for model confidence/type module.

These modules comprise 18 rules giving a defuzzified value that express **model accuracy**, and the rulebase is designed in a similar way as for previous module.

4.2.5.3 Sensor Information Module

The user provides information about position, velocity, force and torque sensor availability. Then the information validity is estimated as:

Validity of information = *f*(**information uncertainty/signal filtering**) [%] (4.6)

Thus, for each available sensor, the validity of information is in function of information uncertainty and information filtering. The defuzzified value provides a measure of the sensor's information validity. The 10 rules make a rulebase for each sensor.

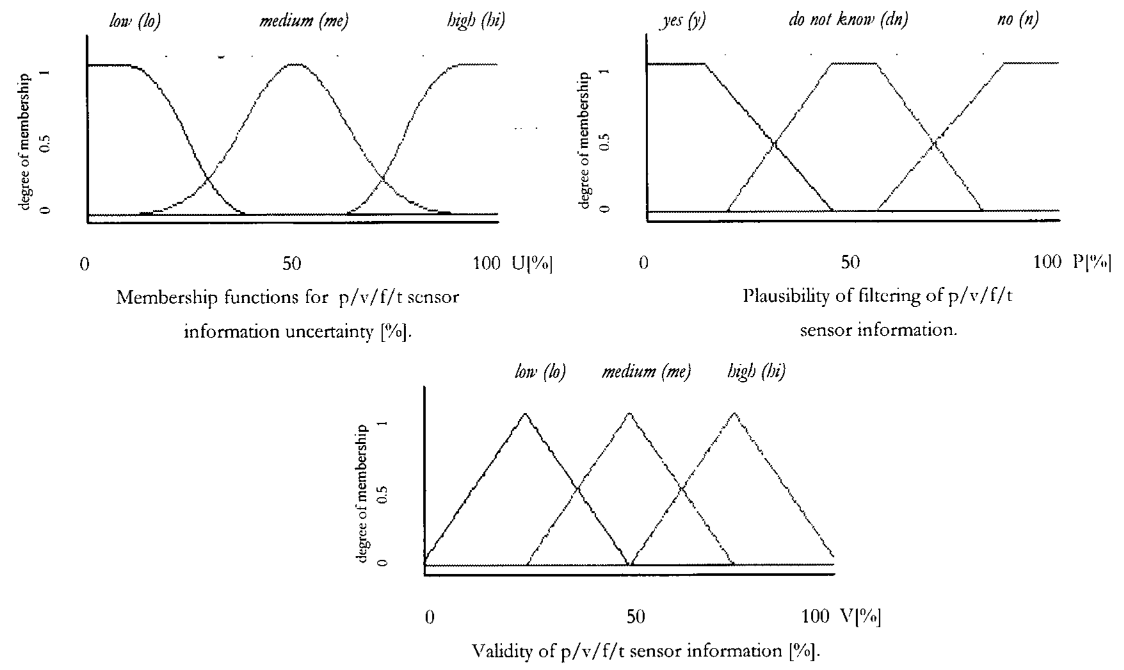


Figure 4.7. The membership functions for sensor information module.

```
If U is lo and P is dn  then V is me,  W=0.7
else if U is hi or P is n  then V is hi,  W=0.5
                                  :
end
```

Table 4.3. The part of rulebase for validity of sensor information.

4.2.5.4 Hardware Module

The computational speed of the controller hardware is important in the sense of allowable control architecture complexity to be applied. This issue is represented over membership functions used to describe computer computation rate and transfer I/O rate.

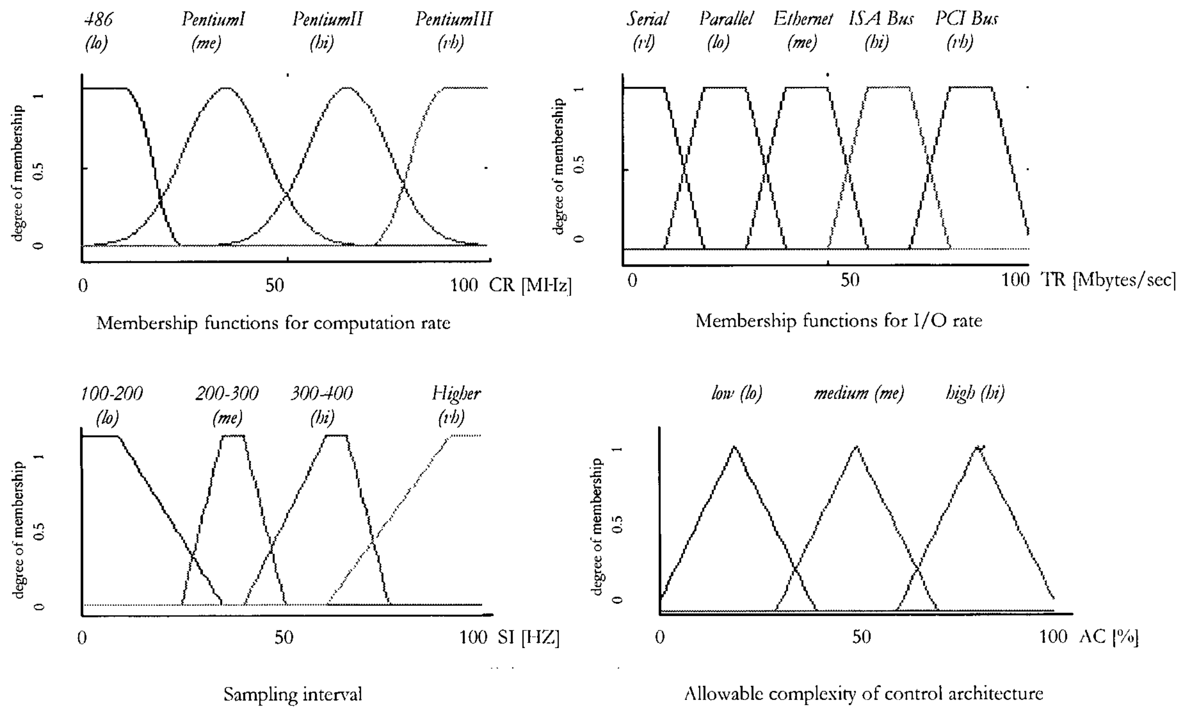


Figure 4.8. The membership functions for hardware module.

The fuzzy intervals used for computation rate and for signal transfer rate are to give the user an idea of comparable computational abilities of the system. For example, operating on PC Pentium II which **computation rate** ranges from 233-450 MHz, and using a **transfer type** 32-bit PCI Bus with **transfer rate** 33 MHz-132 Mbytes/sec, and with imposed **sampling interval** of 500 Hz will be sufficient hardware capacity for real-time control with some complex control architectures as hybrid/impedance control or adaptive control. Thus, the allowable architecture is function of the following variables:

Architecture complexity = *f*(**computation rate/transfer rate/sampling interval**) (4.7)

The rulebase comprises 18 rules, which are defined as is shown in Table 4.4.

If *CR* **is** *lo* **and** *TR* **is** *vl* **and** *SI* **is** *me* **then** *AC* **is** *lo*, W=1.0,
else if *CR* **is** *hi* **and** *TR* **is** *lo* **and** *SI* **is** *lo* **then** *AC* **is** *me*, W=0.6,
else if *CR* **is** *vh* **and** *TR* **is** *vh* **and** *SI* **is** *me* **then** *AC* **is** *hi*, W=1.0,
:
end

Table 4.4. The part of rulebase for hardware module.

4.2.5.5 Stiffness Module

In [20], it is shown that, for given link parameters, interaction between the structural stiffness of the manipulator link and the stiffnesses of the contacting regions of the manipulator tip and the environment results in distinct changes in the modal structure of the manipulator/environment system, as the **effective contact stiffness** increases.

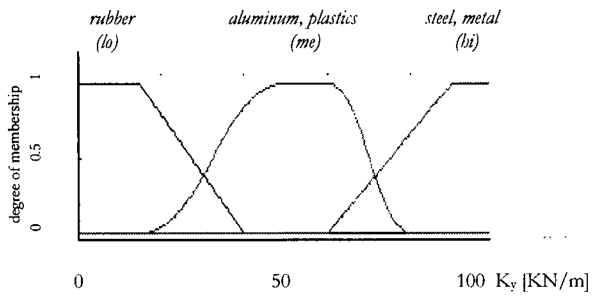
Thus, contact stiffness, which usually ranges from 50 N/m to 5 MN/m has a direct relationship to the system natural frequency. For example, for single rigid link, the effective stiffness is defined as:

$$k_{eff} = \frac{k_{sy} k_{ey}}{k_{sy} + k_{ey}}, \qquad \text{where system natural frequency is} \quad \omega_1^2 \approx \frac{k_{eff} L_1^2}{I_{o1}}$$

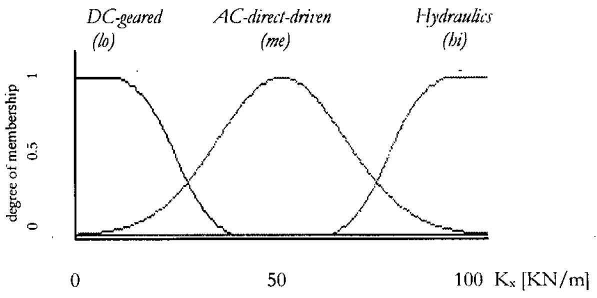
$$\text{and} \quad \omega_2^2 \approx \frac{k_{sy} + k_{ey}}{m_c}. \tag{4.8}$$

It is shown that for manipulator with two rigid links, natural frequencies are proportional to $k_{eff}^{1/2}$ for a given set of link lengths and inertias. From Shannon's theorem, the appropriate sampling interval T can be found to satisfies condition for system stability, $0 < T K_f k_{eff} \leq 1$, where, T – *sampling interval*, K_f – *force control gain* and k_{eff} – *effective stiffness.* These non-fuzzy values are computed, and used as a criterion for selection of upper and lover gain values limits of chosen controller.

For robot control architecture selection, it is appropriate to approximate overall stiffness K_{eff} of both manipulator structure and contact environment in the sense of *low, medium* and *high*. For that purpose, membership functions are created as shown on Fig. 4.9.



Membership functions for environmental stiffness



Membership functions for manipulator joint stiffness

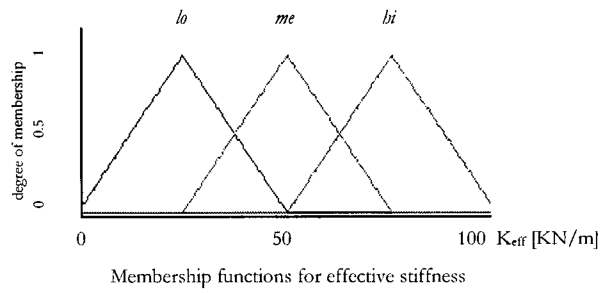


Figure 4.9. Membership functions for stiffness module.

The manipulator stiffness is defined over the type of robot joint driving system. The highest stiffness K_x is assigned to hydraulic joints. However, environment stiffness is approximated over the type of environment that end-effector is in contact with.

Thus, functions that correspond to *low* stiffness values are assigned to materials like rubber or some other soft material, whereas the highest stiffness K_y is assigned for metal, steel, etc. The final fuzzified value is obtained through the rulebase consisting of 9 rules, and the part of this rulebase is given in Table 4.5.

```
If $K_x$ is lo and $K_y$ is lo then $K_{eff}$ is lo, W=1.0,
else if $K_x$ is lo and $K_y$ is me then $K_{eff}$ is me, W=0.7,
else if $K_x$ is me and $K_y$ is hi then $K_{eff}$ is hi, W=0.6,
                                :
end
```

Table 4.5. The part of rulebase for stiffness module.

4.2.5.6 Allowable Planning Task Error

This information is important for choosing appropriate control architecture. The criteria for unconstrained motion are expressed over second finite Laplace theorem. For point-to-point motion the main parameter is final positional error that system produces under step function as an input.

$$\varepsilon_s = \lim_{t\to\infty} \varepsilon(t); \qquad \varepsilon_{\text{sp}} = \lim_{s\to 0} \left.\frac{1}{1+W_{OL}}\right|_{x_d(s)=\frac{1}{s}};$$
$$\text{criteria}: \quad \mathrm{E} = f(\varepsilon_{sp}), \tag{4.9}$$

For continuous path motion (tracking), this criterion represents the normalized standard error over unity range, for a ramp function input to the system.

$$\varepsilon_s = \lim_{t\to\infty} \varepsilon(t); \qquad \varepsilon_{\text{sv}} = \lim_{s\to 0} \left.\frac{1}{\text{s}}\frac{1}{1+W_{OL}}\right|_{x_d(s)=\frac{1}{s^2}};$$

where criteria is $\qquad$ E=*f*(normalized error/unity range), $\qquad$ (4.10)

and normalized error is defined as

$$k = \sqrt{\frac{\sum_{i=1}^{n} (\underline{x} - \overline{\underline{x}})^2}{n-1}};$$

$\underline{x}$ - state vector, criteria is $\mathrm{E} = f(k)$. $\qquad$ (4.11)

For constrained motion, the criteria are related to practical stability analysis, which is beyond the scope of this work. Instead, the criteria for allowable planning error for contact tasks will be substituted. The membership functions are shown on Figure 4.10.

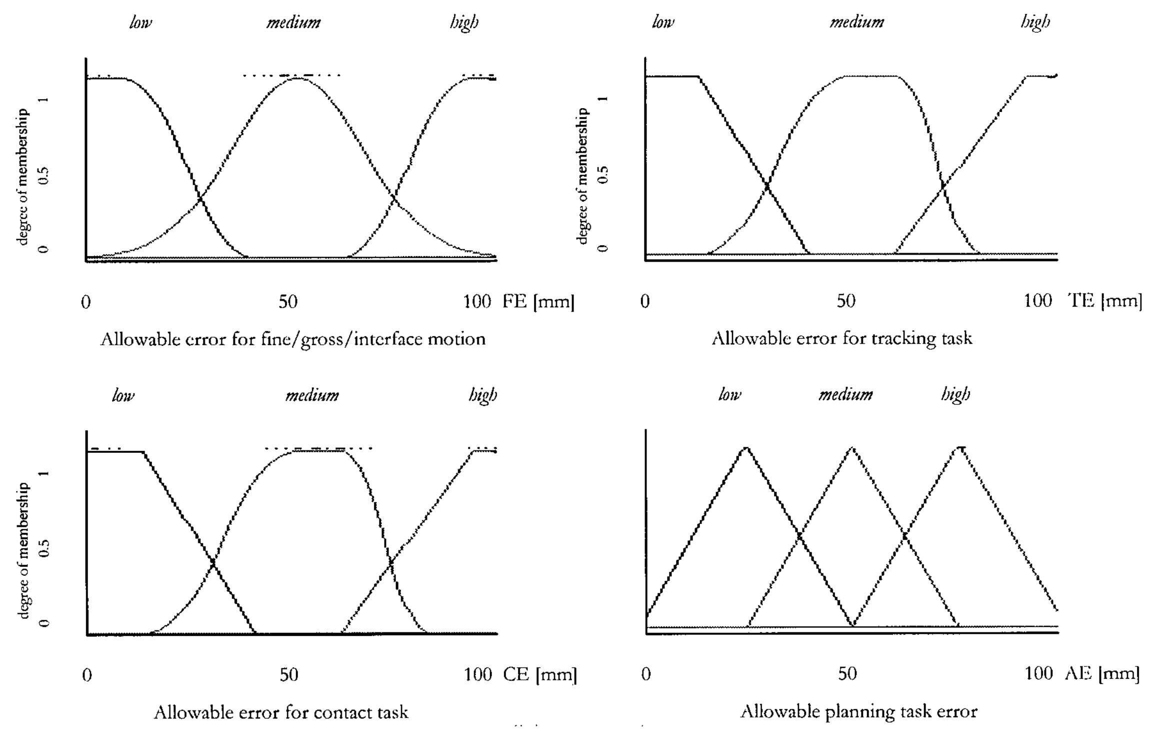


Figure 4.10. Membership functions for allowable planning task error.

The part of rulebase is shown in Table 4.6.

If *FE* **is** *lo* **and** *TE* **is** *lo* **and** *CE* **is** *hi* **then** *AE* **is** *me*, W=0.9,

else *FE* **is** *me* **and** *TE* **is** *lo* **and** *CE* **is** *hi* **then** *AE* **is** *me*, W=0.8,

else *FE* **is** *hi* **and** *TE* **is** *hi* **and** *CE* **is** *hi* **then** *AE* **is** *hi*, W=1.0,

:

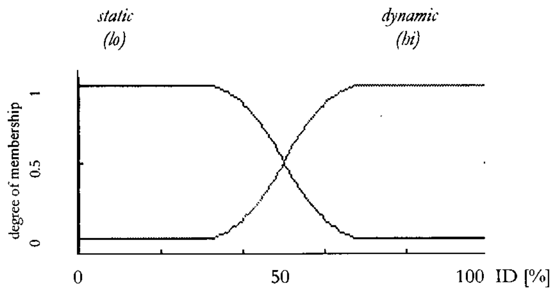
end

Table 4.6. A part of rulebase for allowable planning task error.

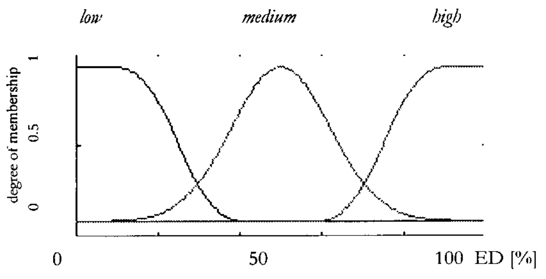
4.2.5.7 Disturbances Module

The effect of disturbances is introduced in fuzzy calculations over the membership functions for **external (environmental) disturbances** that are represented with the fuzzy values, *low* (for example, variable contact caused by deformation in plastic, rubber etc.),

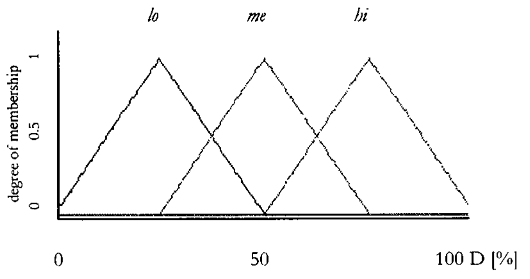
medium (e.g. variable contact caused by deformation of wood, aluminum etc.), and *high* (e.g. variable contact caused by deformation of metal), as well as for **internal (manipulator) disturbances** expressed over fuzzy values for *static* (gravitational moment) and *dynamic* (external moment, coupling etc.) *disturbances.*



Membership functions for type of internal disturbances



Membership functions for type of external disturbances



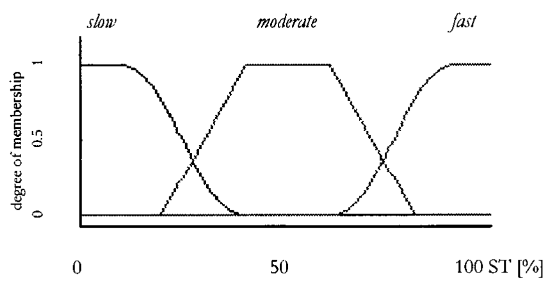
Membership functions overall disturbances

Figure 4.11. Membership functions for disturbances module.

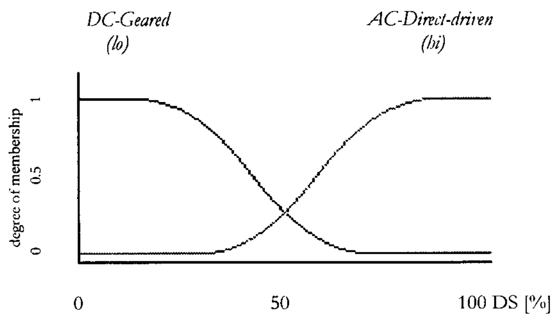
4.2.5.8 Control Structure Module

The control structure module is a part of the control module in the expert system architecture that is responsible according to the available information for making decision about the structure of control algorithm, whether it is going to be **decentralized** or **centralized**.

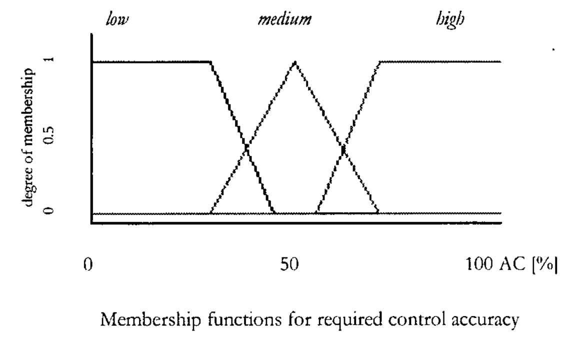
The control structure module is in function of its main parameters, trajectory rate, driving system of joints, rate of response, and required control accuracy. Decentralized control has some advantages and disadvantages over centralized structure. First of all, it is less complex than calculating a nominal centralized control. It is relatively simple to accomplish this type of control on-line, whereas doing on-line centralized control sometimes can be tedious and difficult. However, robot stabilization around the nominal trajectory is poor when using local controllers that do not take into account the dynamic coupling between the joints. On the other hand, direct-drive robots (equivalent moment of inertia is small) have become more commonly used. In such cases, the use of local controllers is not satisfactory, and a centralized structure has to be applied. All these issues and criteria are considered for designing rule base for this module that comprises 15 rules. The membership functions are shown on Figure 4.12.

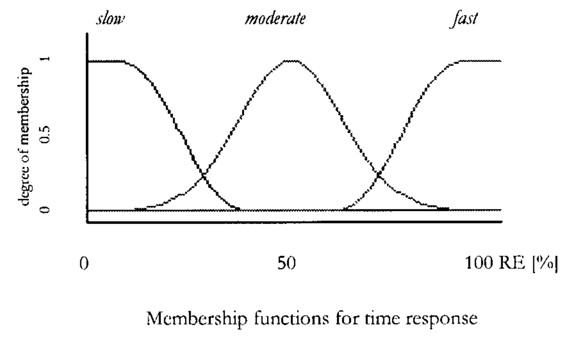


Membership functions for trajectory rate



Membership functions for driving system of robot





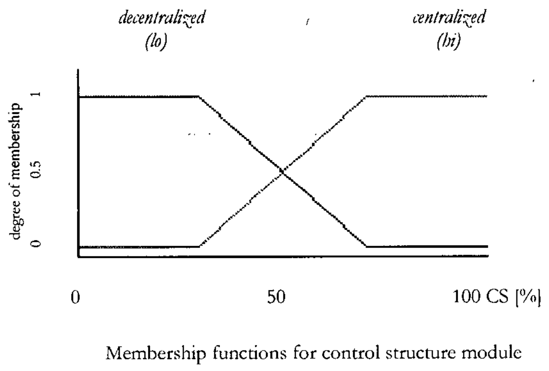


Figure 4.12. Membership functions for control structure module.

```
If ST is lo and DS is lo  and RE is hi and CA is hi then CS is lo,  W=0.9,
else ST is mo and DS is lo  and RE is hi and CA is hi then CS is lo,  W=0.8,
else ST is hi and DS is hi  and RE is hi and CA is me then CS is hi,  W=1.0,
                                      :
end
```

Table 4.7. A part of rulebase for control structure module.

4.2.5.9 Controller Type Module

This module is responsible for making decision about the low-level controller to be incorporated into the control architecture based on the criteria set by a user. Thus, the main parameters for choosing the controller are, allowable overshoot, degree of stability, required rise time and allowable error. The module consists of eleven rules and membership functions are derived and presented on next figures.

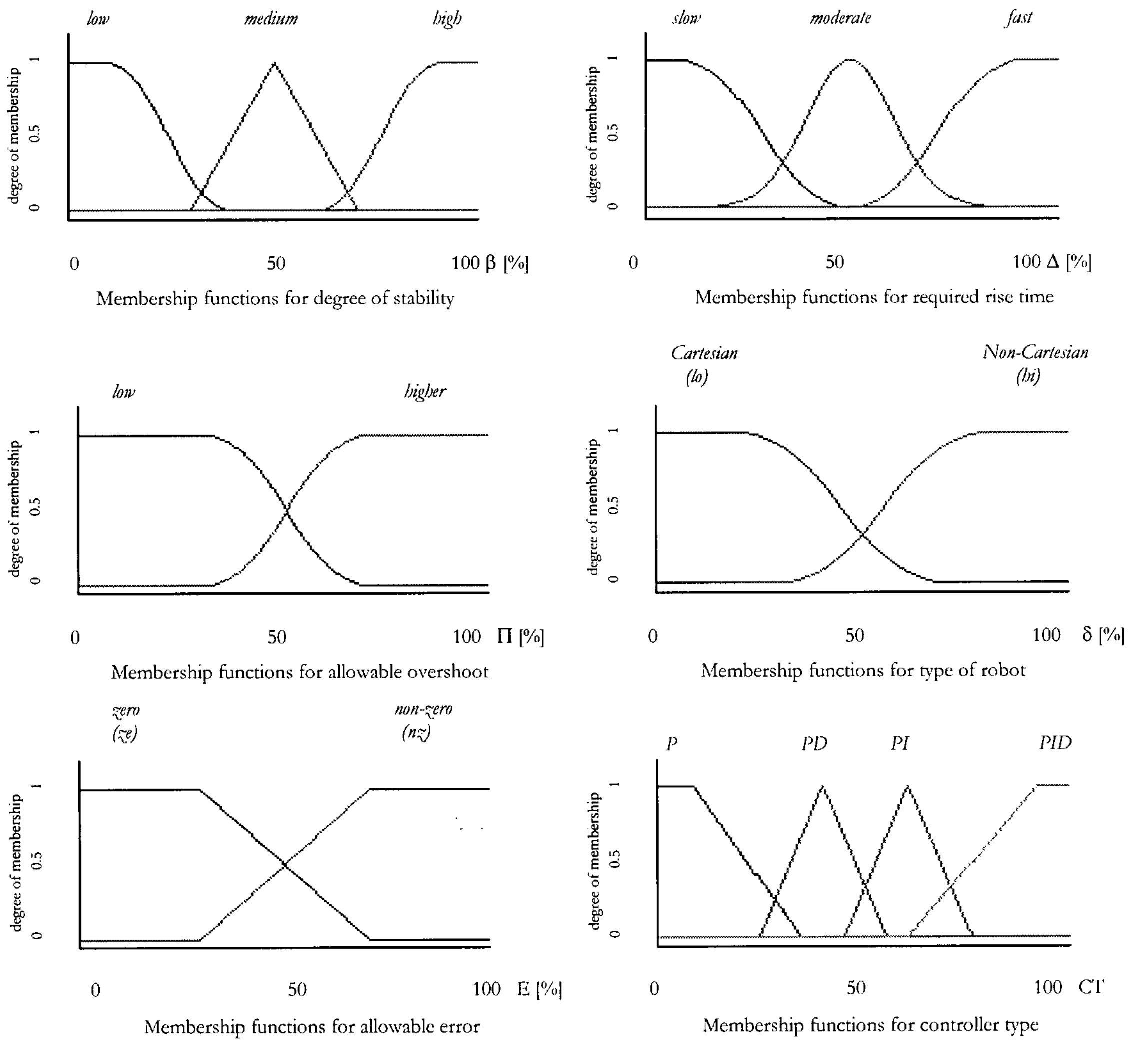


Figure 4.13. Membership functions for controller type module.

If β **is** *hi* **and** Π **is** *lo* **and** Δ **is** *lo* **and** δ **is** *hi* **and** E **is** *nz* **then** *CT* **is** *P*, W=0.9,

else if β **is** *lo* **and** Π **is** *lo* **and** Δ **is** *lo* **and** δ **is** *hi* **and** E **is** *z* **then** *CT* **is** *P*, W=0.7,

else if β **is** *hi* **and** Π **is** *hi* **and** Δ **is** *lo* **and** δ **is** *hi* **and** E **is** *z* **then** *CT* **is** *PID*, W=0.7,

:

end

Table 4.8. The part of rulebase for controller type module.

4.2.6 Summary

In this section a taxonomy for robot control, representing on robot tasks, motion dynamics, control architectures and controllers. This taxonomy is generated based on identified characteristics important to the selection of robot control architecture and robot controller.

Based on this taxonomy, a robot control expert system, consisting of controller architecture module (CAM) and a controller module (CM) have been developed. In the following chapters a set of simulations and experiments are presented which verify the control scheme selections made by the expert system.

Chapter V

Simulation Results

In this chapter, the results obtained through the simulation of two-link planar arm for various types of robot tasks are presented. The objective of these simulations is to verify the decisions made by the TREX system. A comparison of the selected controller system (architecture and controller) against the second choice system is presented in such case.

5.1 The Two-Link Planar Arm

For the simulation result purposes, the two-link planar arm (Figure 5.1) is considered. The simulation of the 2-dof robot is performed using the Matlab Robotics Toolbox, and appropriate model-based block diagrams schemes are derived in Simulink. The dynamic model is presented in Appendix A and adopted parameters of planar arm are given in Table 5.1.

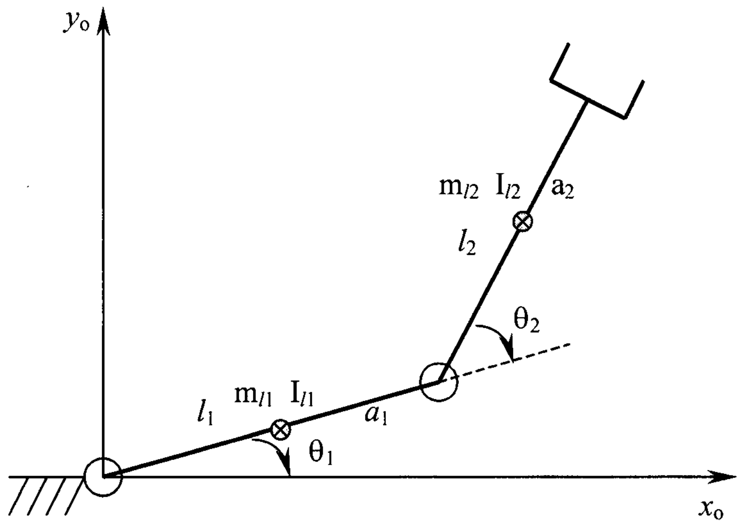


Figure 5.1. Two-link planar arm.

Parameter	Symbol	Value
generalized joint coordinates	$q = (q_1, q_2)$	variable
functional S-coordinates	$s = (x, y)$	variable
mass of the first link	m_{l1}	50 kg
mass of the second link	m_{l2}	50 kg
moment of inertia with respect to the axis of first rotor	I_{m1}	10 kg m^2
moment of inertia with respect to the axis of second rotor	I_{m2}	10 kg m^2
moment of inertia relative to the center of mass of the first link	I_{l1}	0.01 kg m^2
moment of inertia relative to the center of mass of the second link	I_{l2}	0.01 kg m^2
mass of the first rotor	m_{m1}	5 kg
mass of the second rotor	m_{m2}	5 kg
first link length	a_1	1 m
second link length	a_1	1 m
distance of the center of mass of first link from the first joint axis	l_1	0.5 m
distance of the center of mass of second link from second joint axis	l_2	0.5 m
viscous friction coefficient at the motor shafts	$F_{m1} = F_{m2}$	0.01 N m s/rad
actuators' armature resistance	$R_{a1} = R_{a1}$	10 ohm
actuators' torque constant	$k_{t1} = k_{t2}$	2 N m/A
actuators' voltage constant	$k_{v1} = k_{v2}$	2 V s/rad.

Table 5.1. The parameters of two-link planar arm.

The motion of the controlled arm was simulated on PC Pentium 120 MHz, by adopting a discrete-time implementation of different control schemes with two distinctive sampling intervals, 1 ms and 5 ms for both constrained and free arm motion.

For this robot, a typical, general-purpose 'robust' independent joint control architecture with PD controller may be used. The results for these general-purpose architecture compared to the control architectures selected by the expert system, for given criteria, are given in the following sub-sections.

5.2 Continuous Path Motion Task

5.2.1 Task Description

The desired arm trajectory has a typical trapezoidal velocity profile. The end-effector desired motion is a path along the joint space rectilinear path from $q_i=[0\ \pi/4]^T$ to $q_f=[\pi/2\ \ \pi/2]^T$ posture. The trajectory duration is $t_f = 1$ sec.

5.2.2 Criteria and Available Information

The criteria and information available to the expert system are given in the Table 5.2.

Criteria	Information Available
- precise positioning with minimal error - high-speed trajectory - good disturbances rejection - small initial oscillations about desired path	- position and velocity sensor information available - validity of sensor information at *'high'* level - level of disturbances at *'medium'* level - hardware capability at *'medium'* level - model uncertainties at *'low'* level

Table 5.2. Criteria and Available Information for CP motion task.

5.2.3 Control Architecture Selected by Expert System

According to the available information obtained by user and the task description, the expert system is required to decide which control architecture is most suitable for a given task. As shown in Figure 5.2, the CAM selects a computed torque control architecture with the highest membership. The second choice is an independent joint control architecture.

For precise positioning, a PID controller is commonly used. However, in this case, due to the simplicity of the task, a PD controller can still achieve good results, and fulfill the given criteria. The D term in PD controller, gives a necessary high-speed of the robot arm with no necessary overshoots from I term if PID controller has been utilized.

Thus, among the other architectures, a computed torque architecture with a PD controller, is the expert system's decision with the highest membership. This scheme after the tuning accomplishes the given criteria (see Figure 5.5).

5.2.4 Applied Control Law

The block diagram of utilized architecture is presented on Figure 5.3. This technique of nonlinear compensation and decoupling is very attractive from a control viewpoint since the nonlinear and coupled manipulator dynamics is replaced with *n* linear and decoupled second-order subsystems [32]. The stabilizing control law *y* is chosen as:

$$y = -K_P q - K_D \dot{q} + r, \tag{5.1}$$

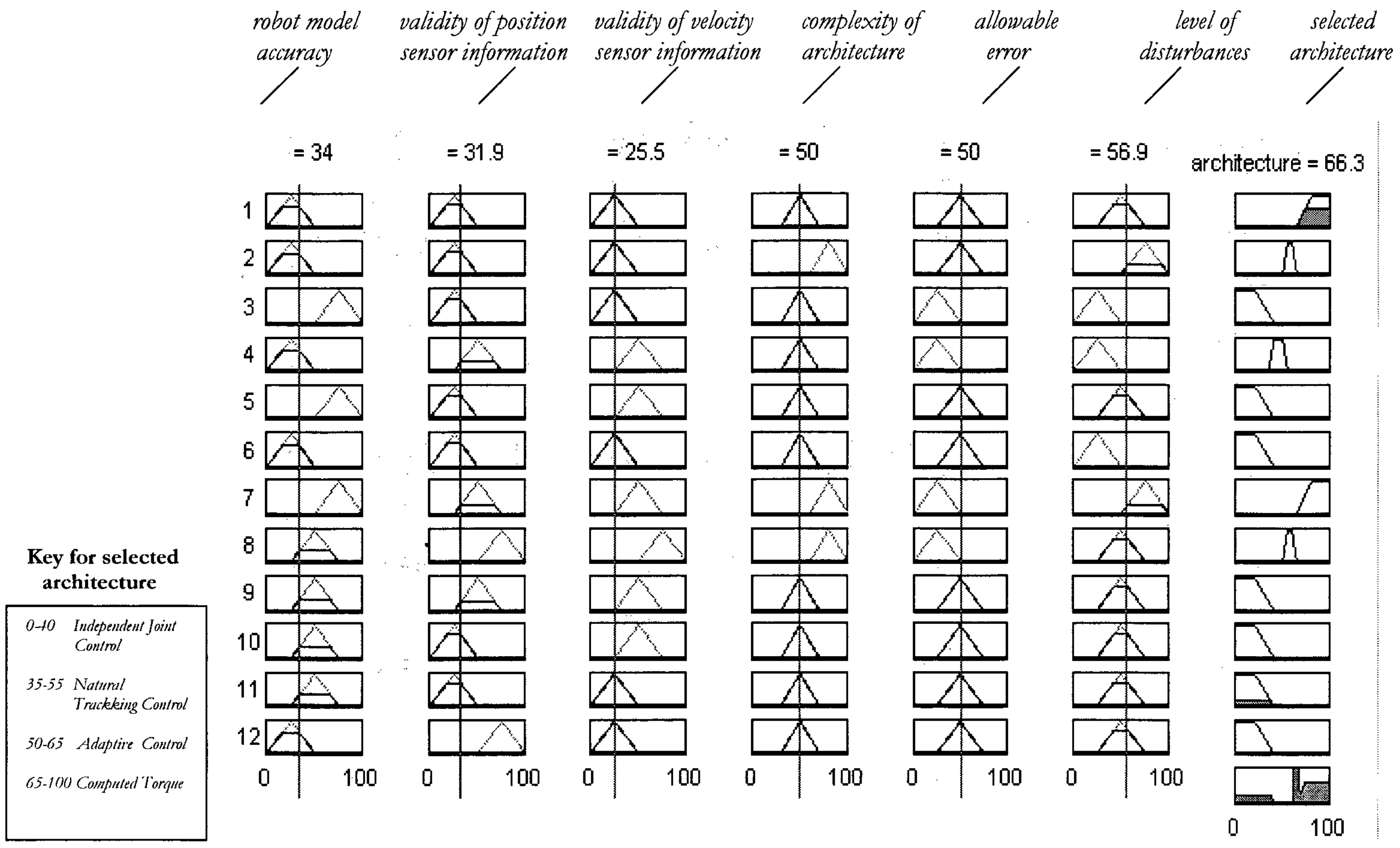


Figure 5.2. The CAM Matlab editor for free robot motion.

Given any desired trajectory $q_d(t)$, tracking of this trajectory is ensured by choosing

$$r = \ddot{q}_d + K_D \dot{q}_d + K_P q_d. \tag{5.2}$$

The applied control law is given in Appendix C.

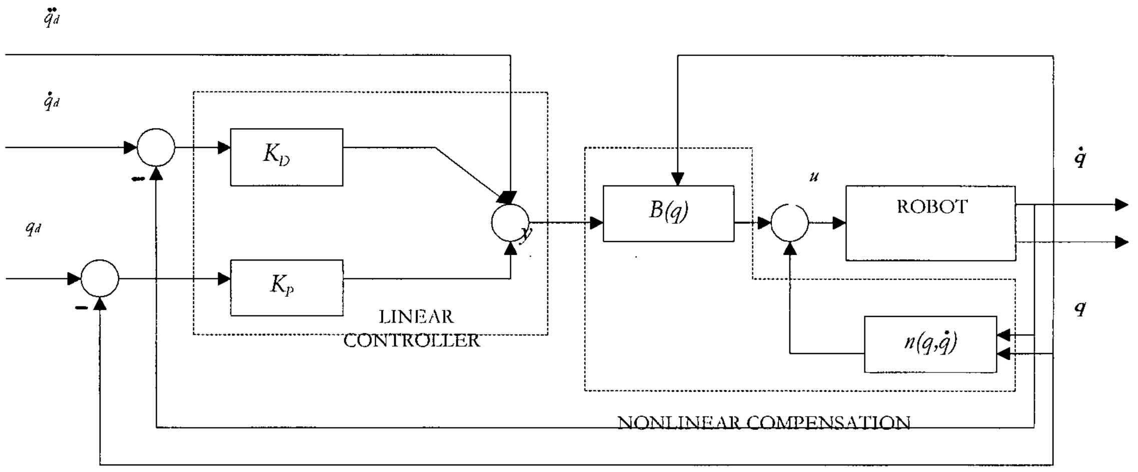


Figure 5.3. Computed torque with PD controller.

5.2.4 Discussion and Comparison of Results

Figures 5.4 and 5.5 represents the time history of the joint torques and of the tip positions errors for the same desired trajectory with different control architectures applied for this type of motion. The gains of chosen controllers are tuned manually.

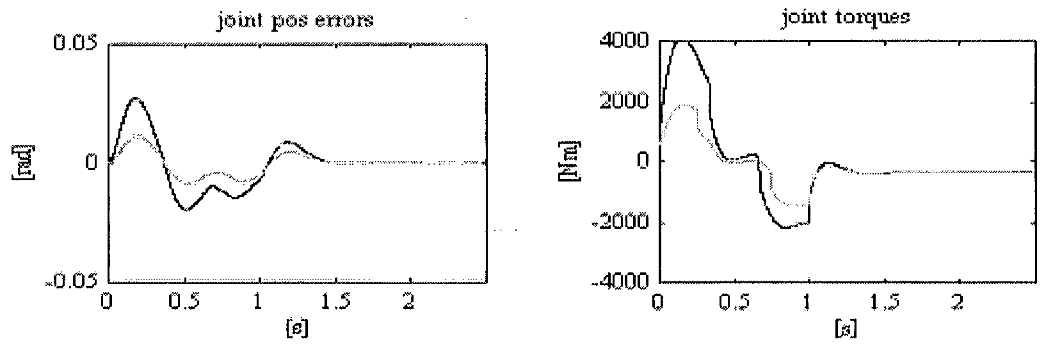


Figure 5.4. Time response with PD controller, t_s=1ms, K_P=6.25, K_V=32.

Figure 5.4 shows results obtained by using the decentralized standard (independent joint) control architecture with a PD controller. The gains of the PD actions have been tuned so as to obtain the most satisfactory response.

The next figure shows a response obtained with the architecture chosen by the expert system, namely; computed torque architecture with PD controller. This architecture achieves good disturbance rejection, and outperforms a previous scheme, especially after the controller tuning.

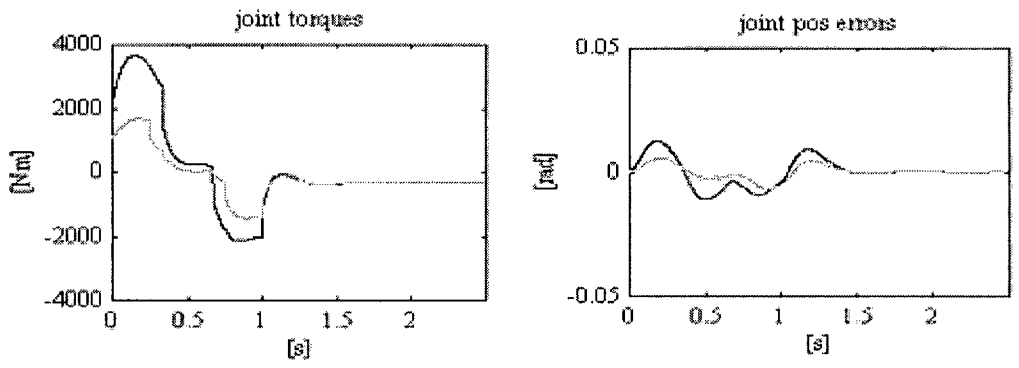


Figure 5.5. Computed torque with PD controller, t_s=1ms K_P=6.25, K_V=32.

5.3 Continuous Path Motion Task with Tip Payload

5.3.1 Task Description

The desired arm trajectory has a typical trapezoidal velocity profile. The end-effector desired motion is a path along the joint space rectilinear path from $q_i=[0\ \ \pi/4]^T$ to $q_f=[\pi/2\ \ \pi/2]^T$ posture. The concentrated tip payload is of weight m_L = 10 kg. The trajectory duration is t_f=1 sec.

5.3.2 Criteria and Available Information

The criteria and information available to the expert system are given in the Table 5.3.

Criteria	Information Available
- tracking with small error - low-speed trajectory - small initial oscillations about desired path - small settling time	- position and velocity sensor information available - validity of sensor information at *'high'* level - level of disturbances at *'high'* level - hardware capability at *'high'* level - model uncertainties at *'high'* level - sampling interval of controller at *'low'* level - stiffness at *'low'* level

Table 5.3. Criteria and available information for CP motion task with tip payload.

5.3.3 Control Architecture Selected by Expert System

According to the available information obtained by user and the task description (Table 5.3), the expert system is required to decide which control architecture is most suitable for a given task. The main difference between this task and the task explained in previous section is in the changing tip payload and speed of trajectory. By introducing the payload at the tip of end-effector, the dynamics of the system is changed (link masses, inertia etc.).

Thus, there must be either some additional sub-system introduced into the control loop that will compensate for these changes or the calculation of input signals to the control architecture must be based on estimation and adaptation of certain parameters. The most important information that characterize this case are: a 'high' level of uncertainties involved in robot model and a 'high' level of disturbances, which in this case represents 'internal' type of disturbances. According to these and other information gathered from user, the expert system selects the adaptive control architecture with the highest membership. Second choice is independent joint controller and third is computer torque control architecture (Figure 5.6).

For the controller, the same reasoning is used in this case as in the previous one. For precise positioning, a PID controller is commonly used. In this case one of the criteria specifies a 'small' error and not zero state error which indicates it may not be necessary to implement the controller with the integrator. A PD controller can still achieve good results, and fulfill the given criteria. The PD controller, gives the necessary high-speed of the robot arm with no overshoot.

Thus, among the other architectures, the adaptive control with PD controller, is expert system's decision with the highest membership. This scheme after the tuning accomplishes the given criteria (see Figure 5.10).

5.3.4 The Control Law

The block diagram of utilized architecture is presented on Figure 5.7. The possibility of finding adaptive control laws is ensured by the property of linearity in the parameters of the dynamic model of the manipulator.

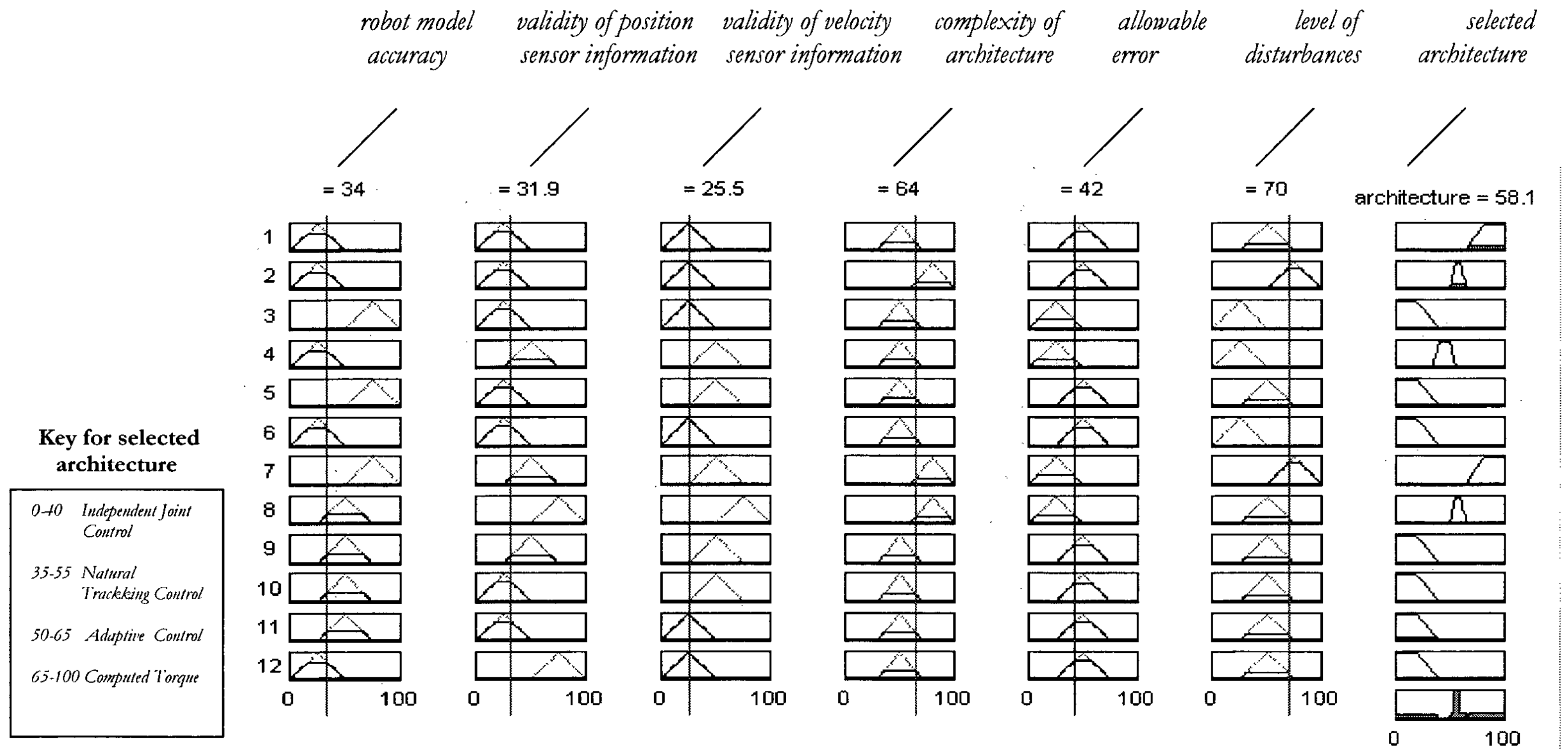

robot model accuracy
validity of position sensor information
validity of velocity sensor information
complexity of architecture
allowable error
level of disturbances
selected architecture
= 34
= 31.9
= 25.5
= 64
= 42
= 70
architecture = 58.1
1
2
3
4
5
6
7
8
9
10
11
12
0
100
Key for selected architecture
0-40 Independent Joint Control
35-55 Natural Trackking Control
50-65 Adaptive Control
65-100 Computed Torque

Figure 5.6. The CAM Matlab editor for CP motion with tip payload.

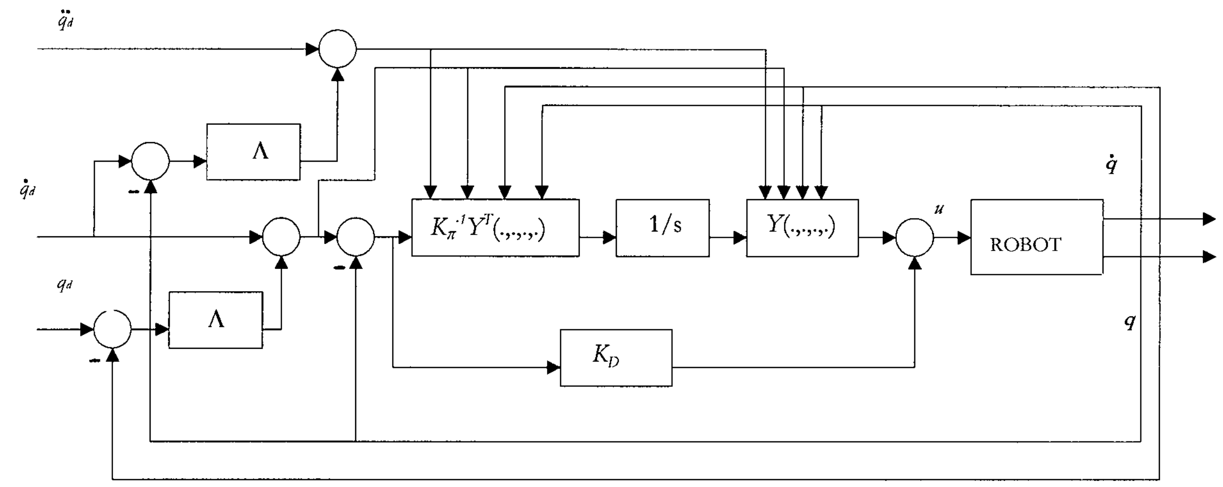


Figure 5.7. Adaptive control architecture with PD controller.

The control law is given as

$$u = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\pi} + K_D(\dot{\tilde{q}}+\Lambda\tilde{q}), \tag{5.3}$$

and parameter adaptive law is

$$\dot{\hat{\pi}} = K_\pi^{-1} Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)(\dot{\tilde{q}}+\Lambda\tilde{q}). \tag{5.4}$$

The complete control law is described in Appendix C.

5.3.5 Discussion and Comparison of Results

The figures 5.8, 5.9. and 5.10 represent the time history of the joint torques and of the tip positions errors for the same desired trajectory with different control architectures applied for this task. In order to emphasize the importance of proper decision made by expert system, comparison are made between two schemes used very often where some type of compensation is necessary, namely, inverse dynamic control with nonlinear compensation and adaptive control scheme with payload estimation and parameter estimation.

Figure 5.8 shows results obtained by using the inverse dynamic control architecture with PD controller and no compensation for load changes, implemented in operational space. This scheme cannot stabilize a robot movement about desired path and shows unsatisfactory results. The same scheme is applied with load compensation and results are plotted on Figure 5.9, and this scheme tends to stabilize robot motion but still perform sluggish behavior. The main reason for getting rather poor results with the use of these schemes is because, implementation of inverse dynamic control laws requires that parameters of the system dynamic model are accurately known. On the other hand, the model is usually known with a certain degree of uncertainty due to imperfect knowledge of manipulator mechanical parameters, existence of unmodeled dynamics, and model dependence on end-effector payloads not exactly known and thus not perfectly compensated.

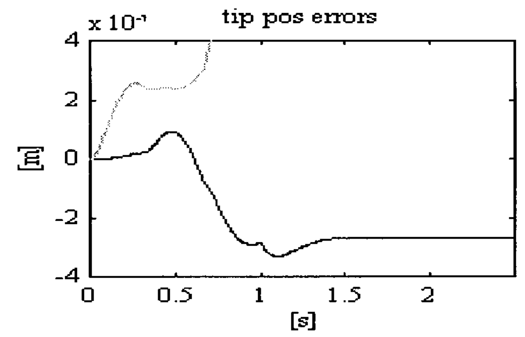


Figure 5.8. Inverse Dynamic Control with no load compensation, t_s = 1 ms.

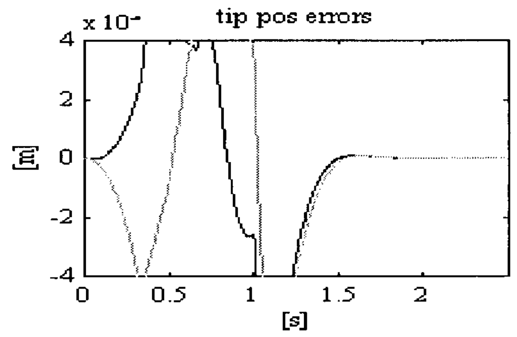


Figure 5.9. Inverse Dynamic Control with load compensation, t_s = 1 ms.

In the case of 'imperfect' compensation, a viable approach is the introduction of an additional term that will provide 'robustness' to the control system. Thus, robust control architecture that is designed to cope primarily with model uncertainties and outperforms inverse dynamic control with the same imposed condition (gains, sampling interval), could be used in this case but requires an increase of the sampling time needed to compute the control law. The gains of chosen controllers are tuned manually.

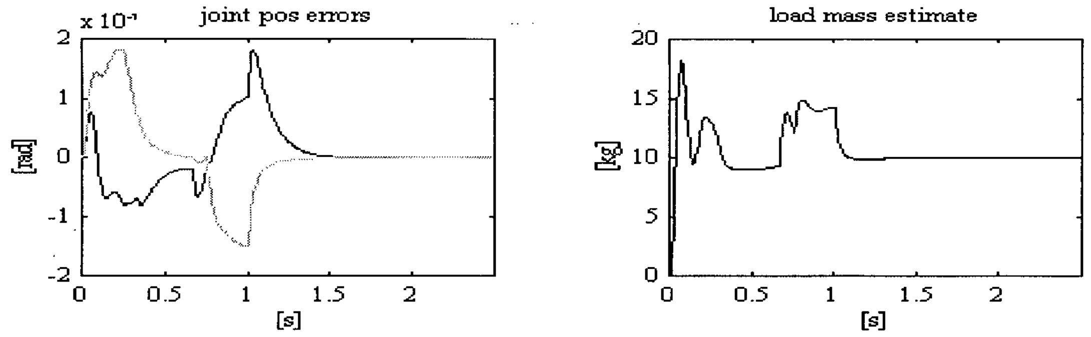


Figure 5.10. Adaptive control architecture with load mass estimation, t_s = 1ms.

5.4 Continuous Path Motion with Applied Force

5.4.1 Task Description

The desired end-effector trajectory has a typical trapezoidal velocity profile, and robot is in contact with elastically compliant plane. The elastic plane is purely frictionless. The end-effector desired motion is a path along the joint rectilinear path from position q_i=[1+0.1(2)$^{1/2}$]^T to q_f=[1.2+0.1(2)$^{1/2}$]^T.

5.4.2 Criteria and Available Information

The criteria and information available to the expert system are given in the Table 5.4.

Criteria	Information Available
- precise position control - precise force control with - slow-speed trajectory - good disturbances rejection - relatively small desired force F_d=150 N - highly stiff environment	- validity of sensor information at *'medium'* level - force, velocity, position sensor available - model accuracy at *'medium'* level - level of disturbances at *'medium'* level - hardware capability at *'high'* level - model uncertainties at *'low'* level - external disturbances at *'medium'* level

Table 5.4. Criteria and available information for CP motion with applied force.

5.4.3 Control Architecture Selected by Expert System

For this type of task, several different control architectures can be applied. Precise positioning indicates the use of impedance control, which is not suitable for this task because of the external disturbances introduced into system. Parallel control architecture always requires fully available sensor information, which is not true in this case.

The highly stiff environment suggests stiffness control architecture as a viable approach, but as this scheme has no capability for good disturbances rejection, thus, some other approach is recommended. The expert system's final decision is explicit hybrid position/force control where proportional-integral PI controller satisfies desired goals in the force control direction (see Figure 5.12). The fuzzy reasoning of CAM for this task is given over the Matlab editor shown in Figure 5.11.

Key for selected architecture

0-12	*Adaptive Control*
10-20	*Stiffness Control*
18-25	*Damping Control*
22-45	*Impedance Control*
35-60	*Hybrid Position Force Control*
50-65	*Hybrid Impedance Control*
60-80	*Explicit Position Force Control*
75-90	*Robust Control*
85-100	*Hybrid Position Torque Control*

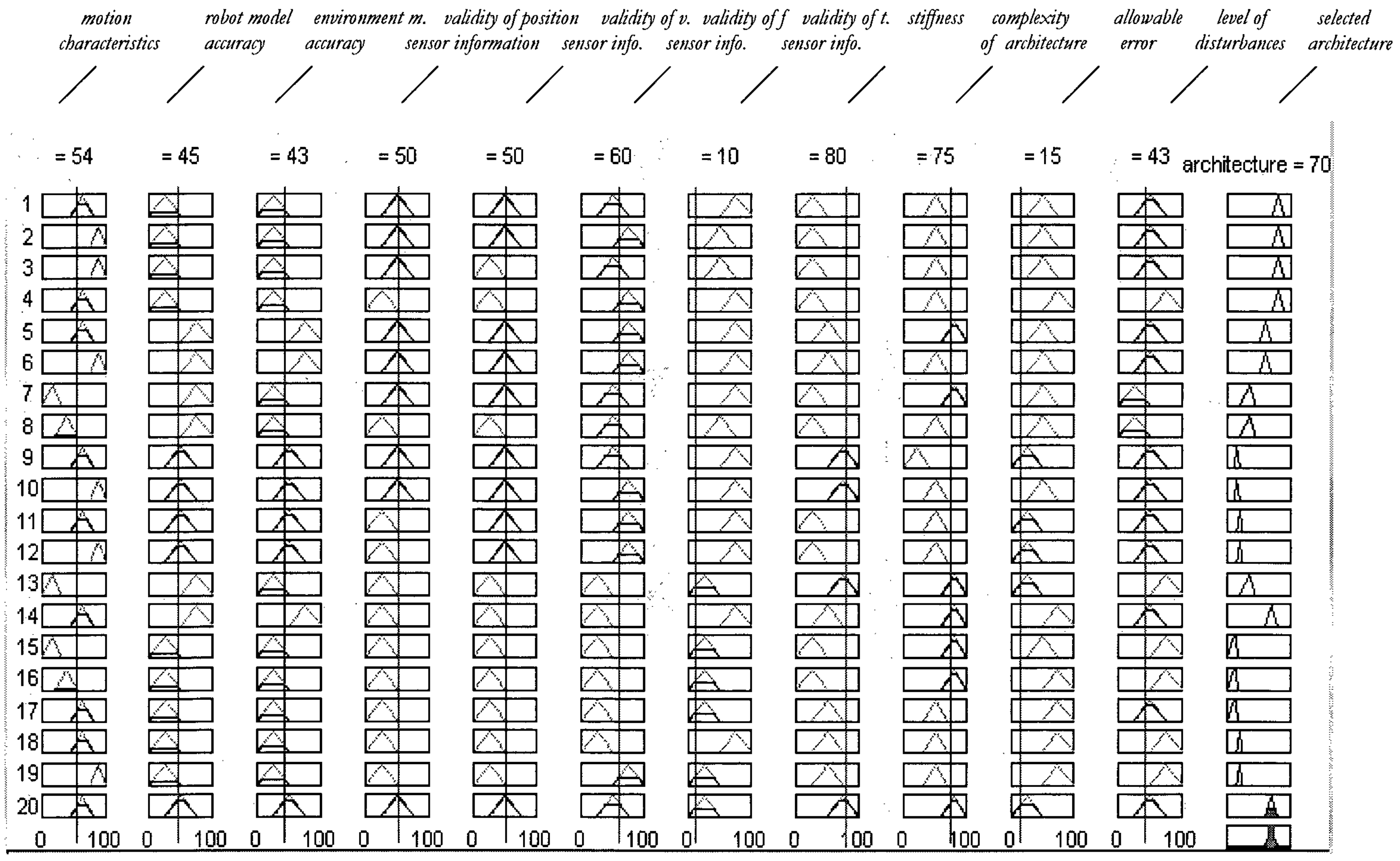


Figure 5.11. The CAM Matlab editor for constrained motion.

5.4.4 The Control Law

In the force control direction, the control law

$$y = J_A^{-1}(q) M_d^{-1}(-K_D \dot{x} + K_P(x_F - x) - M_d \dot{J}_A(q,\dot{q})\dot{q}), \tag{5.5}$$

where x_F is a suitable reference to be related to a force error and M_d is a mass matrix. The force error can be symbolically expressed as

$$x_F = S_F(h_{Ad} - h_A), \tag{5.6}$$

where S is a selection matrix whose elements give the control action. . A convenient choice for S_F is a PI action

$$S_F = K_F + K_I \int^t (.) d\zeta, \tag{5.7}$$

and with a proper choice of the matrices K_D, K_P, K_F, and K_I, the system is asymptotically stable

5.4.5 Discussion and Comparison of Results

Figures 5.12 and 5.13 show comparison of the results of the two top selected control architectures, when robot is in contact with environment. In general, hybrid position/force control (HC) with explicit force control gives better results than implicit hybrid position/force control. In general, explicit force control architecture, where force control signals are used to generate the torque inputs for the actuators in the robot's joints, has a faster response and less complex dynamics. Implicit force control has slow response to force perturbation.

For a less stiff environment, the results are almost the same. For highly stiff environment, the implicit hybrid control gives large oscillations in force control response, which produce settling time for force controller action more then 1 sec. whereas in the explicit hybrid control scheme, the settling time is less then 0.5 sec. and the error is smaller.

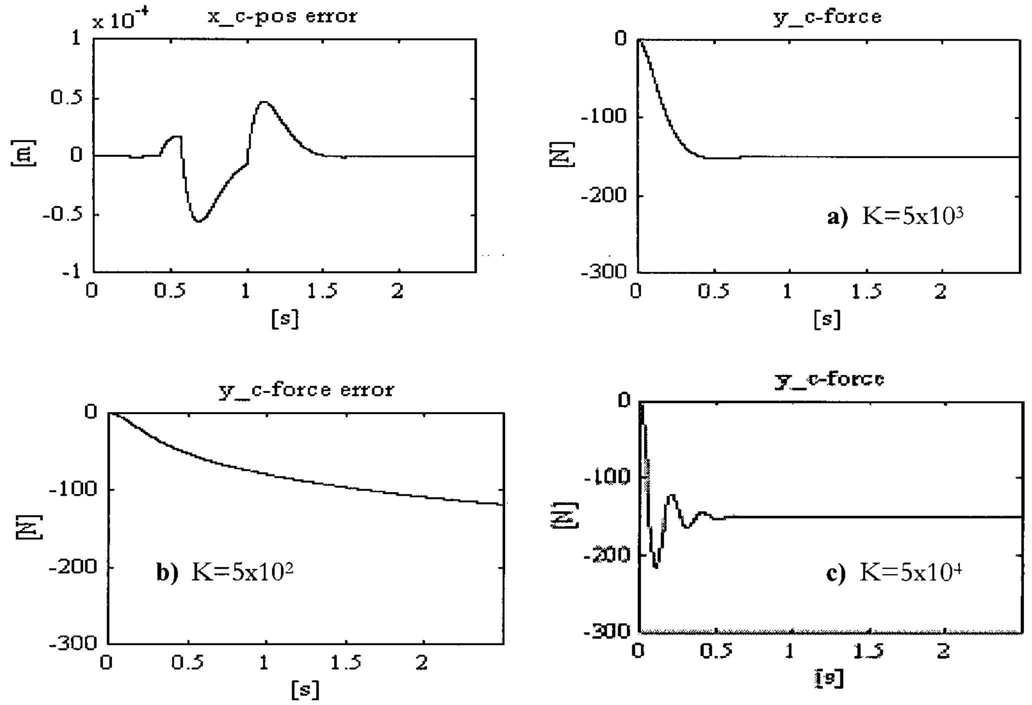
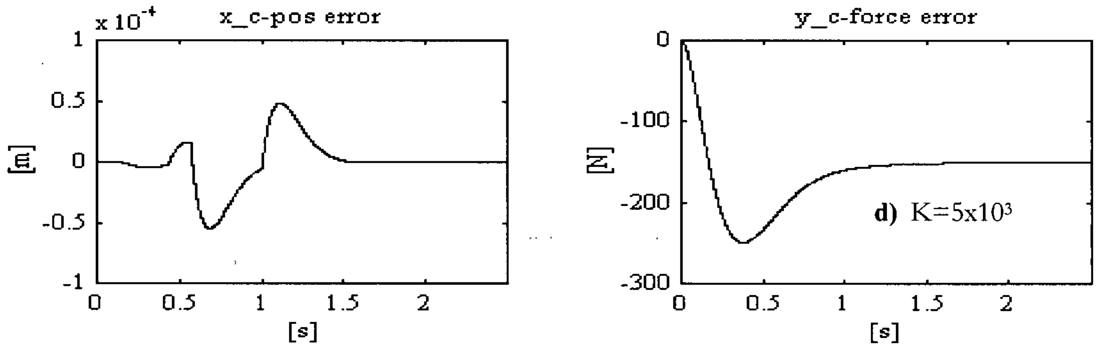


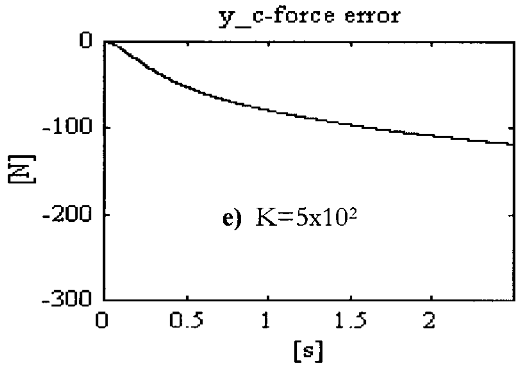
Figure 5.12. Explicit Hybrid Position/Force Control, ts=1ms, F_d=150N,

a) environment stiffness K=5x10^3

b) more compliant environment K=5x10^2

c) less compliant environment K=5x10^4





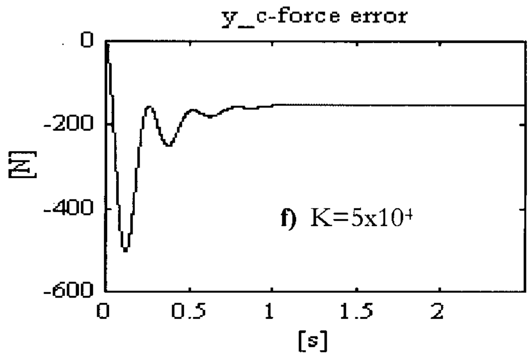


Figure 5.13. Implicit Hybrid Position/Force Control, ts=1ms, F_d=150N,

d) environment stiffness K=5x10^3

e) more compliant environment K=5x10^2

f) less compliant environment K=5x10^4

The gains of chosen controllers are tuned manually.

5.5 Summary

The simulations demonstrate how the expert system selects a control architecture-controller pair for sample tasks. The comparisons given demonstrate that, based on the rules provided, an appropriate selection is made. In the following chapter the expert system is utilized for control selection for a 5-DOF industrial robot.

Chapter VI

"I believe that nothing is unconditionally true,
and hence I am opposed to every statement of
positive truth and every man who makes it"
-H. L. Mencken

Experimental Results

In following sections, experimental results based on decision made by expert system explained in previous chapter are presented. The trajectory for both free motion and constrained robot motion has been defined in an operational space using the MATLAB Robotics Toolbox [66]. The "Control Toolbox" consisting of different control architectures and controllers is implemented in C-language, and serves as a database for designed expert system. The robot dynamic model is presented and explained in details in Appendix A.

6.1 Experimental Setup

The experiments are performed on SCORBOT ER VII 5-dof robot in the Industrial Automation Laboratory at UBC (Figure 6.1). The robot is controlled by a TMS320C32 DSP board, interfaced with two axis control cards, each capable of handling three axes simultaneously. An open architecture real-time operating system (ORTS) [55] is used in the implementation of the control algorithms and in the reading the generated trajectories and feeding them into the control loop at the different controller frequency. The signal from I/O card is sent to the amplifier from where the amplified current signal is sent to each

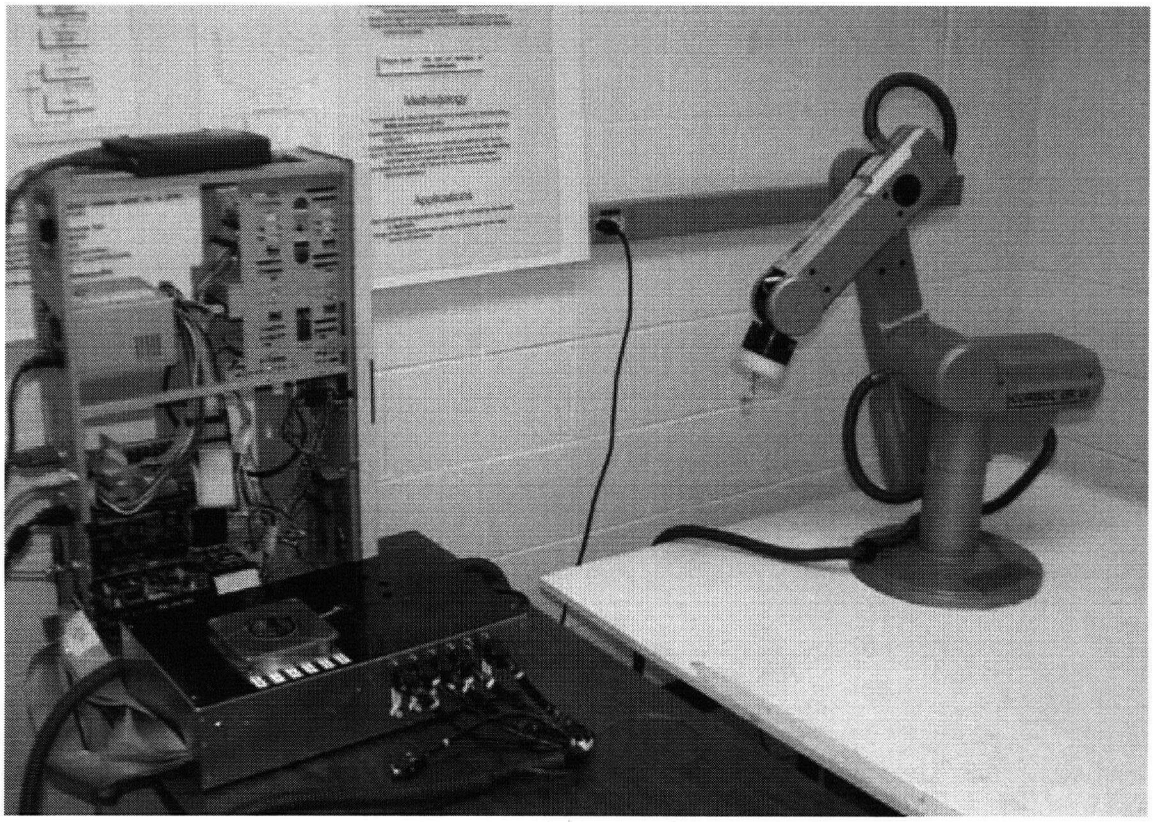


Figure 6.1: Experimental setup.

robot joint. The I/O cards and a DSP board were developed in the Manufacturing Automation Laboratory, UBC. The hardware architecture is shown on Figure 6.2.

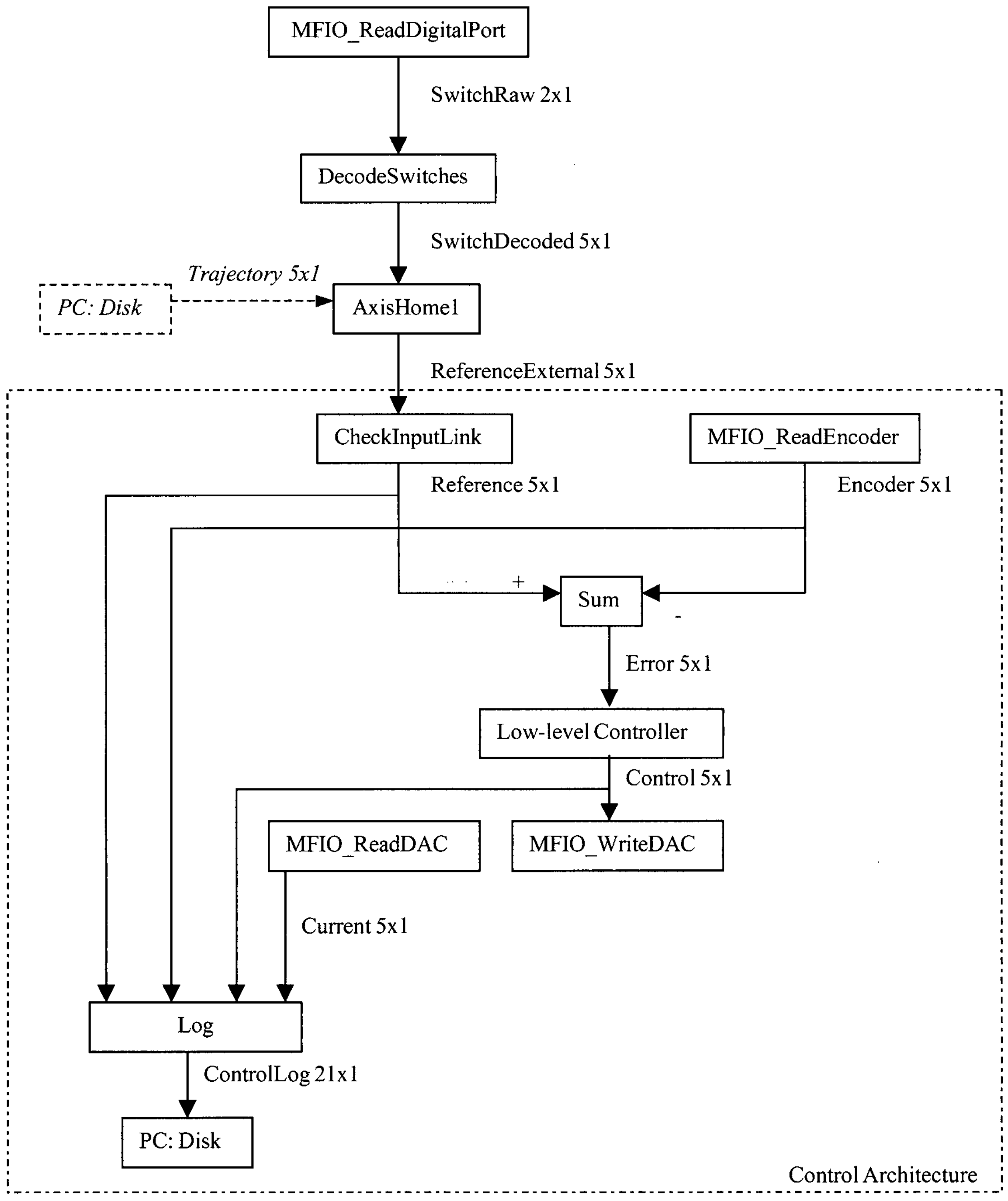


Figure 6.2. The hardware control architecture.

MFIO_*ReadDigitalPort* does the initialization of the amplifier channels, and *AxisHome1* serves for axes calibration and for setting the robot in its "home" position. *Sum* makes comparison of reference (desired) signal from *AxisHome1* and the actual signal from MFIO_*ReadEncoder*, and the output (error) from *Sum* is the input to discrete-digital *low-level-controller*. Then, the new set of control signals for robot control are generated. MFIO_*WriteDAC* applies that signal and finally, *Log* file summarizes all information and saves it to the hard disk as a matrix consisting of 21 columns.

6.2 Point-to-Point Motion Task

The point-to-point robot motion comprises three different phases. Firstly, the robot moving from its "home" position to initial point in Cartesian space, secondly, performing point-to-point motion task, and thirdly, moving back from final reached point to "home" position. Two basic point-to-point motion tasks in operational space with no obstacles through these experiments are considered: firstly, performing robot task with desired slow trajectory and high-speed trajectory, and secondly, performing task with and without the payload concentrated at the tip of end-effector.

6.2.1 Point-to-Point Trajectory Generation

Using the MATLAB Robotics Toolbox, a trajectory for a point-to-point (PTP) motion in operational space has been defined with a function:

PTP_traj(r_i, r_f, time, t_s, no_points, pause_time)

where `ri`	- initial point, defined with three points x, y, z in Cartesian/operational space, 3x1 vector function, [m].
`rf`	- final point, defined with three points η, γ, φ in Cartesian/operational space, 3x1 vector function, [m].
time	- total time along the path [sec].
ts	- sampling interval [sec].
no_points	- number of points along the desired path.
pause_time	- pause time that robot takes to stabilize its movement around the initial and final point.

6.2.2 Task Description

Available information and criteria for performing this task are obtained from user and shown in Table 6.1 and Table 6.2. All values are scaled from 0-100.

Module	Information obtained from user	Value	Membership
`motion_type`	- Type of motion, T=5%, - Following of desired/specific path (tracking), F=10%, - Control type during the contact (if exists), C=0,	DMC=15	*ptp*
`model_accuracy`	- Robot model type, RT=23, - Robot model confidence, RC=87%,	A=75%	*hi*
`position_sensor`	- Information uncertainty, IU=18%, - Signal filtered, SF=15%,	VP=86%	*hi*
`velocity_sensor`	- Information uncertainty, U=23%, - Signal filtered, SF=15%,	VV=78%	*hi*
`hardware`	- Computation rate, CR=78, - Transfer rate, TR=95, - Sampling rate, SI=49,	AC=80%	*me*
`disturbances`	- External disturbances, ED=0%, - Internal disturbances, ID=15%,	D=20%	*lo*

Table 6.1. Available information for control architecture selection.

Module	Criteria	Value
`control_structure`	- trajectory rate, ST= 12 - driving system of the joints, DS= 20 - response, RE= 80 - control accuracy, AC=50	**CS=29** *0-70 decentralized* *50-100 centralized*
`positioning_ controller_type`	- degree of stability, β= 33 - overshoot, Π= 40 - rise time, Δ= 80 - allowable error, E= 0 - robot type, RT=10 - introduced payload, P=0	**CT=81** *0-22 P controller* *15-40 PD controller* *30-54 PI controller* *50-70 PD+G controller* *65-100 PID controller*

Table 6.2. Criteria for controller selection.

6.2.3 Expert System Decision

According to the information obtained by user and the task description, the expert system is required to decide which control architecture is most suitable for a particular task. The task is a simple point-to-point motion, with low-speed trajectory and the response is required to be fast with zero-state error. The expert system selected decentralized /independent/ joint control scheme with a PID controller.

According to the given information about the task, values and membership for the expert system's modules, the priority is given to an independent control architecture as an architecture with highest membership and PID controller (Figure 6.3 and 6.4).

6.2.4 Applied Control Law

The block diagram of utilized architecture is presented on Figure 6.5. The zero-order hold is embedded into the architecture of ORTS. The low-pass filter is designed as second order system, and the applied control law is presented below.

Key for selected architecture

0-40	Independent Joint Control
35-55	Natural Tracking Control
50-65	Adaptive Control
65-100	Computed Torque

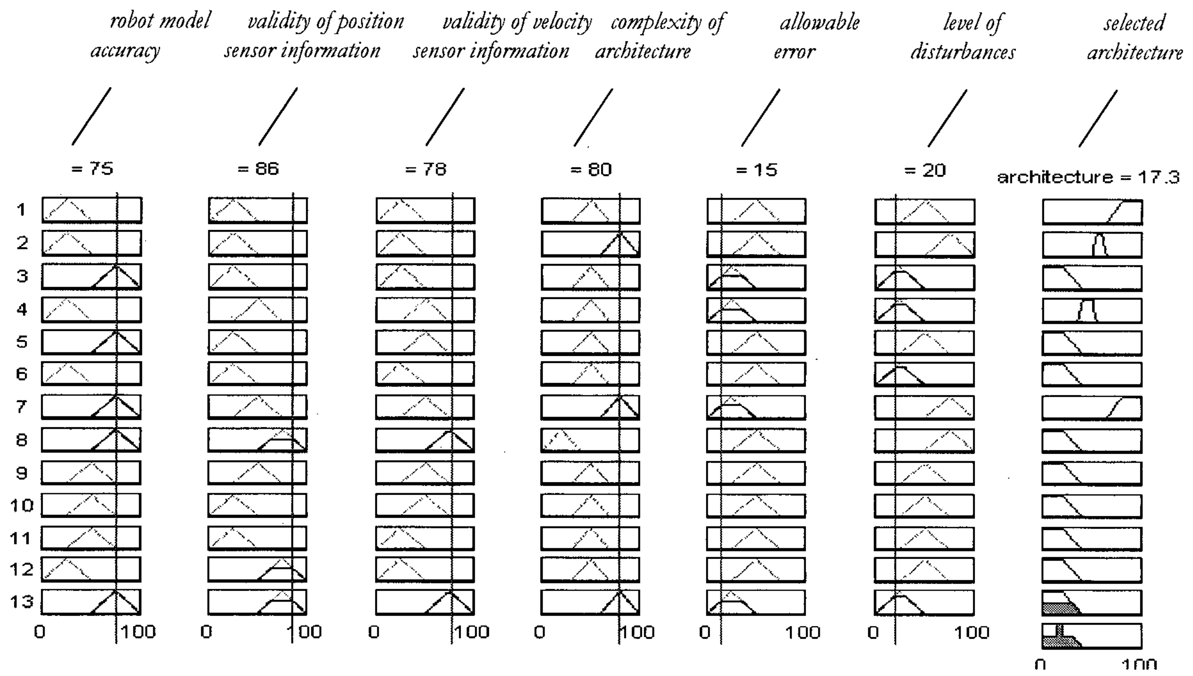


Figure 6.3. The CAM Matlab editor for free robot motion.

Key for selected controller

0-22	P controller
15-40	PD controller
30-54	PI controller
50-70	PD+G controller
65-100	PID controller

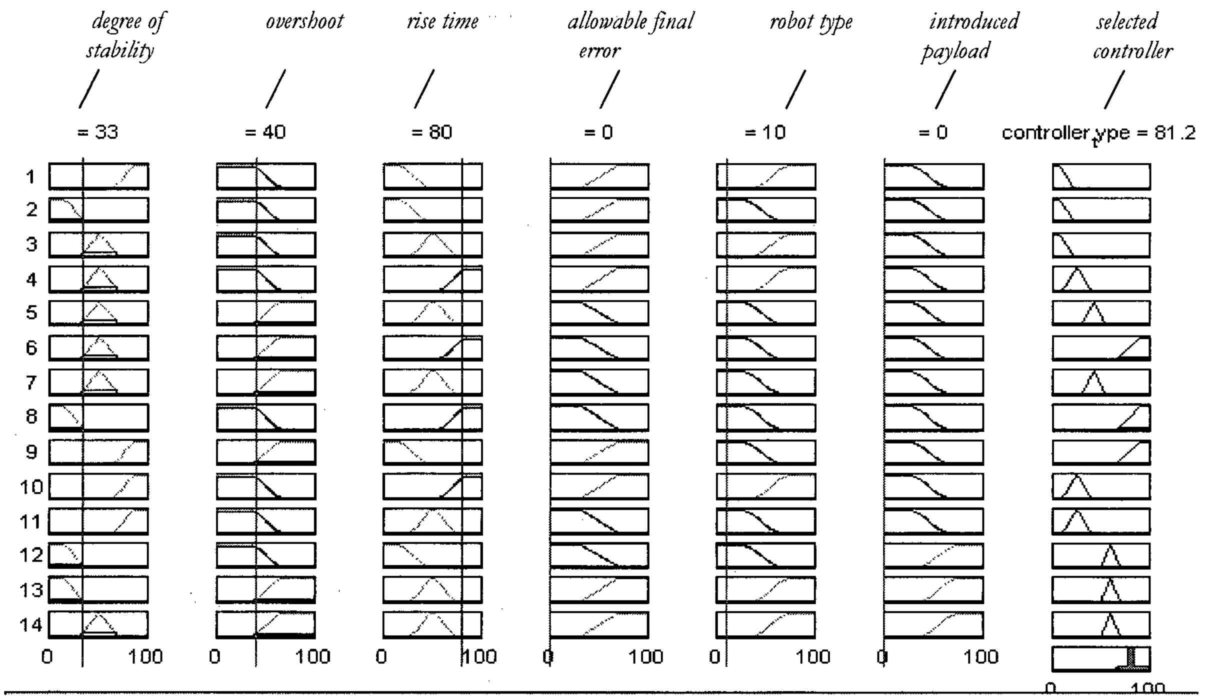


Figure 6.4. The CM Matlab editor.

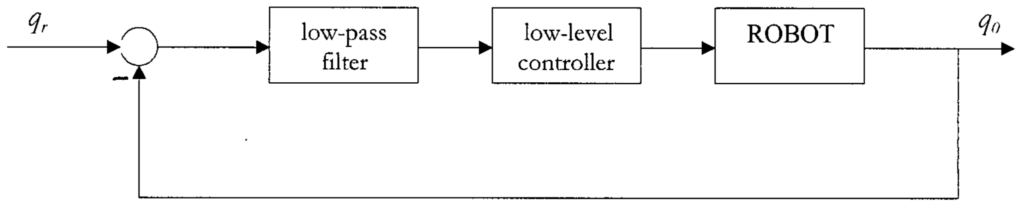


Figure 6.5. Independent joint controller with PID controller.

Thus, the digital PID control signal is calculated as

$$\begin{aligned} u[k] &= b[k]e[k] + b[k-1]e[k-1] + b[k-2]e[k-2] \\ &\quad - (a[k]u[k-1] + a[k-1]u[k-2]). \end{aligned} \tag{6.1}$$

The complete control law is derived in Appendix D.

6.2.5 Discussion and Comparison of Results

The Figures 6.6 and 6.7 represents the time history of the second axis position error and applied motor current with independent control architecture utilized with different types of controllers. The gains of chosen controllers are tuned manually.

Comparison of results shows that independent joint control architectures with PD and P controller gives rather sluggish behavior especially for first tree links, and non of these schemes can achieve zero state error. However, the selected PID controller after gain tuning fulfills required criteria.

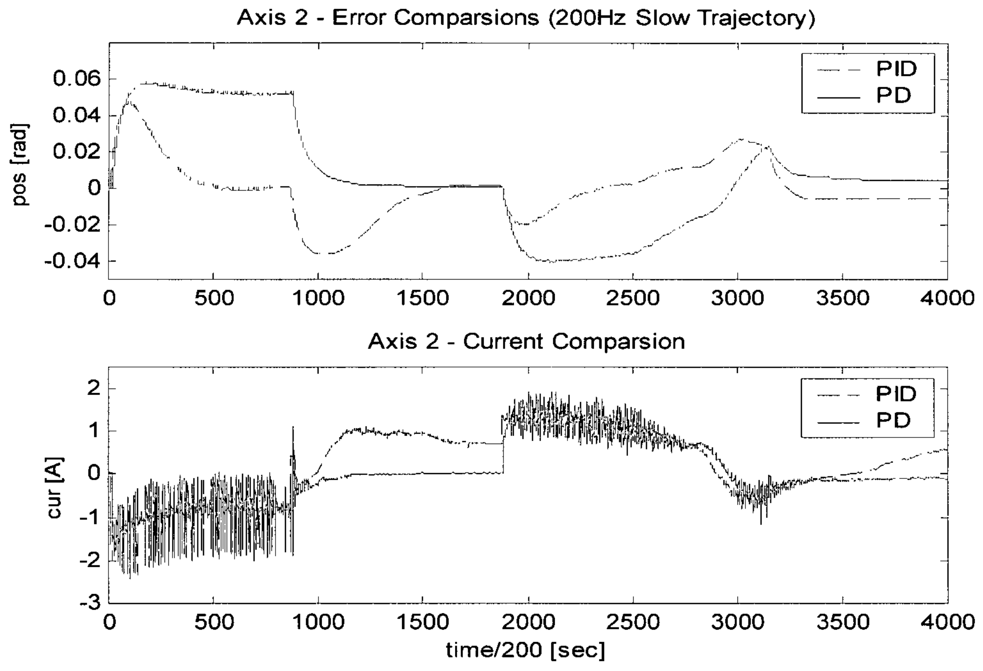


Figure 6.6. Time response of second-axis for PTP task with PID and PD controllers.

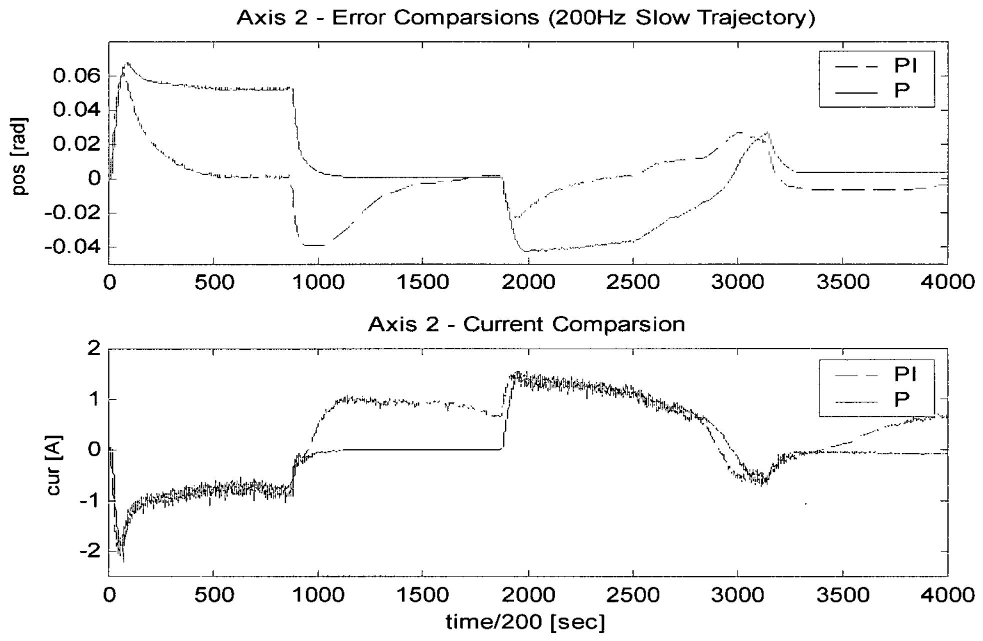


Figure 6.7. Time response of second-axis for PTP task with PI, and P controllers.

6.3 Point-to-Point Motion Task with the Payload

The point-to-point motion task with the payload concentrated at the tip of end-effector will be considered next. In this process, the payload is considered as internal disturbance that has to be compensated. The particular task is detailed below.

6.3.1 Task Description

Available information and criteria for performing this task are obtained from user and shown in Table 6.3 and Table 6.4. All values are scaled from 0-100.

Module	**Information obtained from user**		**Membership**
`motion_type`	- Type of motion, T=5%, - Following of desired/specific path (tracking), F=10%, - Control type during the contact (if exists), C=0,	DMC=15	*ptp*
`model_accuracy`	- Robot model type, RT=23, - Robot model confidence, RC=87%,	A=75%	*hi*
`position_sensor`	- Information uncertainty, IU=18%, - Signal filtered, SF=15%,	VP=86%	*hi*
`velocity_sensor`	- Information uncertainty, U=23%, - Signal filtered, SF=15%,	VV=78%	*hi*
`hardware`	- Computation rate, CR=78, - Transfer rate, TR=95, - Sampling rate, SI=49,	AC=80%	*me*
`disturbances`	- External disturbances, ED=10%, - Internal disturbances, ID=85%,	D=50%	*me*

Table 6.3. Available information for control architecture selection.

Module	Criteria	Membership
`control_structure`	- trajectory rate, ST=12 - driving system of the joints, DS=20 - response, RE=80 - control accuracy, AC=50	**CS=29** *0-70 decentralized* *50-100 centralized*
`positioning_ controller_type`	- degree of stability, β=60 - overshoot, Π=20 - rise time, Δ= 75 - allowable error, E= 40 - robot type, RT=10 - introduced payload, P=90 -	**CT=60** *0-22 P controller* *15-40 PD controller* *30-54 PI controller* *50-70 PD+G controller* *65-100 PID controller*

Table 6.4. Criteria for controller selection.

6.3.2 Point-to-Point Trajectory

Using the MATLAB Robotics Toolbox [66], a trajectory for a point-to-point (PTP) motion in operational space has been defined with as:

```
PTP_traj( [0.4; -0.2; 0.1], [0.4; 0.4; 0.1], 2, 0.005, 60, 5 )
```

6.3.3 Expert System Decision

According to the information obtained by user and the task description, the expert system is required to decide which control architecture is most suitable for a particular task. The task is a simple point-to-point motion, with low-speed trajectory and the response is required to be fast with small error. Overall disturbances are at medium level that requires some type of compensation. The expert system selected a decentralized /independent joint control/ with PD + gravity compensation controller (see Figures 6.8 and 6.9).

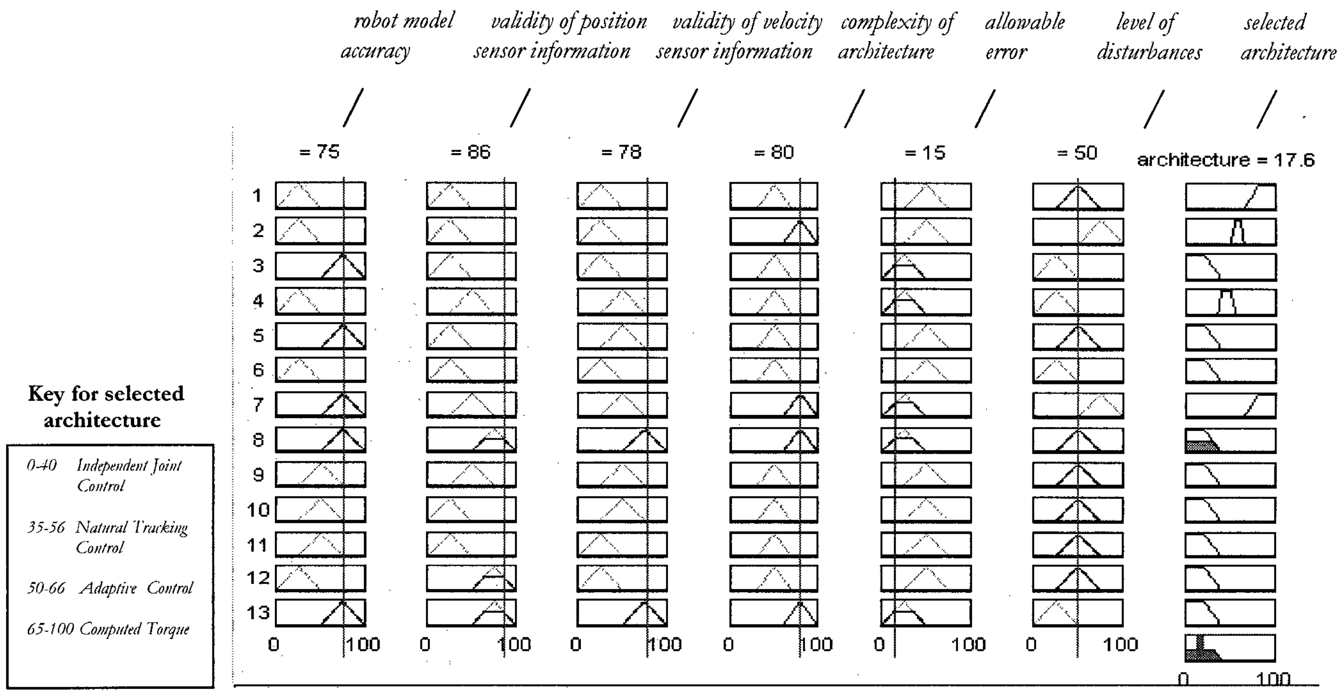

robot model accuracy = 75
validity of position sensor information = 86
validity of velocity sensor information = 78
complexity of architecture = 80
allowable error = 15
level of disturbances = 50
selected architecture
architecture = 17.6
Key for selected architecture
0-40 Independent Joint Control
35-56 Natural Tracking Control
50-66 Adaptive Control
65-100 Computed Torque

Figure 6.8. The CAM Matlab editor for free robot motion.

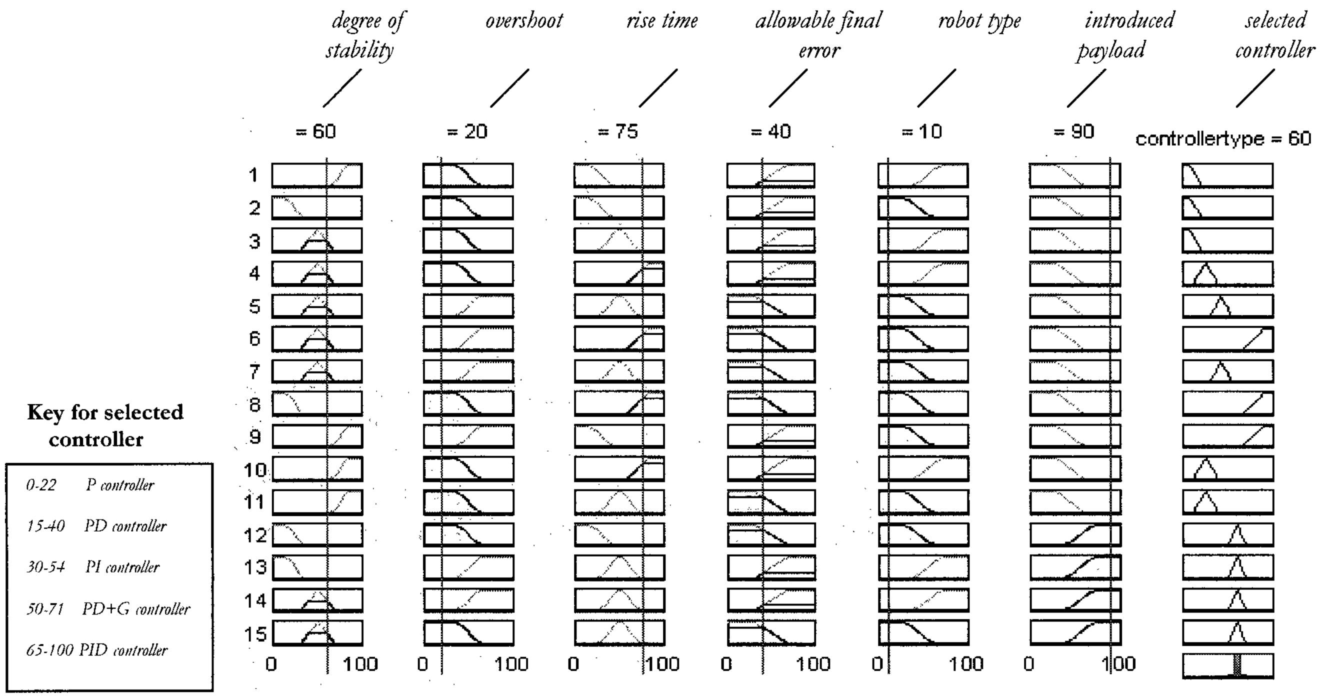


Figure 6.9. The CM Matlab editor.

6.3.4 Applied Control Law

The block diagram of utilized architecture is presented on Figure 6.10. The zero-order hold is embedded into the architecture of ORTS. The low-pass filter is designed as second order system, and the applied control law is presented below.

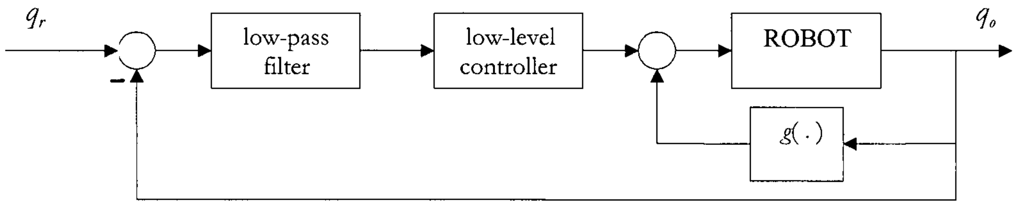


Figure 6.10. Independent joint controller with PD+G controller.

The gravitational terms already divided by the motor parameters for the first three links are:

g[0] = 0, (6.2)

g[1] = (-3.2373cos(q[1]) – 5.15cos(q[1] + q[2])) / 1.7677, (6.3)

g[2] = (-5.15cos(q[1] + q[2])) /6.6269, (6.4)

where joint angles are:

$$\theta_1 = q[0], \qquad \theta_2 = q[1], \qquad \theta_3 = q[2]. \tag{6.5}$$

Thus, the digital PD + G control signals for the first three axis are calculated as

$$u[0] = g[0] + pd[0], \tag{6.6}$$

$$u[1] = g[1] + pd[1], \tag{6.7}$$

$$u[2] = g[2] + pd[2]. \tag{6.8}$$

The complete control law is derived in Appendix D.

6.3.5 Discussion and Comparison of Results

The Figure 6.11 represents the time history of the second axis position error and applied motor current with PID and PD+G controllers. Comparison of results shows that independent joint control architectures with PID controller is unstable with initially imposed gains. Thus, PD + G controller after gain tuning fulfills required criteria. The gains of chosen controllers are tuned manually.

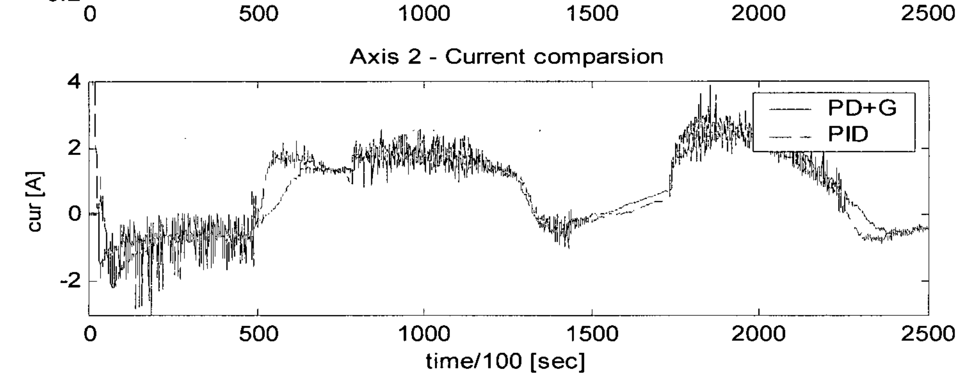


Figure 6.11. Time response of second-axis for PTP task with payload and PID and PD+G controller.

6.4 Point-to-Point Motion Task with High-Speed Trajectory

The point-to-point motion task with a high-speed trajectory will be considered next. In this process, due to high-speed trajectory the whole robot dynamics should be calculated. The particular task is detailed below.

6.4.1 Task Description

Available information and criteria for performing this task are obtained from user and shown in Table 6.5 and Table 6.6. All values are scaled from 0-100.

Module	**Information obtained from user**		**Membership**
`motion_type`	- Type of motion, T=5%, - Following of desired/specific path (tracking), F=10%, - Control type during the contact (if exists), C=0,	DMC=15	*ptp*
`model_accuracy`	- Robot model type, RT=23, - Robot model confidence, RC=25%,	A=29%	*lo*
`position_sensor`	- Information uncertainty, IU=18%, - Signal filtered, SF=15%,	VP=86%	*hi*
`hardware`	- Computation rate, CR=68, - Transfer rate, TR=75, - Sampling rate, SI=49,	AC=53%	*me*
`disturbances`	- External disturbances, ED=0%, - Internal disturbances, ID=15%,	D=20%	*lo*
`velocity_sensor`	- Information uncertainty, IU=18%, - Signal filtered, SF=70%,	VP=56%	*me*

Table 6.5. Available information for control architecture selection.

Module	Criteria	Membership
`control_structure`	- trajectory rate, ST=80 - driving system of the joints, DS=20 - response, RE=80 - control accuracy, AC=50	**CS=32** *0-70 decentralized* *50-100 centralized*
`positioning_ controller_type`	- degree of stability, β=33 - overshoot, Π=40 - rise time, Δ= 80 - allowable error, E= 0 - robot type, RT=10 - introduced payload, P=10	**CT=81** *0-22 P controller* *15-40 PD controller* *30-54 PI controller* *50-70 PD+G controller* *65-100 PID controller*

Table 6.6. Criteria for controller selection.

6.4.2 Point-to-Point Trajectory

Using the MATLAB Robotics Toolbox [66], a trajectory for a point-to-point (PTP) motion in operational space has been defined with as:

```
PTP_traj( [0.4; -0.2; 0.1], [0.4; 0.4; 0.1], 1, 0.005, 60, 5 )
```

6.4.3 Expert System Decision

According to the information obtained by user and the task description, the expert system is required to decide which control architecture is most suitable for a particular task. The task is a simple point-to-point motion, with high-speed trajectory and the response is required to be fast with small error. Overall disturbance is at low level, and model accuracy is at low level. The expert system selected decentralized /independent joint control/ with PID controller scheme with the highest membership. Usually, for high-speed trajectory and precise applications, the whole manipulator dynamic should be calculated with commonly utilized computer torque architecture. Due to low level in model accuracy this architecture is second choice in expert system decision (see Figures 6.12 and 6.13).

Key for selected architecture

0-40	*Independent Joint Control*
35-57	*Natural Tracking Control*
50-67	*Adaptive Control*
65-100	*Computed Torque*

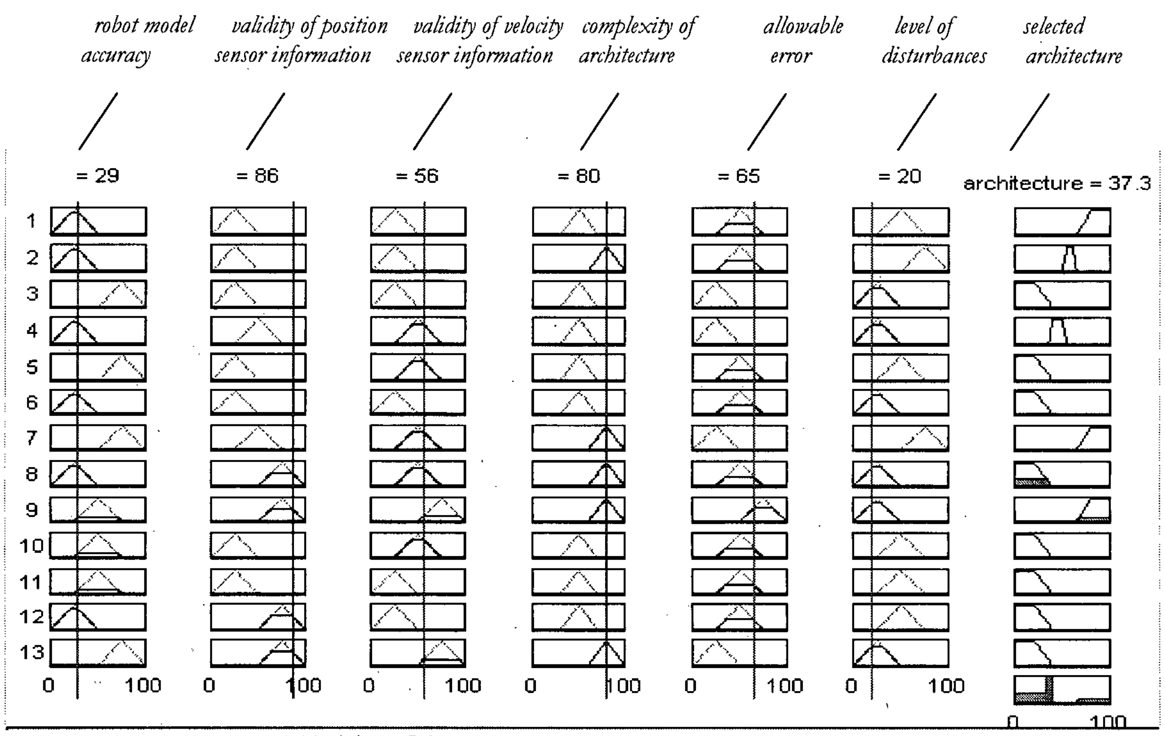


Figure 6.12. The CAM Matlab editor for free robot motion.

Key for selected controller

0-22	*P controller*
15-40	*PD controller*
30-54	*PI controller*
50-72	*PD+G controller*
65-100	*PID controller*

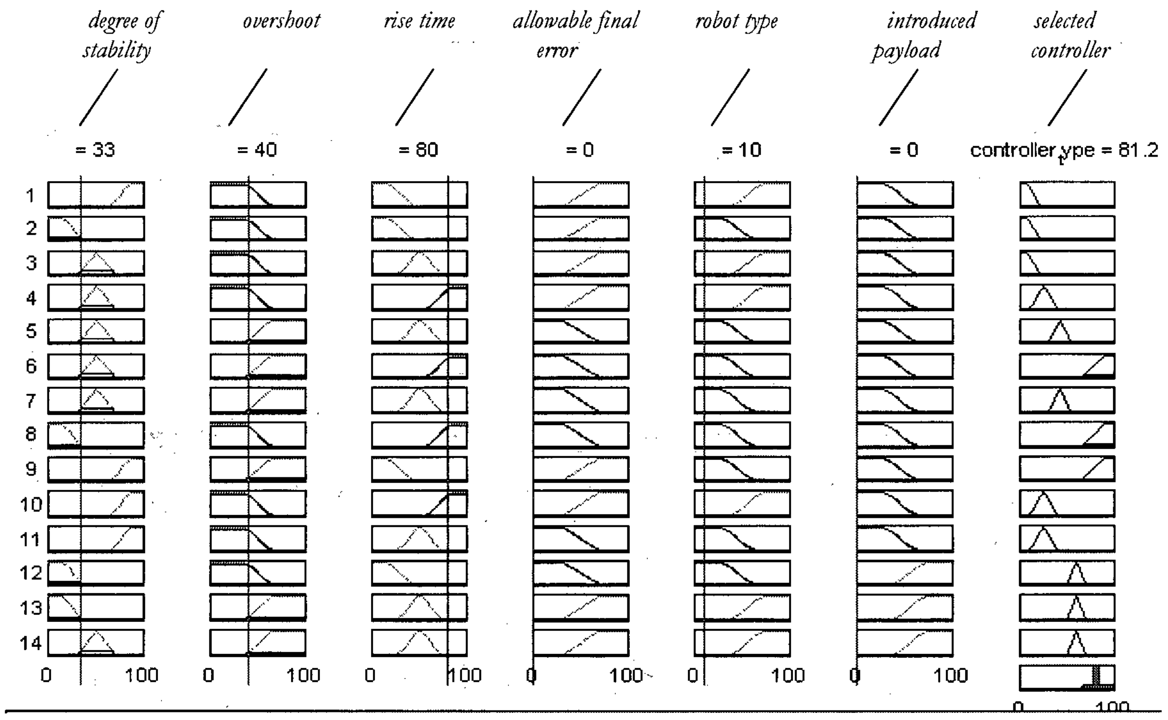


Figure 6.13. The CM Matlab editor.

6.4.4 Applied Control Law

Two different control laws for this task are utilized. The first one is independent joint control architecture with PID controller that is explained in previous section, and the computer torque control architecture with PID controller.

The block diagram computer torque architecture utilized for this task is presented on Figure 6.14. The zero-order hold is embedded into the architecture of ORTS. Due to the lack of velocity sensor information the signal from encoder is sent through the designed first order filter and the approximation of velocity signal is made. The low-pass filter is designed as second order system, and the applied control law is presented below.

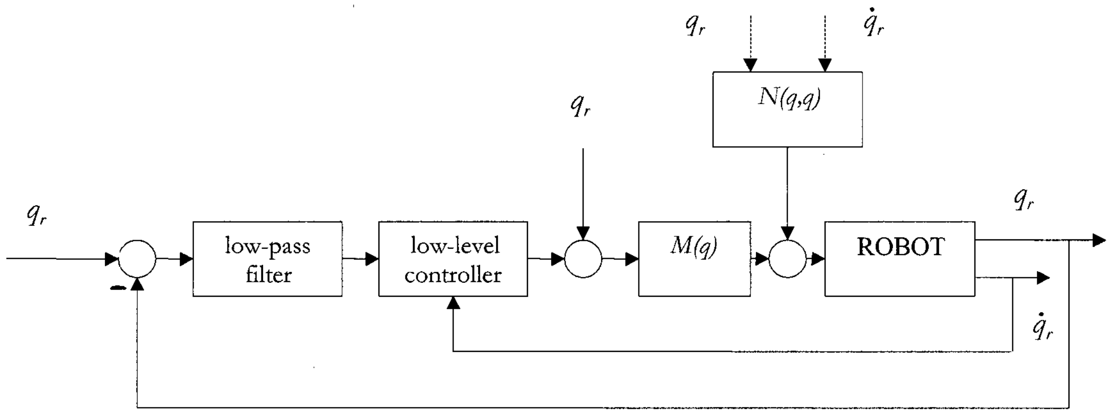


Figure 6.14. Computer torque with PID controller.

For this task only first three links of manipulator are considered and controlled. The control signals for first three links are given, respectively:

$$u[0] = n[0] + 0.5657(m[0][0]\ddot{q}_d[0] + m[0][1]\ddot{q}_d[1] + m[0][2]\ddot{q}_d[2]) + m[0][0]pid[0] + m[0][1]pid[1] + 3.7488m[0][1]pid[2], \quad (6.9)$$

$$u[1] = n[1] + 0.5657(m[1][0]\ddot{q}_d[0] + m[1][1]\ddot{q}_d[1] + m[1][2]\ddot{q}_d[2]) + m[1][0]pid[0] + m[1][1]pid[1] + 3.7488m[1][1]pid[2], \quad (6.10)$$

$$u[2] = n[2] + 0.5657(m[2][0]\ddot{q}_d[0] + m[2][1]\ddot{q}_d[1] + m[2][2]\ddot{q}_d[2]) + m[2][0]pid[0] + m[2][1]pid[1] + 3.7488m[2][1]pid[2]. \quad (6.11)$$

The control law in complete form is derived in Appendix D.

6.4.5 Discussion and Comparison of Results

The Figure 6.15 represents the time history of the second axis position error and applied motor current with two architectures utilized; independent joint control with PID controller and computer torque with the same low-level controller.

Comparison of results shows that independent joint control architectures with PID controller after the gains are tuned gives slightly better results than implemented computer torque architecture with same type of low-level controller. The gains of chosen controllers are tuned manually.

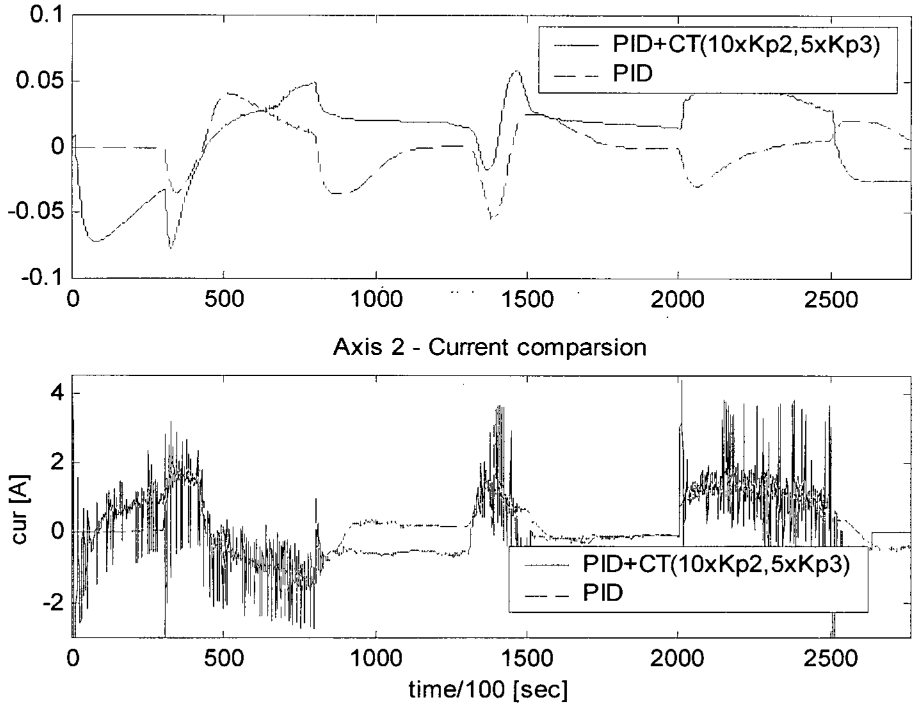


Figure 6.15. Time response of second-axis for PTP task with PID and CT controllers.

Chapter VII

Conclusions and Recommendations

7.1 Summary

In this thesis, a broad literature review on main issue of robotics control, and its sub-issues regarding manipulator structure, environment, tasks, and communications, was performed. The key issues or characteristics which drive selection of robot control architectures and robot controllers are identified as: dynamic motion characteristics, robot and environment model accuracy, effective stiffness, validity of sensor information, hardware capabilities, internal and external disturbances, etc. As well, a comparisons of different robot control architectures and robot controllers based on the issues of robustness, response, reliability and complexity were performed. Based on the review and comparison a taxonomy for robot control, which classifies different robot industrial tasks, manipulator characteristic motions, control architectures and controllers was developed. Using this taxonomy and identified characteristics commonly encountered in industrial robotic tasks, the rules and criteria for selection of robot control architectures and controllers within an expert system database were developed.

The fuzzy rulebase expert system was developed for the selection of control architecture and associated controller for specific robot task taking into account the information about the manipulator and environment stiffness, disturbances effect, models accuracy, etc., as well as criteria about the robot performance set by the user. The expert system processes the user information through the twelve different expert system's modules, and the final decision is made based on fuzzyfication of these twelve outputs and final defuzzification where a crisp value represents a certain control architecture/controller pair.

Simulations and experiments are done both for free and constrained robot motion to verify both the taxonomy and expert system based on this taxonomy. Comparison of results of chosen and other control scheme applied shows the importance of proper selection of correct control algorithm.

7.2 Recommendation

It is of interest to indicate some of possible future investigation subjects. A clear formulation and specification of dynamic control of both unconstrained and contact tasks are further required. Further simulation and experimental tests of recently proposed compliant motion control algorithms, such as parallel position/force control, adaptive and variable structure algorithms, and, particularly, dynamic control of robot interacting with a dynamic environment are also of interest.

In the impedance control, further advances are to be expected in adaptation of target impedances to complex tasks requirements. The robust control continues to be in the focus of control design.

Further comparison of available control architectures, definition of benchmark tests, investigation of compliant control in uncertain and dynamic environment, examination of non-linear effects in robot and environment dynamics, solving control problems at higher control levels, etc., are certainly some areas deserving further theoretical and experimental studies.

Due to the lack of functions developed for the Fuzzy Logic Toolbox used in the project, the expert system environment is restricted to work off-line. Thus, some future work into on-line implementation is important, specifically, for on-line tuning. The proposed tuning system is described in Appendix E.

Given the large number of rules implemented in this system, some work on combining/reducing this rulebase would be advantageous to speed up the system decision making.

Bibliography

[1] Whitney, D. "Historical perspective and state of the art in robot force control", *The Int. J. of Robotic Research,* V.6, No.1, 1987.

[2] Colbaugh, R., Seraji, H. "Force tracking in impedance control", *ICRA '93*, p.499-506.

[3] Wen, T. "A unified perspective on robot control: The energy Lyapunov function approach", *Proc. 29th Conference on Decision and Control,* 1990, p.1968-1973.

[4] Khosla, P. "Choosing sampling rates for robot control", *ISA'87*, p. 121-127.

[5] Hogan, N. " On the stability of manipulators performing contact tasks", *IEEE J. of Rob. and Auto.,* V.4., No 6, 1988.

[6] Mason, M. T. "Compliance and force control for computer controlled manipulators", *IEEE Trans. On Systems, Man, and Cybernetics*, SMC-11(6), 1981, p. 418-432.

[7] Hogan, N. "Impedance control: An approach to manipulation, Part 1 – Theory, Part 2 – Implementation, Part 3 – Application, *Journal of Dynamic systems, Measurement and Control,* 1985, p. 1-24.

[8] Chande, A. M.; Newcomb, R. W., "A decision tree for inflight data processing for robot spacecraft trajectory guidance", *IEEE Int. Conf.on Robotics and Automation,* 1985, p. 215-220.

[9] Foulloy, L.; Burg, B.; LaMotte, E., "A rule-based decision system for the robotization of metal laser cuting", *IEEE Int. Conf. On Robotics and Automation,* 1985, p. 192-197.

[10] Anderson, R. L., "A robot ping-pong player – experiment in real-time intelligent control", *MIT Press*, Cambridge, MA, 1988.

[11] Slotine, J.J.E.; Yang, H.S., "Fast algorithms for near-minimum time control of robot manipulators", *The Int. J. of Robotics Research,* Vol. 13, No. 6, 1994, p. 521-532.

[12] McClamroch, N. H., Wang, D., "Feedback stabilization and tracking in constrained robots", *IEEE Trans. on Automatic Control,* 33(5), 1988, p. 419-426.

[13] Vukobratovic, M., "How to control robots interacting with dynamic environment", *J. of Intelligent and Robotic Systems,* 19, 1997, p. 119-152.

[14] Stokic, D., Vukobratovic, M., "Contribution to practical stability analysis of robot interacting with dynamic environment", *Proc. Of the first ECPD Int. Conf. On Advances Robotics and Intelligent Automation,* 1995, p. 693-699.

[15] Hollerbach, J., An, C., "The role of dynamic models in Cartesian force control of manipulators", *Int. J. of Robotics Research,* 8(4), 1989, p. 51-72.

[16] Altintas, Y.; Erol, N. A.; Ito, M., "Modular tools for motion control and process control system design", *Open Architecture Control Systems ITIA series,* Vol. 2, 1998, p. 183-198.

[17] Vukobratovic, M.; Karan, B., "Experiments with fuzzy logic robot control with model-based dynamic compenstaion in nonadaptive decentralized control scheme", *Int. J. of Robotics and Automation,* Vol. 11, No. 3, 118-131, 1996.

[18] Stokic, D; Vukobratovic, M., "Hystorical perspectives and state of the art in joint force sensory feedback control of manipulation robots", *Robotica,* 11, 149-157, 1993.

[19] An, C.H.; Hollerbach, J., "Dynamic stability issues in force control of manipulators ", *Proc. IEEE Intern. Conf. On Robotics and Automation,* Raleigh, 1987.

[20] Latornell, D. J.; Cherchas, D. B.; Wong, R., "Dynamic Characteristics of Constrained Manipulators for Contact Force Control Design", *Int. J. of Robotics Research,* Vol. 17, No. 3, 211-231, 1998.

[21] Constantinescu, D., "Smooth Time Optimal Trajectory Planning for Industrial Manipulators", *M.A.Sc. Thesis*, 1998.

[22] O'Dor, M., "Identification of Salmon Can-Filling Defects", *M.A.Sc. Thesis*, 1998.

[23] Vukobratovic, M.; Stokic, D., "Upravljanje Manipulacionim Robotima – Analiza – Sinteza – Vezbe", *Tehnicka Knjiga*, Beograd, YU, 1990.

[24] de Silva, C. W., "Intelligent Control of Robotics Systems with Application in Industrial processes", *Robotics and Automation Systems*, 21, 221-237, 1997.

[25] Croft, E. A.; Benhabib, B.; Fenton, R. G., "Near-Time Optimal Robot Motion Planning for On-line Applications", *Journal of Robotics Systems*, 12 (8), 553-567, 1995.

[26] Bobrow, J. E.; Dubowsky, S.; Gibson, J. S., "Time-Optimal Control of Robotics Manipulator Along Specified Paths", *The Int. Journal of Robotics Research*, 4 (3), 3-17, 1985.

[27] Luh, J. Y. S; Lin, C. S., "Approximate Joint Trajectories for Control of Industrial Robots along Cartesian Paths", *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-14 (3), 444-450, 1984.

[28] Hoffman, M.; Malowany, A; Angeles, J., "Near-Minimum-Time Trajectories for Pick-and-Place Operations", *ASME Int. Conference on Computers in Engineering*, 433-438, San Francisco, California, 1988.

[29] Jang, R.; Gulley, N., "Fuzzy Logic Toolbox User's Guide", *Springler – Verlag*, New York, 1995.

[30] The Math-Works Inc., "MATLAB – High-Performance Numeric Computation and Visualization Software", *Springler – Verlag*, 1992.

[31] The Math-Works Inc., "SIMULINK – Simulation Software", *Springler – Verlag*, 1992.

[32] Siciliano, B.; Sciavicco, L., "Modeling and Control of Robot Manipulators", *The McGraw-Hill*, 1996.

[33] Lewis, F. L.; Abdallah, C. T.; Dawison, D. M., "Control of Robot Manipulators", *Macmillan*, 1993.

[34] De Luca, A.; Manes, C., "On the Modeling of Robots in Contact with Dynamic Environment", *Proc. 5th International Conference on Advanced Robotics*, Pisa, 1991.

[35] Goldenberg, A. A., "Analysis of Force Control Based on Linear Models", *Proc. IEEE Intern.Conf. on Robotics and Automation*, Nice, 1992.

[36] Surdilovic, D., "Contact Stability Issues in Position Based Impedance Control: Theory and Experiments", *Proc. IEEE Intern. Conf. on Robotics and Automation*, Minneapolis, 1996.

[37] Duffy, J., "The Fallacy of Modern Hybrid Control Theory that is Based on 'Orthogonal complements' of twist and wrench spaces", *Journal of Robotics Systens*, 7(2), 139-144, 1990.

[38] Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation", *IEEE J. on Robotics and Automation*, vol. 3, no. 1, 1987.

[39] De Luca, A.; Manes, C.; Nicolo, F., " A Task Space Decoupled Approach to Hybrid Control of Manipulators", *Proc. IFAC Symp. on Robot Control*, Karlsruhe, 1988.

[40] Raibert, M. H.; Craig, J. J., "Hybrid Position/Force Control of Manipulators" *ASME J. Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, 1981.

[41] Lipkin, H.; Duffy, J., "Hybrid Twist and Wrench Control for a Robotic Manipulator", *ASME J. Mechanisms, Transmission, and Automation in Design*, vol. 110, no. 2., 1988.

[42] Zhang, H.; Paul, R., "Hybrid Control of Robot Manipulator", *Proc. IEEE Intern. Conf. on Robotics*, 1985.

[43] Nicoladi, S.; Surdilovic, D.; Schott, J., " Development of a Space Controller with Advanced Sensory-Based Control Capabilities", *Proc. 7th Intern. Conference on Advanced Robotics*, Sant Feliu de Guixols, Spain, 1995.

[44] Whitney, D. E., "Force Feedback Control of Manipulator Fine Motions", *ASME J. Dynamic Systems, Measurement, and Control*, 1980.

[45] Salisbury, J. K. "Active Stiffness Control of a Manipulator in Cartesian Coordinates", *Proc. 19th IEEE Conference on Decision and Control*, 1980.

[46] Maples, J. A.; Becker, J. J., "Experiments in Force Control of Robotics Manipulators", *Proc. IEEE Int. Conf. on Robotics and Automation*, 1986.

[47] Paul, R. P.; Shimano, B., "Compliance and Control", *Proc. Joint Automatic Control Conf.*, San Francisco, 1976.

[48] Luh, J. Y. S.; Fisher, W. D.; Paul, R.P.C., "Joint Torque Control by a Direct Feedback for Industrial Robots", *IEEE Transaction on Automatic Control*, vol. 28. no. 2, 1983.

[49] Shin, K. G.; Lee, C. P., "Compliant Control of Robotic Manipulators with Resolved Acceleration", *Proc. 24th IEEE Conf. Decision and Control*, 1985.

[50] Sciavicco, L.; Chiaverini, S., "Force/Position Control of Manipulators in Task Space with Dominance in Force", *Proc. 2nd IFAC Symposium on Robot Control*, Karlsruhe, 1988.

[51] Chiaverini, S.; Siciliano, B.; Villani, L., "Force/Position Regulation of Compliant Robot Manipulators", *IEEE Transaction on Automatic Control*, vol.39., no. 3, 1994.

[52] Vukobratovic, M.; Ekalo, Y., "Unified Approach to Control Laws Synthesis for Robotic Manipulators in Contact with Dynamic Environment", *Proc. IEEE Intern. Conf. on Robotics and Automation*, 1993.

[53] Vukobratovic, M.; Ekalo, Y., "New Approach to Control of Robotics Manipulators Interacting with Dynamic Environment", *Robotica*, vol. 14, no.1, 1996.

[54] Vukobratovic, M.; Surdilovic, D. T., "Control of Robotic Systems in Contact Tasks: An Overview", *Izvesti Akademii Nauk. Teorij i Sistemi Upravljanja*, no. 5, 173-192, 1997.

[55] Erol, N. A.; Altintas, Y., "Open Architecture Modular Tool Kit for Motion and Process Control", *ASME International Mechanical Engineering Congress and Exposition*, ASME Publication MED, 15-22, Dallas, Texas, 1997.

[56] Greenfeld, I.; Hansen, F.; Fehlinger, J.; Pavlakos, E., "MOSAIC System Description, Specification and Planning", *Technical Report No. 452, Robotics Report No. 201*, New York University, 1989.

[57] Anderson, B. M.; Cole, J. R.; Holland, R. G., "An Open Standard for Industrial Controllers", *Manufacturing Review*, vol. 6, no. 3, 181-191, 1993.

[58] Yellowley, I.; Pottier, P. R., "The Integration of Process and Geometry within an Open Architecture Machine Tool Controller", *International Journal of Machine Tools and Manufacture*, vol. 34, no. 2, 277-293, 1994.

[59] Lo, C.; Koren, Y., "Evaluation of Servo-Controllers for Machine Tools", *Proceedings of the 1992 American Control Conference*, vol. 1, 370-374, 1992.

[60] Pasek, Z.; Ulsoy, G.; Koren, Y.; Birla, S.; Shin, K. G., " An Open Architecture Based Real Time Controller for Machining Processes", *Proc. of the 1996 NSF Design and Manufacturing Grantees Conf.*, 273-274, 1996.

[61] Teltz, R.; Urbasik, K.; Shawky, A.; Elbestawi, M. A., "Sensor Based Planning and Control for Open Architecture Manufacturing", *Technical Papers of the North American Manufacturing Research Institution of SME*, 199-204, 1995.

[62] Lundholm, T., "A Flexible Real Time Adaptive Control System for Turning", *Anals of the CIRP*, vol. 40, 441-444.

[63] Tung, E. D.; Tomizuka, A. M., "Feedforward Tracking Control Design Based on the Identification of Low Frequency Dynamics", *ASME Journal of Dyn. Sys. Measurement and Control*, vol. 115, no. 4, 189-196.

[64] Minbashian, B.; Warwick, K., "Decision Rules for Intelligent Parallel Controllers", *University Press*, Department of Cybernetics, University of Reading, 1989.

[65] The Math-Works Inc., "Fuzzy Logic Toolbox", *Springler – Verlag*, 1992.

[66] The Math-Works Inc., "Robotics Toolbox", *Springler – Verlag*, 1994.

Appendix A

Robot Dynamic Models

For simulation and experimental result purposes, which are explained in the Chapter V and Chapter VI, the two different robot dynamic model are used and there are derived and explained in the following sections.

A.1 Two-Link Planar Arm

In the simulation presented in Chapter V, the two-link planar arm is used. Two degrees of mobility, in fact, are enough to understand the physical meaning of all dynamic terms, especially the joint coupling terms [32]. The parameters of two-link manipulator used for all necessary calculations are given in the Table A.1.

Parameter	Symbol	Value
generalized joint coordinates	q = (q_1, q_2)	variable
functional S-coordinates	**s** = (x, y)	variable
mass of the first link	m_{l1}	50 kg
mass of the second link	m_{l2}	50 kg
moment of inertia with respect to the axis of first rotor	I_{m1}	10 kg m^2
moment of inertia with respect to the axis of second rotor	I_{m2}	10 kg m^2
moment of inertia relative to the center of mass of the first link	I_{l1}	0.01 kg m^2
moment of inertia relative to the center of mass of the second link	I_{l2}	0.01 kg m^2
mass of the first rotor	m_{m1}	5 kg
mass of the second rotor	m_{m2}	5 kg
first link length	a_1	1 m
second link length	a_1	1 m
distance of the center of mass of first link from the first joint axis	l_1	0.5 m
distance of the center of mass of second link from second joint axis	l_2	0.5 m
viscous friction coefficient at the motor shafts	$F_{m1} = F_{m2}$	0.01 N m s/rad
actuators' armature resistance	$R_{a1} = R_{a1}$	10 ohm
actuators' torque constant	$k_{t1} = k_{t2}$	2 N m/A
actuators' voltage constant	$k_{v1} = k_{v2}$	2 V s/rad.

Table A.1. The parameters of two-link planar arm.

It was shown that the equation of motion in the absence of external end-effector forces and, for simplicity, of static friction is described by

$$B(q)\ddot{q}+C(q,\dot{q})\dot{q}+F_v\dot{q}+g(q)=\tau. \tag{A - 1}$$

The inertia matrix is:

$$B(q)=\begin{bmatrix} b_{11}(\vartheta_2) & b_{12}(\vartheta_2) \\ b_{21}(\vartheta_2) & b_{22} \end{bmatrix}, \tag{A - 2}$$

$$\begin{aligned} b_{11} &= I_{l_1}+m_{l_1}l_1^2+k_{r1}^2 I_{m_1}+I_{l_2}+m_{l_2}(a_1^2+l_2^2+2a_1l_2c_2) \\ &\quad +I_{m_2}+m_{m_2}a_1^2, \end{aligned} \tag{A - 3}$$

$$b_{12}=b_{21}=I_{l_2}+m_{l_2}(l_2^2+a_1l_2c_2)+k_{r2}I_{m_2}, \tag{A - 4}$$

$$\mathrm{b}_{22}=I_{l_2}+m_{l_2}l_2^2+k_{r2}^2 I_{m_2}. \tag{A - 5}$$

The Christoffels symbols are computed from inertia matrix, according to:

$$c_{ijk}=\frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k}+\frac{\partial b_{ik}}{\partial q_j}-\frac{\partial b_{jk}}{\partial q_i}\right), \tag{A - 6}$$

$$c_{111}=\frac{1}{2}\frac{\partial b_{11}}{\partial q_1}=0, \tag{A - 7}$$

$$c_{112}=c_{121}=\frac{1}{2}\frac{\partial b_{11}}{\partial q_2}=-m_{l_2}a_1l_2s_2=h, \tag{A - 8}$$

$$c_{122}=\frac{\partial b_{12}}{\partial q_2}-\frac{1}{2}\frac{\partial b_{22}}{\partial q_1}=h, \tag{A - 9}$$

$$c_{211} = \frac{\partial b_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial b_{11}}{\partial q_2} = -h, \tag{A - 10}$$

$$c_{212} = c_{221} = \frac{1}{2}\frac{\partial b_{22}}{\partial q_1} = 0, \tag{A - 11}$$

$$c_{222} = \frac{1}{2}\frac{\partial b_{22}}{\partial q_2} = 0, \tag{A - 12}$$

leading to the matrix

$$C(q,\dot{q}) = \begin{bmatrix} h\dot{\vartheta}_2 & h(\dot{\vartheta}_1 + \dot{\vartheta}_2) \\ -h\dot{\vartheta}_1 & 0 \end{bmatrix}. \tag{A - 13}$$

Computing the skew - symetric matrix N leads to

$$N(q,\dot{q}) = \dot{B}(q) - 2C(q,\dot{q}), \tag{A - 14}$$

$$= \begin{bmatrix} 2h\dot{\vartheta}_2 & h\dot{\vartheta}_2 \\ h\dot{\vartheta}_2 & 0 \end{bmatrix} - 2\begin{bmatrix} h\dot{\vartheta}_2 & h(\dot{\vartheta}_1 + \dot{\vartheta}_2) \\ -h\dot{\vartheta}_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -2h\dot{\vartheta}_1 - h\dot{\vartheta}_2 \\ 2h\dot{\vartheta}_1 + h\dot{\vartheta}_2 & 0 \end{bmatrix}. \tag{A - 15}$$

Non-zero gravitational terms are calculated as follows:

$$g_1 = (m_{l_1} l_1 + m_{m_2} a_1 + m_{l_2} a_1) g c_1 + m_{l_2} l_2 g c_{12}, \tag{A - 16}$$

$$g_2 = m_{l_2} l_2 g c_{12}. \tag{A - 17}$$

In the absence of friction and tip contact forces, the resulting equations of motion are

$$(I_{l_1} + m_{l_1} l_1^2 + k_{r1}^2 I_{m_1} + I_{l_2} + m_{l_2}(a_1^2 + l_2^2 + 2a_1 l_2 c_2) + I_{m_2} + m_{m2} a_1^2)\ddot{\vartheta}_1$$
$$+ (I_{l_2} + m_{l_2}(l_2^2 + a_1 l_2 c_2) + k_{r1} I_{m_2})\ddot{\vartheta}_2 - 2m_{l_2} a_1 l_2 s_2 \dot{\vartheta}_1 \dot{\vartheta}_2$$
$$- m_{l_2} a_1 l_2 s_2 \dot{\vartheta}_2^2 + (m_{l_1} l_1 + m_{m_2} a_1 + m_{l_2} a_1) g c_1 + m_{l_2} l_2 g c_{12} = \tau_1, \qquad \text{(A - 18)}$$

$$(I_{l_2} + m_{l_2}(l_2^2 + a_1 l_2 c_2) + k_{r2} I_{m_2})\ddot{\vartheta}_1 + (I_{l2}^2 + m_{l_2} l_2^2 + k_{r2}^2 I_{m_2})\ddot{\vartheta}_2$$
$$+ m_{l_2} a_1 l_2 s_2 \dot{\vartheta}_1^2 + m_{l_2} l_2 g c_{12} = \tau_2, \qquad \text{(A - 19)}$$

where τ_1 and τ_2 denote the torques applied to the joints, and $c_x = \cos\vartheta_x$, $s_x = \sin\vartheta_x$, and

$c_{xy} = \cos(\vartheta_x + \vartheta_y)$, $s_{xy} = \sin(\vartheta_x + \vartheta_y)$.

Parameterization of the dynamic model (A-18) and (A-19) is derived as follows

$$\pi = [\pi_1 \;\; \pi_2 \;\; \pi_3 \;\; \pi_4 \;\; \pi_5 \;\; \pi_6 \;\; \pi_7 \;\; \pi_8]^T, \qquad \text{(A - 20)}$$

$$\pi_1 = m_1 = m_{l_1} + m_{m_2}, \qquad \pi_6 = m_{l_2}(l_2 - a_2),$$
$$\pi_2 = m_{l_1}(l_1 - a_1), \qquad \pi_7 = \hat{I}_2 = I_{l_2} + m_{l_2}(l_2 - a_2)^2,$$
$$\pi_3 = \hat{I}_1 = I_{l_1} + m_{l_1}(l_1 - a_1)^2 + I_{m_2}, \qquad \pi_8 = I_{m_2},$$
$$\pi_4 = I_{m_1},$$
$$\pi_5 = m_2 = m_{l_2}.$$

If the model of dynamic motion is expressed in the form of

$$\tau = Y(q,\dot{q},\ddot{q})\pi, \tag{A - 21}$$

then the regressor is

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{16} & y_{17} & y_{18} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & y_{26} & y_{27} & y_{28} \end{bmatrix}, \tag{A - 22}$$

$$y_{11} = a_1^2\ddot{\vartheta}_1 + a_1 g c_1,$$

$$y_{12} = 2a_1\ddot{\vartheta}_1 + g c_1,$$

$$y_{13} = \ddot{\vartheta}_1,$$

$$y_{14} = k_{r1}^2\ddot{\vartheta}_1,$$

$$y_{15} = (a_1^2 + 2a_1a_2c_2 + a_2^2)\ddot{\vartheta}_1 + (a_1a_2c_2 + a_2^2)\ddot{\vartheta}_2 - 2a_1a_2s_2\dot{\vartheta}_1\dot{\vartheta}_2$$
$$- a_1a_2s_2\dot{\vartheta}_2 + a_1 g c_1 + a_2 g c_{12},$$

$$y_{16} = (2a_1c_2 + 2a_2)\ddot{\vartheta}_1 + (a_1c_2 + 2a_2)\ddot{\vartheta}_2 - 2a_1s_2\dot{\vartheta}_1\dot{\vartheta}_2 - a_1s_2\dot{\vartheta}_2^2 + g c_{12},$$

$$y_{17} = \ddot{\vartheta}_1 + \ddot{\vartheta}_2,$$

$$y_{18} = \ddot{\vartheta}_1 + k_{r2}\ddot{\vartheta}_2,$$

$$y_{21} = 0,$$

$$y_{22} = 0,$$

$$y_{23} = 0,$$

$$y_{24} = 0,$$

$$y_{25} = (a_1a_2c_2 + a_2^2)\ddot{\vartheta}_1 + a_2^2\ddot{\vartheta}_2 + a_1a_2s_2\dot{\vartheta}_1^2 + a_2 g c_{12},$$

$$y_{26} = (a_1c_2 + 2a_2)\ddot{\vartheta}_1 + 2a_2\ddot{\vartheta}_2 + a_1s_2\dot{\vartheta}_1^2 + g c_{12},$$

$$y_{27} = \ddot{\vartheta}_1 + \ddot{\vartheta}_2,$$

$$y_{28} = k_{r2}\ddot{\vartheta}_1 + k_{r2}^2\ddot{\vartheta}_2.$$

A.2 SCORBOT ER VII Dynamic Model

The dynamic model of SCORBOT ER VII is derived in Industrial Automation Laboratory and taken from [21]. The DH and dynamic parameters of this manipulator for the first three axis are given in Tables A.2 and A.3, respectively. Using them, after performing all the required calculations, the elements of the inertia matrix of the SCORBOT ER VII are:

$$B_{11} = 0.1575c_{2q_2+q_3} + 0.033c_{q_2} + 0.1575c_{q_3} + 0.0478c_{2q_2+2q_3} + 0.27725c_{2q_2}$$
$$+ 0.0525c_{q_2+q_3} + 0.9499, \tag{A - 23}$$
$$B_{12} = -0.019s_{q_2+q_3} - 0.012s_{q_2}, \tag{A - 24}$$
$$B_{13} = -0.019s_{q_2+q_3}, \tag{A - 25}$$

$$B_{21} = -0.019s_{q_2+q_3} - 0.012s_{q_2}, \tag{A - 26}$$
$$B_{22} = 0.315c_{q_3} + 0.94, \tag{A - 27}$$
$$B_{23} = 0.1575c_{q_3} + 0.1656, \tag{A - 28}$$

$$B_{31} = -0.019s_{q_2+q_3}, \tag{A - 29}$$
$$B_{32} = 0.1575c_{q_3} + 0.1656, \tag{A - 30}$$
$$B_{33} = 0.1656. \tag{A - 31}$$

The non - zero elements of the Christoffel symbols are:

$$C_{112} = -0.1575s_{2q_2+q_3} - 0.0478s_{2q_2+2q_3} - 0.0165s_{q_2} - 0.02625s_{q_2+q_3} - 0.27725s_{2q_2}, \tag{A - 32}$$
$$C_{113} = -0.07875s_{2q_2+q_3} - 0.0478s_{2q_2+2q_3} - 0.07875s_{q_3} - 0.02625s_{q_2+q_3}, \tag{A - 33}$$
$$C_{121} = -0.1575s_{2q_2+q_3} - 0.0478s_{2q_2+2q_3} - 0.0165s_{q_2} - 0.02625s_{q_2+q_3} - 0.27725s_{2q_2}, \tag{A - 34}$$
$$C_{122} = -0.012c_{q_2} - 0.019c_{q_2+q_3}, \tag{A - 35}$$
$$C_{123} = -0.019c_{q_2+q_3}, \tag{A - 36}$$
$$C_{131} = -0.07875s_{2q_2+q_3} - 0.0478s_{2q_2+2q_3} - 0.07875s_{q_3} - 0.02625s_{q_2+q_3}, \tag{A - 37}$$
$$C_{132} = -0.019c_{q_2+q_3}, \tag{A - 38}$$

$$C_{133} = -0.019c_{q_2+q_3}, \quad \text{(A - 39)}$$

$$C_{211} = 0.1575s_{2q_2+q_3} + 0.0478s_{2q_2+2q_3} + 0.0165s_{q_2} + 0.02625s_{q_2+q_3} + 0.27725s_{2q_2}, \quad \text{(A - 40)}$$
$$C_{223} = -0.1575s_{q_3}, \quad \text{(A - 41)}$$
$$C_{232} = -0.1575s_{q_3}, \quad \text{(A - 42)}$$
$$C_{233} = -0.1575s_{q_3}, \quad \text{(A - 43)}$$

$$C_{311} = 0.07875s_{2q_2+q_3} + 0.0478s_{2q_2+2q_3} + 0.07875s_{q_3} + 0.02625s_{q_2+q_3}, \quad \text{(A - 44)}$$
$$C_{322} = 0.1575s_{q_3}. \quad \text{(A - 45)}$$

The non - zero components of the gravity vector result as :

$$G_2 = \text{-}3.2373c_{q_2} - 5.15c_{q_2+q_3}, \quad \text{(A - 46)}$$
$$G_3 = -5.15c_{q_2+q_3}. \quad \text{(A - 47)}$$

The SCORBOT ER VII Denavit-Hartenberg parameters, estimated masses and inertia are:

Link	θ **[rad]**	d **[m]**	a **[m]**	α **[rad]**
1	$\theta_1 = 0$	$d_1 = 0.3585$	$a_1 = 0.050$	$\alpha_1 = -\pi/2$
2	$\theta_2 = 0$	$d_2 = -0.037$	$a_2 = 0.300$	$\alpha_2 = 0$
3	$\theta_3 = 0$	$d_3 = 0.0$	$a_3 = 0.250$	$\alpha_2 = 0$

Table A.2. The SCORBOT ER VII Denavit-Hartenberg parameters.

Mass [kg]	I_x [kgm^2]	I_y [kgm^2]	I_z [kgm^2]
$m_1 = 0$	$I_{x1} = 0.00$	$I_{y1} = 0.05$	$I_{z1} = 0.00$
$m_2 = 6.6$	$I_{x2} = 0.10$	$I_{y2} = 0.6$	$I_{z2} = 0.60$
$m_3 = 4.2$	$I_{x3} = 0.02$	$I_{y3} = 0.20$	$I_{z3} = 0.30$

Table A.3. The SCORBOT ER VII estimated masses and inertias.

Appendix B

The Manual for Using the Expert System with Control Toolbox

These detailed explanations about using the expert system and control toolbox are provided as a guide for the potential user wishing to exploit the system's full capabilities. As it is mentioned in previous chapters, one of the main features of this system is its 'openness'. This feature enables users to add/change their own control modules, rules, membership functions in the expert system module as well as add/change the control architecture, controllers, gain tuner in the control toolbox module. Brief explanations about how to create a code and compile it in ORTS are given in the following sections.

B.1 The Control Toolbox

The control toolbox is running on real-time open architecture system ORTS. All control algorithms are created as a fast DSP process, which means that all processes have both the initialization and iteration functions. When a fast process is created, the initialization function is called, and if that function returns *R_Success,* then the process is registered inside the specified process group.

The hardware architecture of ORTS consists of two different environments: a DSP environment and a PC environment. Every new function has to be created in PC environment in C-code and then has to be linked to DSP environment by compiling the code. Example of C-code defined in PC environment is given in Figure B.1.

```
#include "gen-dsp.h"
/* Include the DSP functions inside the gen-dsp.h header file */
#include other source files

#define macro_name char-sequence

#ifdef _FPF_NameOfFunction
/* beginning of process */

typedef struct
{
      Variables
}NameOfStructure

#define   VariableName   (p->VariableName)
/* variable macor definitions for convenience */

rtncode FPF_NameOfFunctionInit (int NumInputLinks, Link **InputLink,
int NumOutputLinks, Link **OutputLink, int NumParameters, float
*Parameter, float Freq, void **Ptr)
/* initialization function */

{
      NameOfStructure *p;
      /*create pointer to data structure */

      if((p=Malloc(sizeof(NameOfStructure)))==NULL) return
R_DSP_Out_of_memory;
      /*allocate memory for local variables */

      *Ptr=p;
      /* assign process data structure address to data passing pointer
*/

      if(NumInputLinks!=InputLinkExpected) {Print("Incorrect number of
      Input Links"); return R_RCP_Error;}
      if(NumOutputLinks!=OutputLinkExpected) {Print("Incorrect number
      of Output Links"); return R_RCP_Error;}
      if(NumParameter!=ParameterExpected) {Print("Incorrect number of
      parameter"); return R_RCP_Error;}
      /* check number of links and parameters */

      /***************************/
       variables and links initialization
       parameters assignment
      /***************************/
      return R_Success;
      /* exit initialization function to start continuous function */
}
#undef   VariableName
#define   VariableName   (((NameOfStructure*)Ptr)->VariableName)
/* variable macro definitions for convenience */

void FPF_NameOfFunction (void *Ptr)
/* continuous function */
{
      /********************************/
       implement and apply the control law
      /********************************/
}

#undef   VariableName
/* undefine variable macros */

#endif
/* end of process */
```

Figure B.1. Example C-code for PC-environment.

For detailed explanation about C functions used in this generalized format of code algorithm, user should refer to ORTS Users Guide [55].

After preparing the C-code, the following procedure is required in order to link the PC-environment and DSP-environment.

1. Open Borland C++.
2. Open `D:\orts\orts.ide`. (Note: if there is no window pops up, click 'view' button and then 'project')
3. Add `FileName.cc` node under DSP software.
4. Add `gen-dsp.h` node under `FileName.cc`.
5. Add `shared.h`, `kernel.h`, and `rtncode.h` nodes under `gen-dsp.h`.
6. Open `D:\orts\project\buildc32.bat`.
7. Add a new line of `cl30 -v30 -x2 -o2 -s -q FileName.cc -eo.o3x -frc:\out\c32 -dDSP_SPECTRUM_C32` inside `buildc32.bat` file.

8. Open `gen-dsp.h` file that is created.
9. Add `#define _FPF_NameOfFunction` in a similar fashion as written in `gen-dsp.h` file.
10. Open `D:\orts\project\userprcs.cc`.
11. Inside `userprcs.cc` file, add

    ```
    rtncode NameOfFunctionInit (int,Link**,int,Link**,int,float*,
    float,void**);
    ```

    under the comment `/*Fast process init functions*/`.
12. Also, add

    ```
    void FPF_NameOfFunction (void);
    ```

    under the comment `/*Fast process functions */`.
13. And add

    ```
    #ifdef _FPF_NameOfFunction
    ("NameOfFunction", FPF_NameOfFunctionInit, FPF_NameOfFunction),
    #endif
    ```

    under the line `struct RC_FAST_PROCESS_LIST_STRUCT RC_FAST_PROCESS_LIST[]`
14. Run `D:\orts\project\buildc32.bat>c32.log` in DOS environment.
15. If there is no error messages shown in `c32.log` file, overwrite

    `D:\expcode\dsp-c32.out` by `D:\orts\project\dsp-c32.out`.

After the linking process, user should create a script file to run the controller in DSP environment. Example of a script file (e.g. `Controller.spt`) used to implement a controller is given in Figure B.2.

```
DSP C32:
Link switchDecoded                    (local,direct,5);
Link referenceExternal                (local,direct,5);
Link controlLog                       (dsp2pc,buffered,30,21);
Link encoder_external                 (local,direct,5);
Link trajectory                       (pc2dsp,buffered,30,5);
```

```
Group switches, priority=1, freq=200:
{
    Link switchRaw(2);
    MFIO_ReadDigitalPort(1,4), output=(switchRaw);
    DecodeSwitches(), input=(switchRaw), output=(switchDecoded);
}

Gain(1,1,1,1,1), input=(trajectory), priority=0, freq=1000,
output=(referenceExternal);

Group control, priority=1, freq=1000:
{
    Link reference(5),encoder(5),error(5),control(5),current(5);

    /* transfer reference signal */
    CheckInputLink(), input=(referenceExternal), output=(reference);

    /* read encoders */
    MFIO_ReadEncoder(1,2,3,4,6), output=(encoder);

    /* read motor current */
    MFIO_ReadADC(1,2,3,4,6), output=(current);

    /* implement PID control */
    Sum(1,-1), input=(reference,encoder), output=(error);

    /*************************************************************/
      Control Architecture Implementation
      e.g. a PID controller

    PIDControl(   40,25,1.5,40,          /* axis 1 - Kp, Ki, Kd, fc */
                  20,20,1.5,40,          /* axis 2 - Kp, Ki, Kd, fc */
                  45,25,1.5,40,          /* axis 3 - Kp, Ki, Kd, fc */
                  55,35,1.5,40,          /* axis 4 - Kp, Ki, Kd, fc */
                  40,30,1,40  )          /* axis 5 - Kp, Ki, Kd, fc */
                ,input=(error), output=(control);

    /*************************************************************/

    /* write to DACs */
    MFIO_WriteDAC(1,2,3,4,6), input=(control);

    /* log control performance */
    Log(1), input=(reference,encoder,control,current),
output=(controlLog);
}

PC:
priorityclass=high;

SaveToDisk("D:\ControlToolbox\Expdata\controller.log"),
input=(controlLog);
ReadFromDisk("D:\ControlToolbox\Trajectories\controller.dat"),
output=(trajectory);
```

Figure B.2. Example script file.

All control algorithms applicable for the SCORBOT ER VII can be found in `D:\ControlToolbox\ControlArchitecture\FileName.dat`. Different types of trajectories can be found in `D:\ControlToolbox\Trajectories\FileName.dat`; all resulting data are stored in `D:\ControlToolbox\Expdata\FileName.dat` folder; and all plots are stored in `D:\ControlToolbox\Plots\FileName.m`.

The following procedure enables the user to run the certain control algorithm and apply it to the SCORBOT ER VII:

1. Turn on the amplifier (black button on the left-hand side).
2. Open the ORTS editor and wait until the `STARTUP.SPT` runs.
3. Switch on the red button located on the back of the amplifier.
4. Open and run the `D:\ControlToolbox\ControlArchitecture\AxisHome01.spt` file. (Note: calibration of robot axis takes place)
5. Switch off the red.
6. Close the ORTS editor.
7. Re-open the ORTS editor and wait until the `STARTUP.SPT` runs.
8. Switch on the red button again.
9. Open and run the `D:\ControlToolbox\ControlArchitecture\Controller.spt` file..
10. Repeat the Step 5 to Step 9 in order to run a new script file.

B.2 The Expert System Toolbox

The expert system is meant to be a "decision maker" for certain control task. Due to the lack of applicable functions that are developed for this version of Fuzzy Logic Toolbox, this part of system works off-line. After the decision for certain robot task is made, the signal is sent to control toolbox explained in previous section, and the control action takes place for desired trajectory. This toolbox is implemented in MATLAB Toolbox v.5.3 using the Fuzzy Logic Toolbox v.2.0.1. Due to the large number of rules that system uses, the system is slow and some other alternatives should be considered for real industrial applications. All necessary files for running the expert system are selected in the `D:\ExpSys\FileName.m`. To run the expert system, the following procedure should be made:

1. Open the MATLAB R11 Command Window.

2. Type: start /* initialization of `start.m` file for beginning of process */

When the process begins the matlab interface is shown on Figure B.3.

```
***********************WELCOME************************

Which type of industrial processes are you interested in?

1. Manufacturing

2. Material Handling

3. Measuring

Enter 1, 2, 3 or 0 for none:

/* if user enter 1 than questionnaire follows */

Which industrial task in this group do you want to accomplish?

1. spot welding

2. spray painting

3. laser cutting

4. arc welding

5. drilling

6. polishing
```

7. deburring

8. screw fastening

9. milling

Enter the number from 1 to 9 or 0 for none of these:

/* If the user type 1 than the next question follows */

Your application is point-to-point non-contact task.

Please provide information about:

1. Model confidence and model type:

/* The matlab window pops-up the membership functions for this module and
after pressing enter button the interface shows next line */

input [a b c d] =[ ] /* user is required to enter a four values input for this module. Given the input values, matlab invokes the **.fis** file associated with this module, calculates and save the final deffuzified value of this module*/

2. Is position sensor available?

1) Yes

2) No

Type 1 or 2:

/* if yes, than window pops-up the membership functions for this module and user has to enter the input values */

3. Is velocity sensor available?

1) Yes

2) No

Type 1 or 2:

4. Hardware information.

5. External and internal disturbances.

6. Allowable planning task error. ...etc.

/* if the chosen application is some of contact-task problems than different questions follows. When the control architecture is chosen than the questions about the associated controller follows */

```
***********************CONTROLLER SELECTION***************************
Please provide the information about the:
1) trajectory rate
2) driving system of the joints
3) rate of response
4) required control accuracy
```

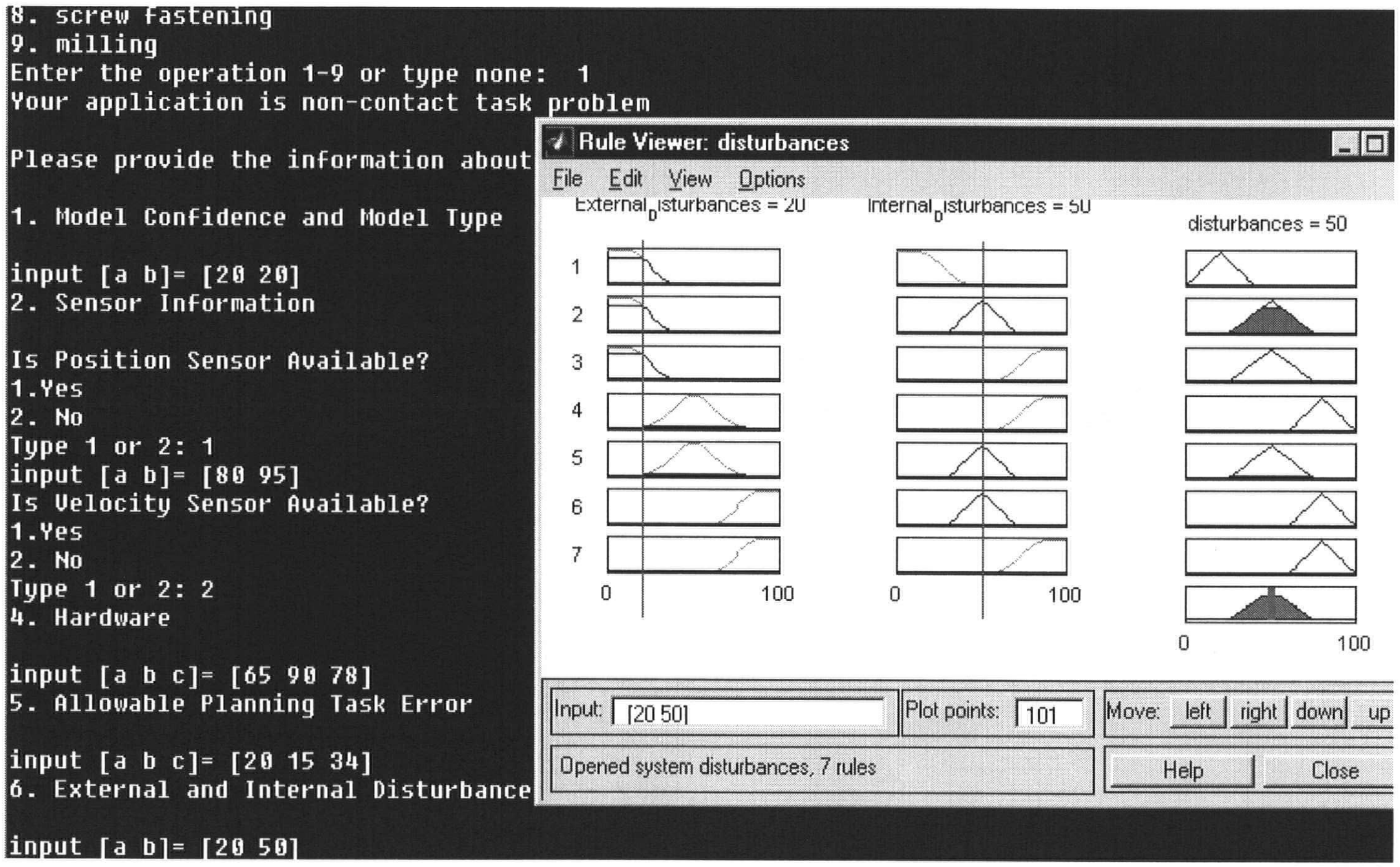
/* windows pops-up the windows with membership functions for this module, and user is required to enter the input values and according to these information the controller structure is chosen */

```
Please provide information about the:
1) degree of stability
2) overshoot
3) rise time
4) allowable final error
5) robot type
6) introduced payload
```

/* according to these information the expert system chose the controller to be incorporated into previously chosen architecture */

Figure B.3. The matlab interface for control architecture and controller selection.

The screen shots of Matlab editor for TREX and data entry screen are shown on next figure.



```
8. screw fastening
9. milling
Enter the operation 1-9 or type none:  1
Your application is non-contact task problem

Please provide the information about

1. Model Confidence and Model Type

input [a b]= [20 20]
2. Sensor Information

Is Position Sensor Available?
1.Yes
2. No
Type 1 or 2: 1
input [a b]= [80 95]
Is Velocity Sensor Available?
1.Yes
2. No
Type 1 or 2: 2
4. Hardware

input [a b c]= [65 90 78]
5. Allowable Planning Task Error

input [a b c]= [20 15 34]
6. External and Internal Disturbance

input [a b]= [20 50]
```

Figure B.4. Screen shots of TREX editor with data entry for one of the modules.

Appendix C

The Simulation Results

In this appendix, the plots with simulation results for 2-dof robot performed using Matlab Toolbox [30], and applied control law are presented for better insight in result difference when different control scheme are utilized for the same task.

C.1 Continuous Path Motion Task

The desired arm trajectory has a typical trapezoidal velocity profile. The end-effector desired motion is a path along the joint space rectilinear path from $\boldsymbol{q}_i$=[0 π/4]^T to $\boldsymbol{q}_f$=[π/2 π/2]^T posture. Trajectory duration is t_f = 1 sec.

C.1.1 Applied Control Law

The block diagram of utilized architecture is presented on Figure C.1. This technique of nonlinear compensation and decoupling is very attractive from a control viewpoint since the nonlinear and coupled manipulator dynamics is replaced with n linear and decoupled second-order subsystems [32]. The manipulator dynamics can be expressed over the relationship:

$$B(q)\ddot{q}+C(q,\dot{q})\dot{q}+F_v\dot{q}+g(q)=\tau. \qquad \text{(C-1)}$$

Consequently, the dynamic model of the system given by the manipulator and drives can be described as:

$$B(q)\ddot{q}+C(q,\dot{q})\dot{q}+F\dot{q}+g(q)=u, \qquad \text{(C - 2)}$$

where

$$F=F_v+K_rK_tR_a^{-1}K_vK_r, \qquad \text{(C - 3)}$$
$$u=K_rK_tR_a^{-1}G_vv_c. \qquad \text{(C - 4)}$$

In (C-3) and (C-4), F is the diagonal matrix accounting for all viscous (mechanical and electrical) damping terms, and u is the vector that is taken as control input to the system.

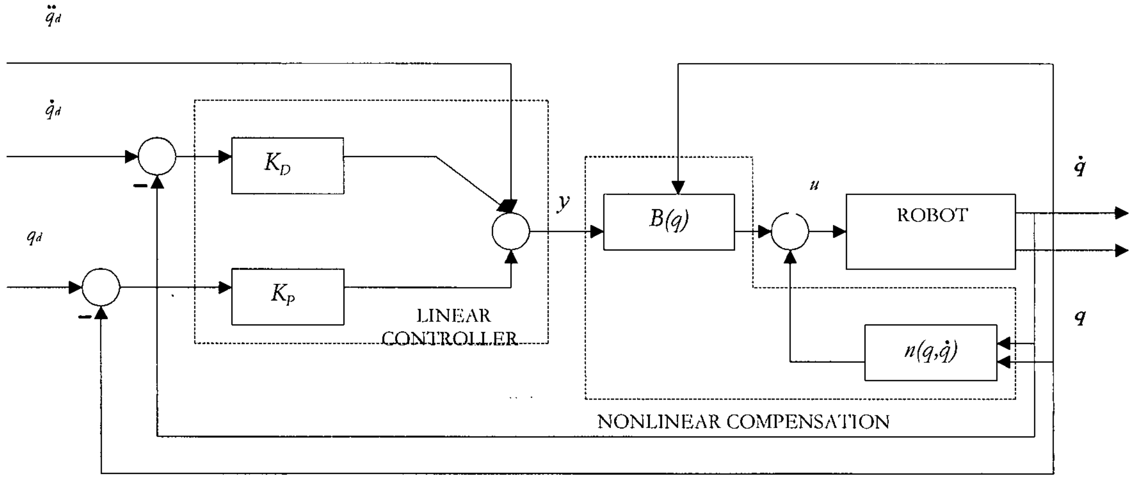


Figure C.1. Computed torque with PD controller.

Considering the equation (C-2), the dynamic model of n-joint manipulator can be reformulated as:

$$B(q)\ddot{q}+n(q,\dot{q})=u, \qquad \text{(C - 5)}$$

where

$$n(q,\dot{q})=C(q,\dot{q})\dot{q}+F\dot{q}+g(q). \qquad \text{(C - 6)}$$

Choosing the control-input *u* as a function of the manipulator state, the system can be described as

$$B(q)y + n(q,\dot{q}) = u, \qquad \text{(C-7)}$$

$$\ddot{q} = y. \qquad \text{(C-8)}$$

The stabilizing control law *y* is chosen as:

$$y = -K_P q - K_D \dot{q} + r, \qquad \text{(C-9)}$$

and leads to the second - order equation,

$$\ddot{q} + K_D \dot{q} + K_P = r. \qquad \text{(C-10)}$$

Choosing K_p and K_D as diagonal matrices of the type

$$K_P = diag\{\omega_{n1}{}^2, ..., \omega_{nn}{}^2\}, \qquad K_D = diag\{2\zeta_1\omega_{n1}, ..., 2\zeta_n\omega_{nn}\}, \qquad \text{(C-11)}$$

gives decoupled system. Given any desired trajectory $q_d(t)$, tracking of this trajectory is ensured by choosing

$$r = \ddot{q}_d + K_D \dot{q}_d + K_P q_d. \qquad \text{(C-12)}$$

The plots are given on Figures C.2 and C.3. Due to high-speed trajectory, and required precise positioning it is necessary to calculate the whole system dynamics, which is achieved choosing computer torque architecture with adequate controller. From given plots, it is obvious that this architecture with properly tuned controller gains satisfies requirements set by user.

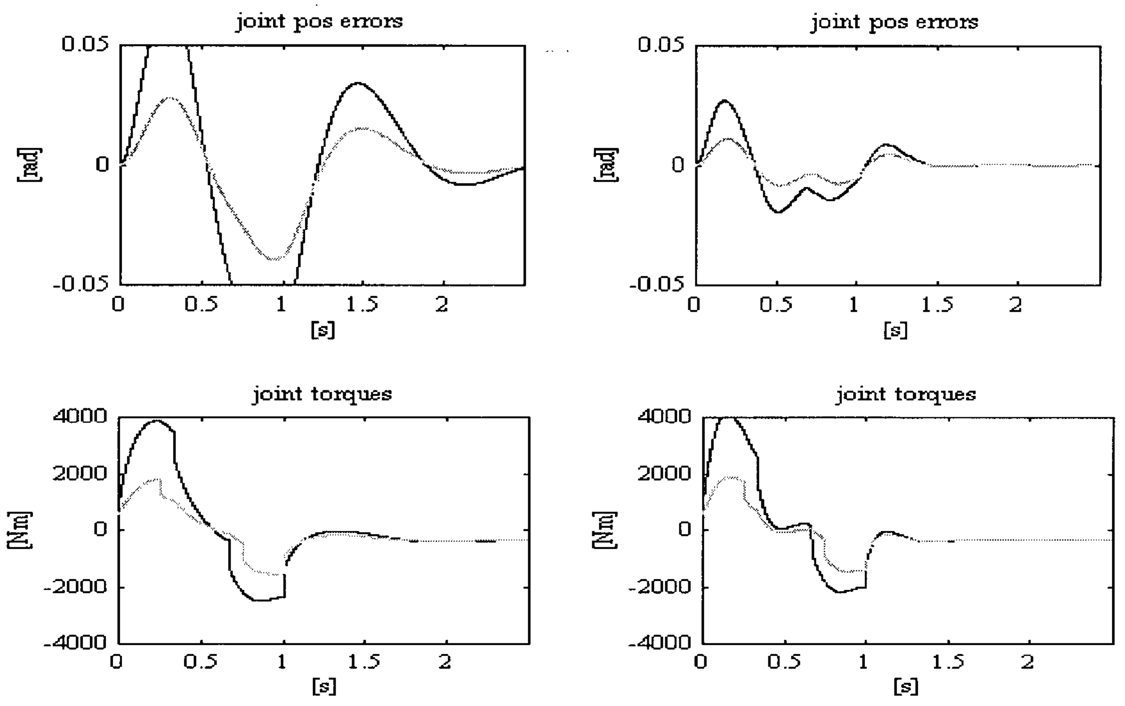


Figure C.2. Independent Joint Control with PD controller and sampling interval t_s=1ms,
a) K_P=5, K_V=10 b) K_P=6.25, K_V=32

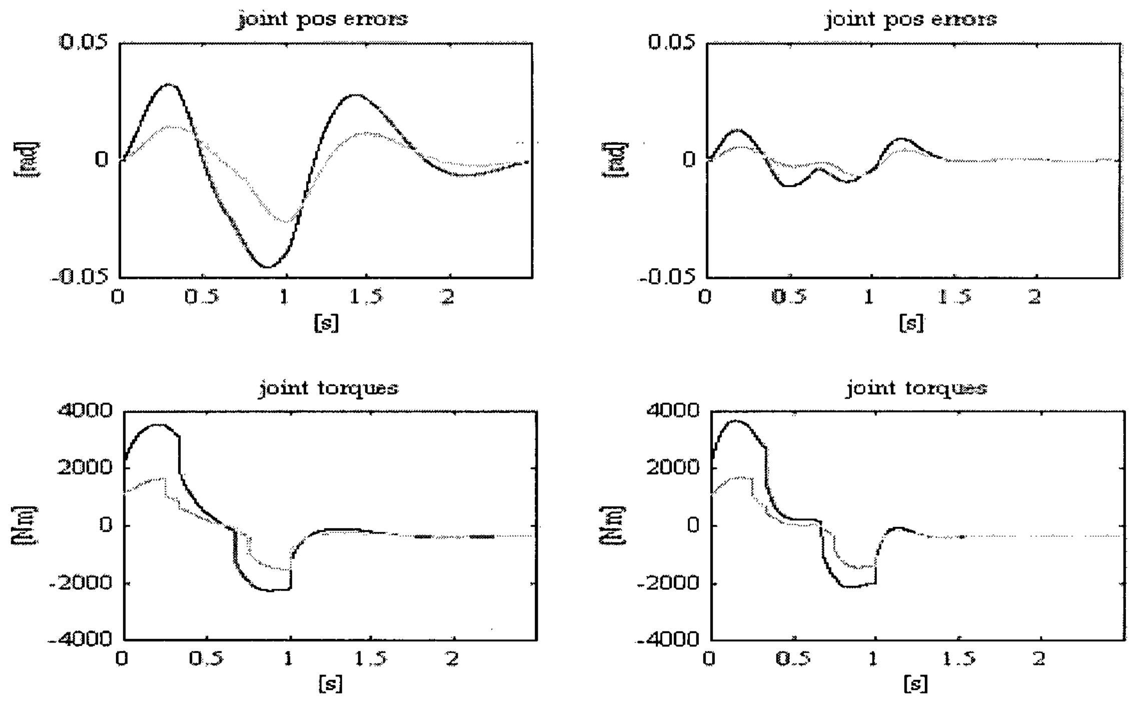


Figure C.3. Computed torque with PD controller, t_s=1ms
a) K_P=5, K_V=10 b) K_P=6.25, K_V=32

C.2 Continuous Path Motion Task with Tip Payload

The desired arm trajectory has a typical trapezoidal velocity profile. The end-effector desired motion is a path along the joint space rectilinear path from q_i=[0 π/4]^T to q_f=[π/2 π/2]^T posture. The concentrated tip payload is of weight m_L = 10 kg. The trajectory duration is t_f=1 sec.

C.2.1 Applied Control Law

The block diagram of utilized architecture is presented on Figure C.4. The possibility of finding adaptive control laws is ensured by the property of linearity in the parameters of the dynamic model of manipulator.

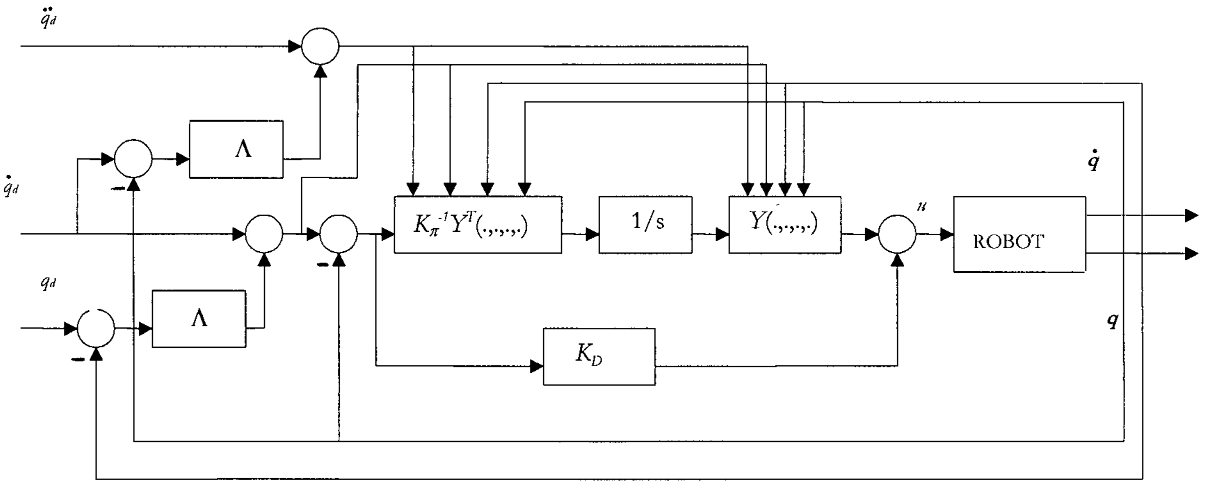


Figure C.4. Adaptive control architecture with PD controller.

Thus, equation (C - 2) can be written as:

$$B(q)\ddot{q}+C(q,\dot{q})\dot{q}+F\dot{q}+g(q)=Y(q,\dot{q},\ddot{q})\pi=u, \qquad \text{(C-13)}$$

where π is a (*p* x 1) vector of constant parameters and *Y* is an (*n* x *p*) matrix that is a function of joint position, velocity and acceleration.

Consider the control law,

$$u=B(q)\ddot{q}_r+C(q,\dot{q})\dot{q}_r+F\dot{q}_r+g(q)+K_D\sigma, \qquad \text{(C-14)}$$

with K_D a positive definite matrix. The choice

$$\dot{q}_r=\dot{q}_d+\Lambda\tilde{q} \qquad \ddot{q}_r=\ddot{q}_d+\Lambda\dot{\tilde{q}}, \qquad \text{(C-15)}$$

with Λ a positive definite matrix, allows expresing the nonlinear compensation and decoupling as a terms as a function of the desired velocity and acceleration, corrected by the current state of the manipulator. The term $K_D\sigma$ is equivalent to a PD controller if σ is taken as

$$\sigma=\dot{q}_r-\dot{q}=\dot{\tilde{q}}+\Lambda\tilde{q}. \qquad \text{(C-16)}$$

Substituting (C - 13) into (C – 12) and considering (C – 15) yields

$$B(q)\dot{\sigma}+C(q,\dot{q})\sigma+F\sigma+K_D\sigma=0. \qquad \text{(C-17)}$$

Differently from the robust control case, the error trajectory is naturally constrained to the subspace $\sigma=0$ without the need of high-frequency control. The control law can be made adaptive with respect to the vector of parameters π [32]. The control law can be modified into:

$$u = \hat{B}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{F}\dot{q}_r + \hat{g} + K_D\sigma = Y(q,\dot{q},\dot{q}_r\,\ddot{q}_r)\hat{\pi} + K_D\sigma, \qquad \text{(C-18)}$$

where $\hat{\pi}$ represents the available estimate on the parameters and, $\hat{B}$, $\hat{C}$, $\hat{F}$, and $\hat{g}$ denote the estimated terms in the dynamic model. Substituing control (C-17) into (C-12) gives:

$$\begin{aligned} B(q)\dot{\sigma} + C(q,\dot{q})\sigma + F\sigma + K_D\sigma &= -\tilde{B}(q)\ddot{q}_r - \tilde{C}(q,\dot{q})\dot{q}_r - \tilde{F}\dot{q}_r - \tilde{g} \\ &= -Y(q,\dot{q},\dot{q}_r\,\ddot{q}_r)\tilde{\pi}, \qquad \text{(C-19)} \end{aligned}$$

where

$$\tilde{\pi} = \hat{\pi} - \pi, \quad \tilde{B} = \hat{B} - B, \quad \tilde{C} = \hat{C} - C, \quad \tilde{F} = \hat{F} - F, \text{ and } \tilde{g} = \hat{g} - g. \qquad \text{(C-20)}$$

Thus, the trajectories of the manipulator described by the model (C-2), under the control law

$$u = Y(q,\dot{q},\dot{q}_r\,\ddot{q}_r)\hat{\pi} + K_D(\dot{\tilde{q}} + \Lambda\tilde{q}), \qquad \text{(C-21)}$$

and the parameter adaptive law,

$$\dot{\hat{\pi}} = K_\pi^{-1} Y^T(q,\dot{q},\dot{q}_r\,\ddot{q}_r)(\dot{\tilde{q}} + \Lambda\tilde{q}), \qquad \text{(C-22)}$$

globally asymptoticaly converge to $\sigma = 0$ and $\tilde{q} = 0$, which implies convergance to zero of $\dot{\tilde{q}}$, $\tilde{q}$.

The plots are given on Figures C.5. - C.7.

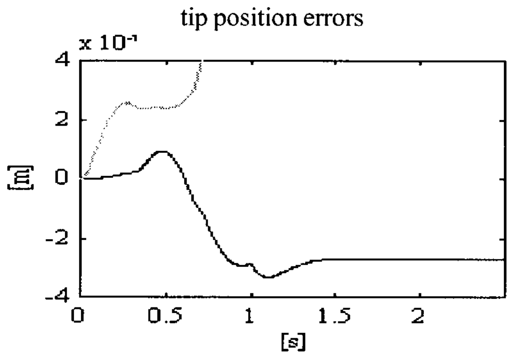


Figure C.5 Inverse Dynamic Control with no load compensation, t_s = 1 ms.

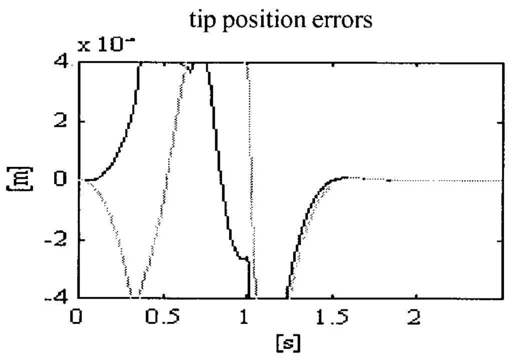
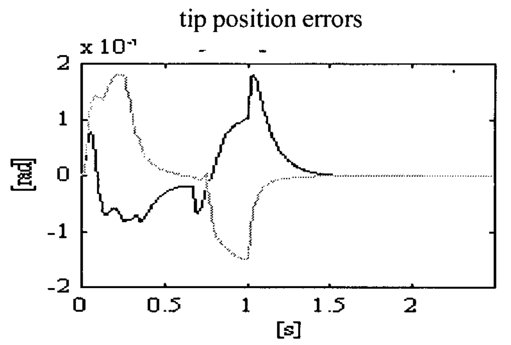


Figure C.6. Inverse Dynamic Control with load compensation, t_s = 1 ms.



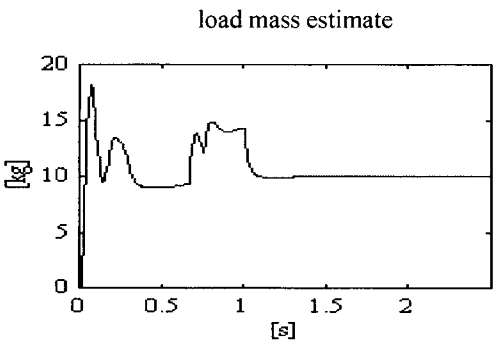


Figure C.7. Adaptive control architecture with load mass estimation, t_s = 1ms.

Comparison of inverse dynamic control with adaptive control scheme shows that latter achieves zero state error in 1.5 sec and load mass is estimated in about 1 sec. which is satisfactory for this application. However, inverse dynamic control without load compensation becomes unstable, whereas the same architecture with load compensation gives rather sluggish behavior, especially during transition process.

C.3 Continuous Path Motion with Applied Force

The desired end-effector trajectory has a typical trapezoidal velocity profile, and robot is in contact with elastically compliant plane. The elastic plane is purely frictionless. The end-effector desired motion is a path along the joint rectilinear path from position q_i=[1+0.1(2)$^{1/2}$]^T to q_f=[1.2+0.1(2)$^{1/2}$]^T.

C.3.1 Applied Control Law

Consider the manipulator dynamic model (C - 2). For the contact task motion, the model can be written in the form

$$B(q)\ddot{q}+C(q,\dot{q})\dot{q}+F\dot{q}+g(q)=u-J^{T}(q)h, \qquad \text{(C-23)}$$

where h is the vector of contact forces exerted by the manipulator's end-effector on the environment. In the force control direction, the control law

$$y=J_{A}^{-1}(q)M_{d}^{-1}(-K_{D}\dot{x}+K_{P}(x_{F}-x)-M_{d}\dot{J}_{A}(q,\dot{q})\dot{q}), \qquad \text{(C-24)}$$

where x_F is a suitable reference to be related to a force error and M_d is a mass matrix. Taking into account Equations (C - 5) and (C - 23) the new relationship is derived,

$$u=B(q)y+n(q,q)+J^{T}(q)h. \qquad \text{(C-25)}$$

Thus, the system can be described now as

$$M_{d}\ddot{x}+K_{D}\dot{x}+K_{P}x=K_{P}x_{F}. \qquad \text{(C-26)}$$

Let h_{Ad} denote the desired constant force reference, the relation between x_F and the force error can be symbolically expressed as:

$$x_F = S_F(h_{Ad} - h_A), \qquad \text{(C – 27)}$$

where S is a selection matrix whose elements give the control action. On the assumption of the elastically compliant environment, Equation (C - 26) becomes

$$M_{d}\ddot{x}+K_{D}\dot{x}+K_{P}(I+S_{F}K_{A})x=K_{P}S_{F}(K_{A}x_{e}+h_{Ad}), \qquad \text{(C-28)}$$

where K_A is environment stiffness matrix. A convenient choice for S_F is a PI action,

$$S_F = K_F + K_I \int (.) d\zeta, \qquad (C-29)$$

and with a proper choice of the matrices K_D, K_P, K_F, and K_I, the system is asymptotically stable. The plots are presented on Figures C.8 and C.9.

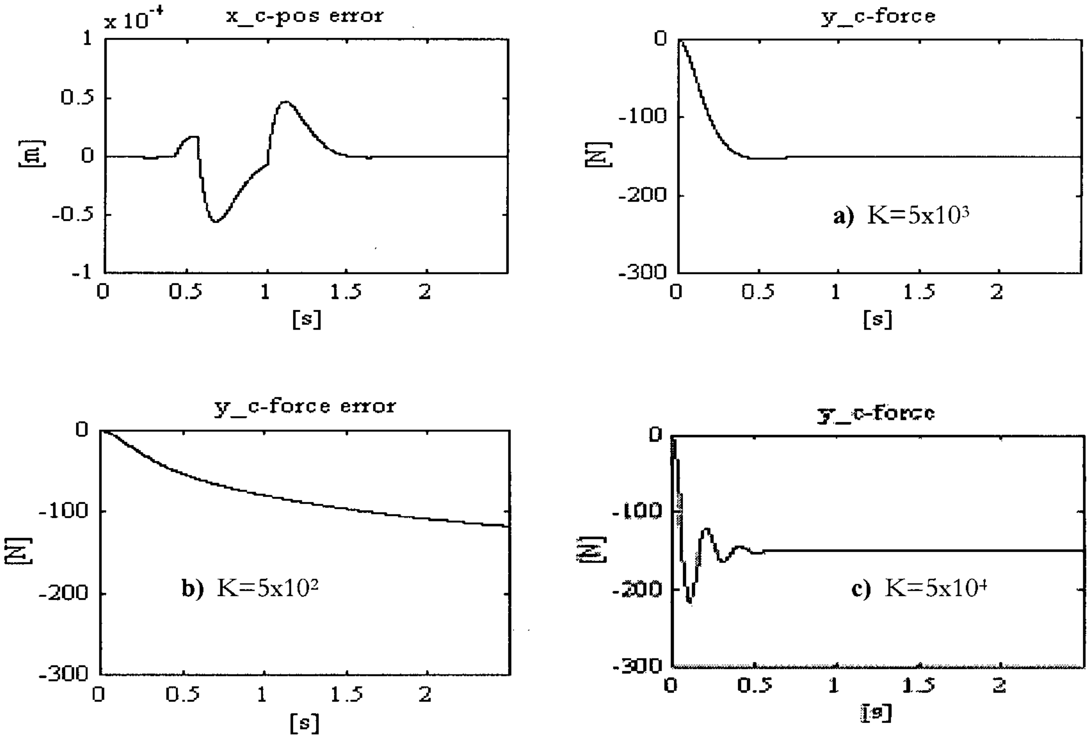


Figure C.8. Explicit Hybrid Position/Force Control, ts=1ms, F_d=150N,

a) environment stiffness K=5x10^3

b) more compliant environment K=5x10^2

c) less compliant environment K=5x10^4

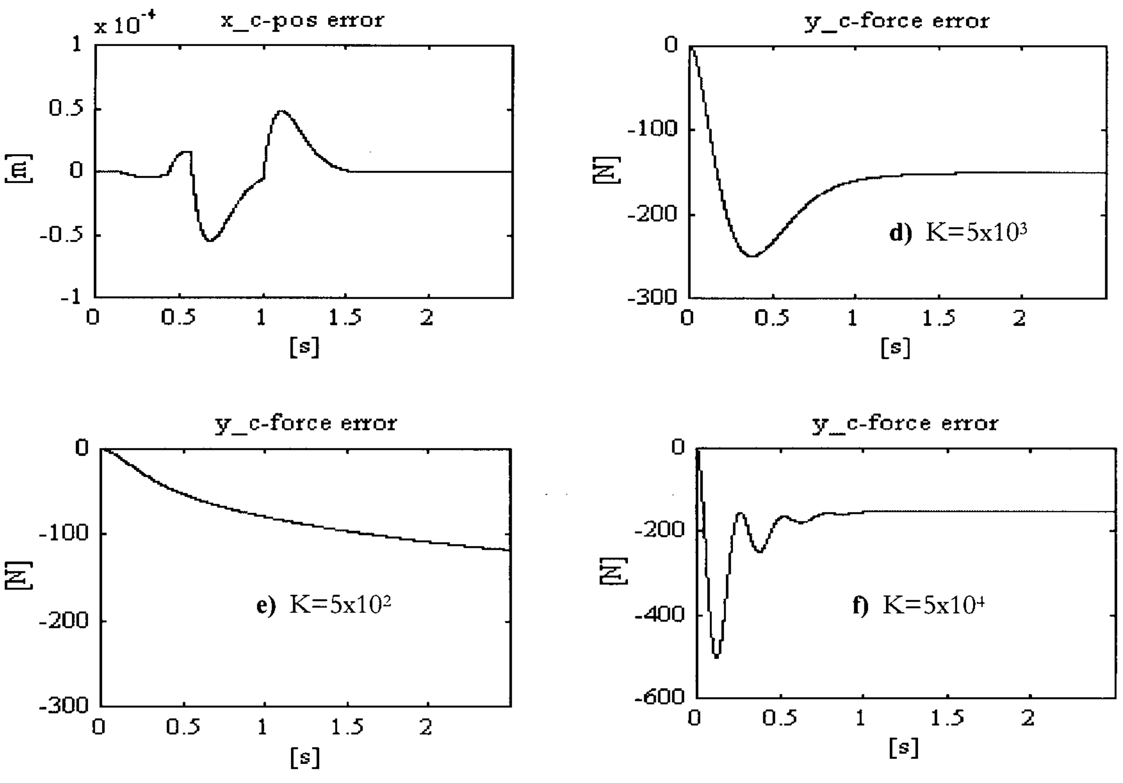


Figure C.9. Implicit Hybrid Position/Force Control, ts=1ms, F_d=150N,

d) environment stiffness K=5x10^3

e) more compliant environment K=5x10^2

f) less compliant environment K=5x10^4

These two architectures give almost the same results in the positioning control part. However, during transition process, implicit hybrid control architecture performs big oscillations in the force control part, especially when robot encounters highly stiff environment. That can cause the damaging of object that the operation is performed on. Even for less stiff environment this architecture gives some overshoot during transition process. In general, explicit hybrid control performs better for this operation in all types of environment.

Appendix D

The Experimental Results

In this appendix, the plots with experimental results for all five SCORBOT ER VII axes are presented. The plots provide better insight in the differences in system response when various algorithms are utilized for the same task.

D.1 The Point-to-Point Motion Task

The results are given for simple point-to-point slow robot motion where the trajectory is given by:

```
PTP_traj( [0.4; -0.2; 0.1], [0.4; 0.4; 0.1], 2, 0.005, 60, 5 )
```

D.1.1 Applied Control Law

The block diagram of utilized architecture is presented on Figure D.1. The zero-order hold is embedded into the architecture of ORTS. The low-pass filter is designed as second order system, and the applied control law is presented below.

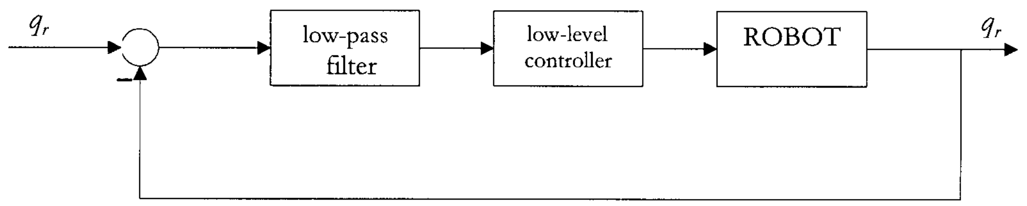


Figure D.1. Independent joint controller with PID controller.

The transfer function of low-pass filter represented in discrete time domain is taken as

$$W_F(z) = \frac{b_2 z^2 + b_1 z + b_0}{z^2 + a_1 z + a_0}, \qquad \text{(D - 1)}$$

and the PID control law in time domain is calculated as

$$u(t) = K_c[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}], \qquad \text{(D - 2)}$$

where the parameters of digital PID controller are calculated as:

$$K_P = K_{PC} - K_{IC}T/2, \qquad K_I = K_{IC}T, \qquad K_D = K_{DC}/T, \qquad \text{(D - 3)}$$

and K_{PC}, K_{DC}, and K_{IC} are predefined PID controller gains in continuos time domain. The sampling period T is defined as $T = 1/Freq$ and in this case control architecture operates on frequency $Freq = \omega_S = 200$ Hz.

The digital filter parameters are calculated as:

b[k] = $K_P + K_I + K_D$, $\qquad$ a[k] = -(1 + $dpole$),

b[k-1] = -(($dpole$ + 1)K_P + $dpoleK_I$ + $2K_D$, $\qquad$ a[k-1] = $dpole$,

b[k-2] = $dpoleK_P + K_D$, $\qquad$ (D - 4) where discrete-time pole is defined as

$$dpole = (1/(2\pi CuttOffFreq))/(T + (1/(2\pi CuttOffFreq))). \qquad \text{(D - 5)}$$

According to Shannon's theorem, the cut-off frequency is chosen not to be larger than half of the sampling frequency.

$$|X(j\omega)| = s, \qquad \forall \omega \geq \frac{\omega_S}{2}. \qquad \text{(D - 6)}$$

Thus, the digital PID control signal is calculated as

$$\begin{aligned} u[k] &= b[k]e[k] + b[k-1]e[k-1] + b[k-2]e[k-2] \\ &\quad - (a[k]u[k-1] + a[k-1]u[k-2]). \end{aligned} \qquad \text{(D - 7)}$$

The plots are shown on Figures D.2 – D.11.

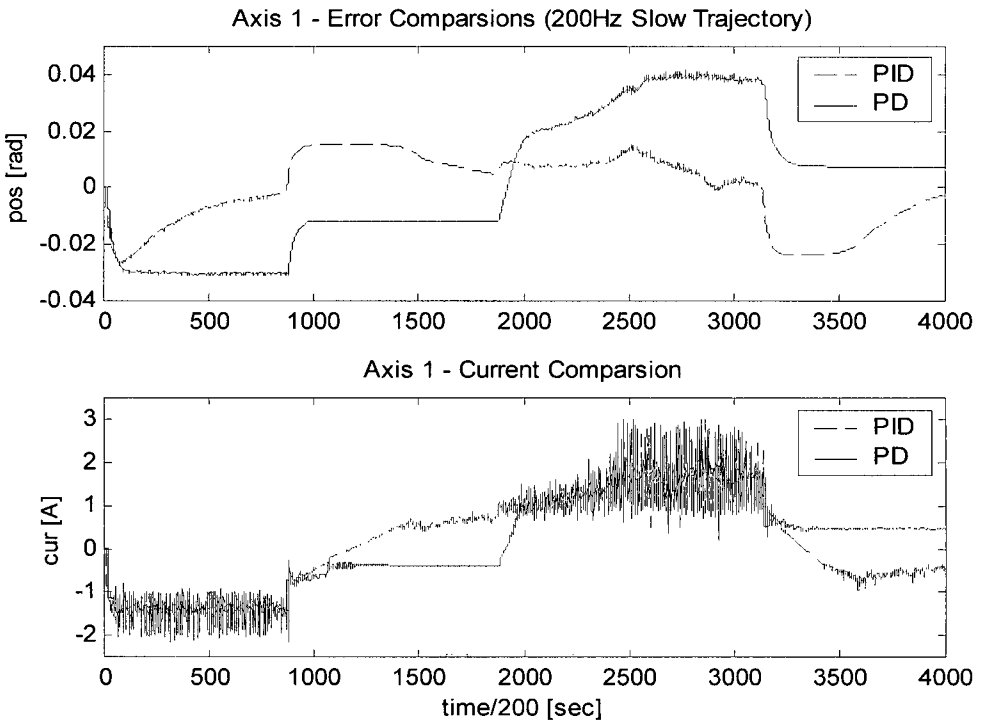


Figure D.2. The first-axis plots for slow robot motion with PID and PD controllers.

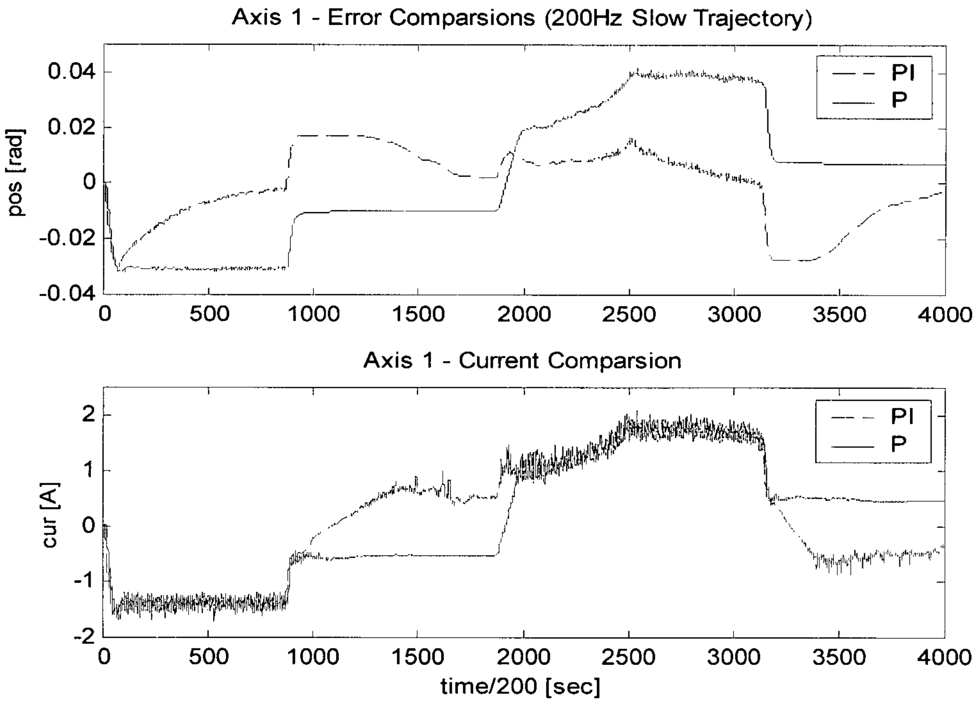


Figure D.3. The first-axis plots for slow robot motion with PI and P controllers.

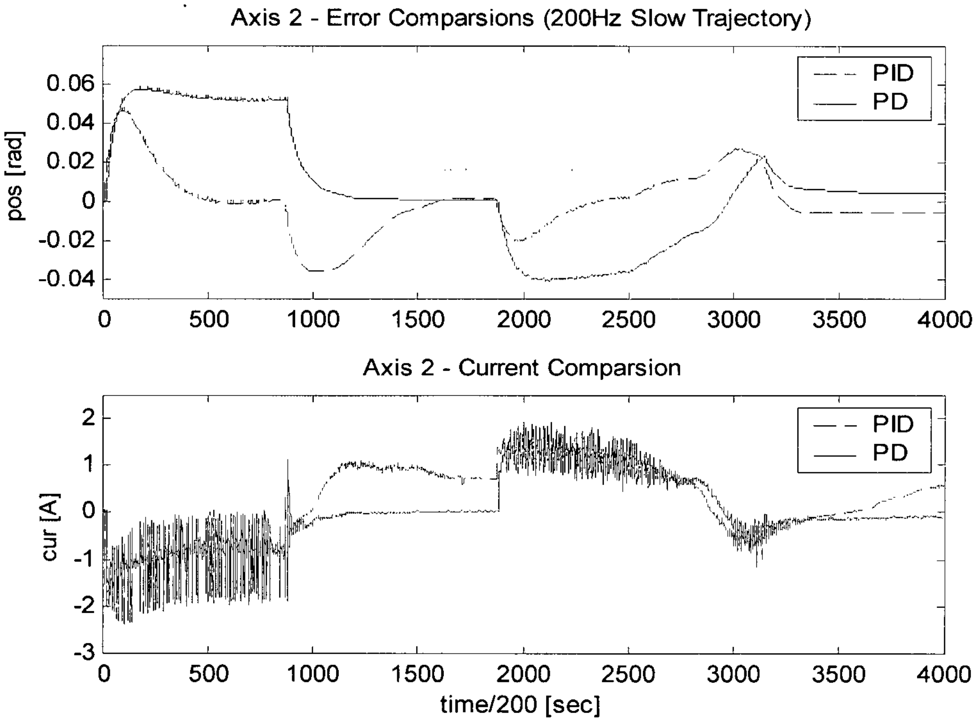


Figure D.4. The second-axis plots for slow robot motion with PID and PD controllers.

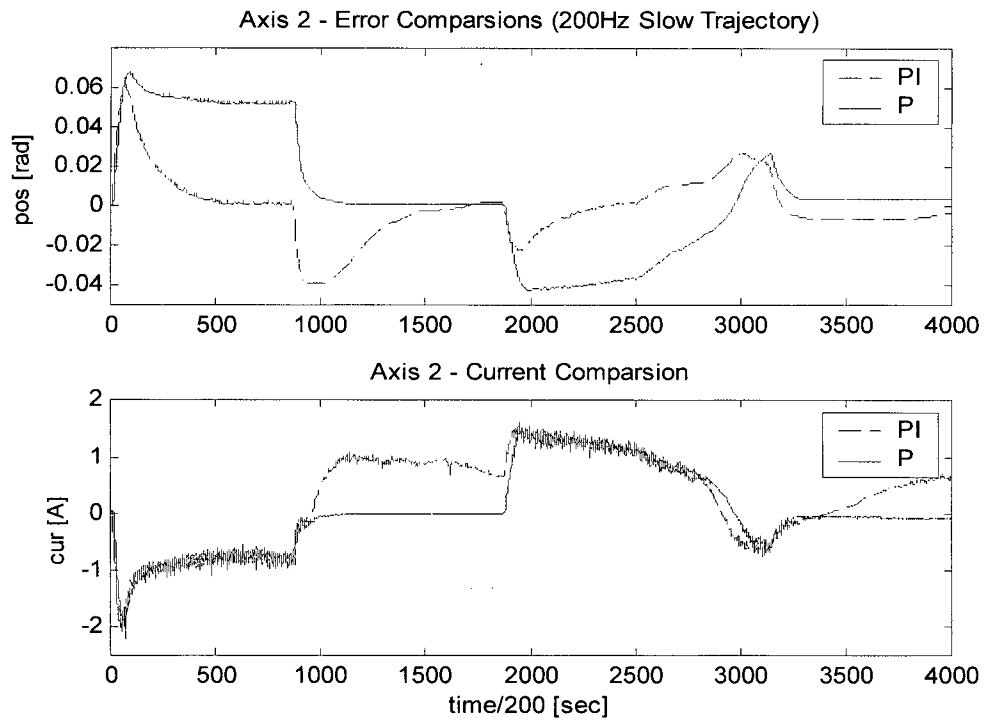


Figure D.5. The second-axis plots for slow robot motion with PI and P controllers.

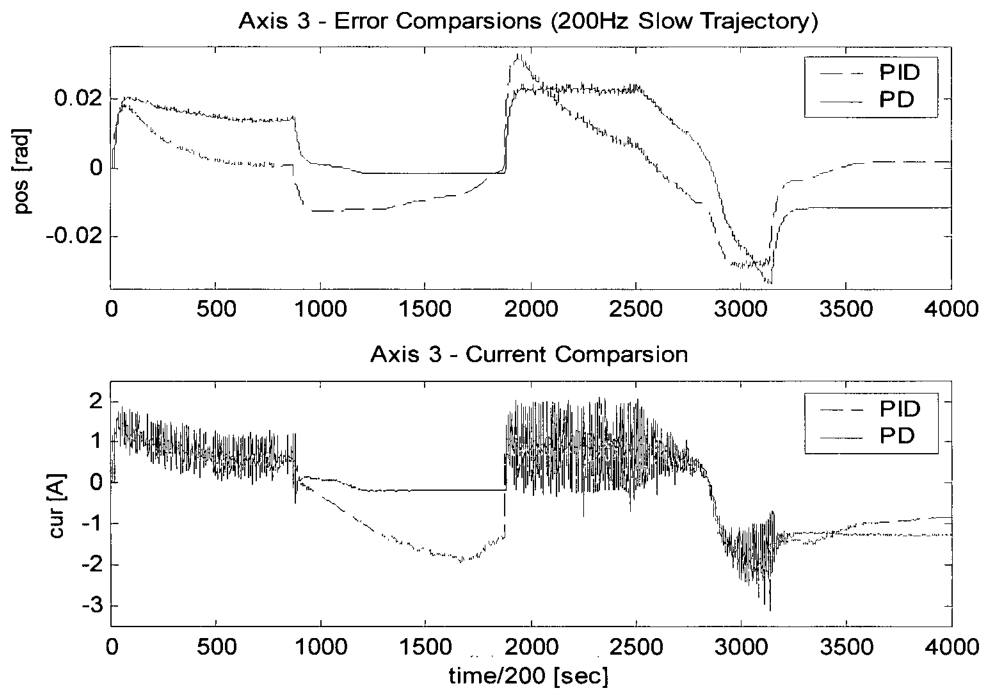


Figure D.6. The third-axis plots for slow robot motion with PID and PD controllers.

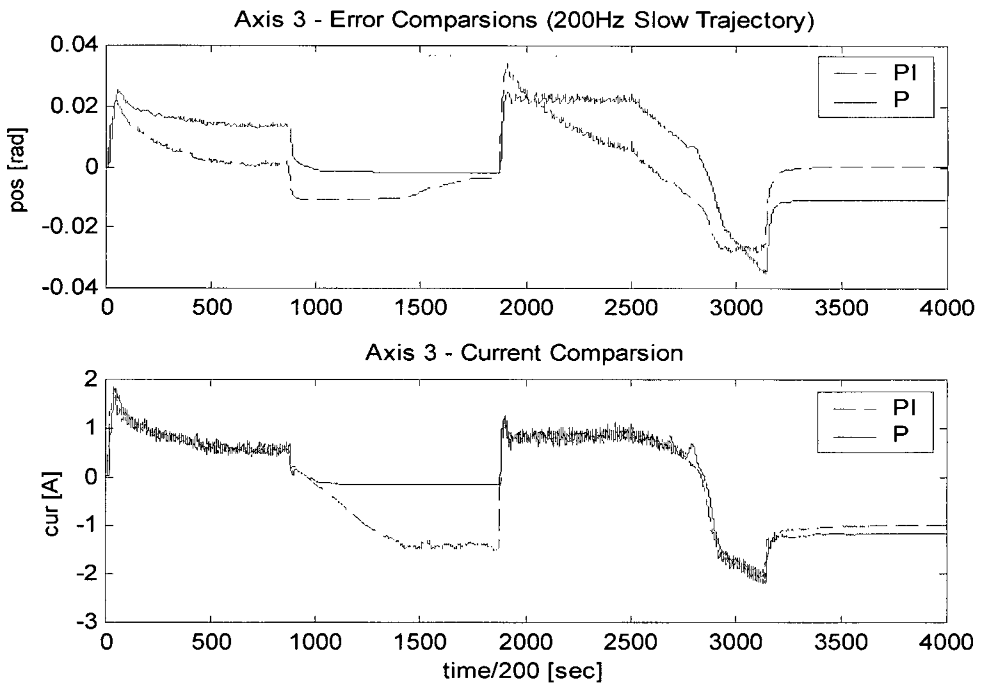


Figure D.7. The third-axis plots for slow robot motion with PI and P controllers.

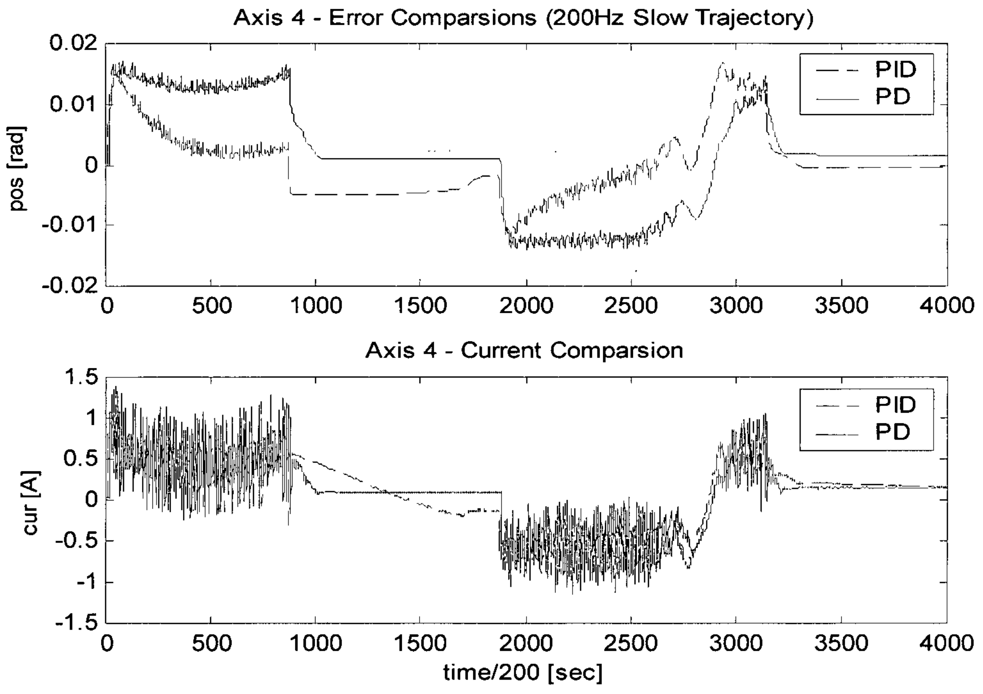


Figure D.8. The fourth-axis plots for slow robot motion with PID and PD controllers.

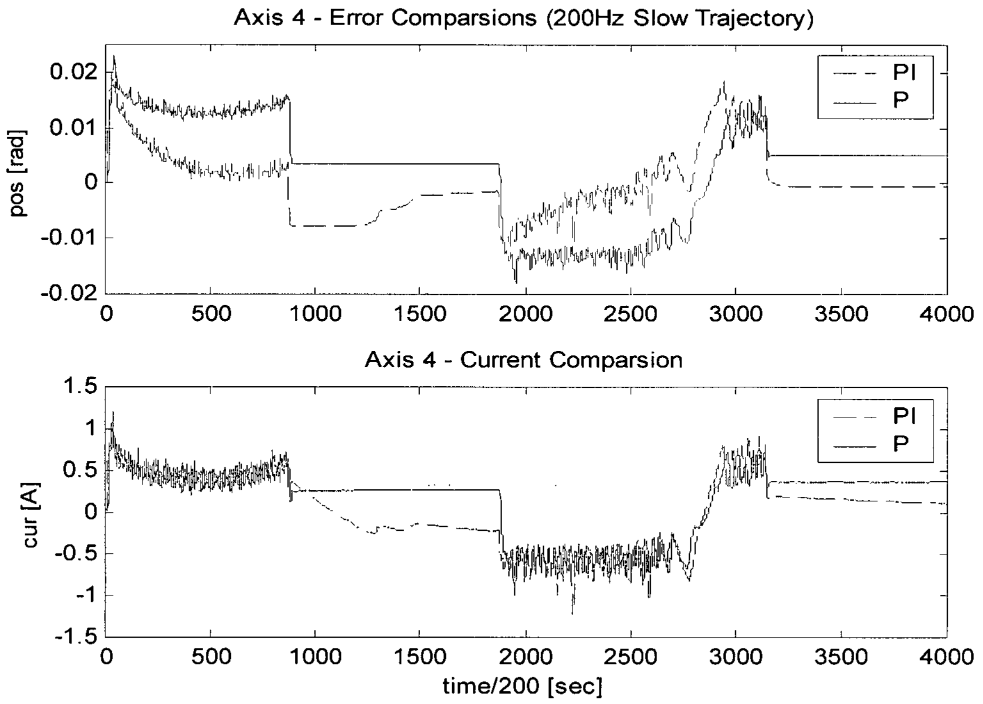


Figure D.9. The fourth-axis plots for slow robot motion with PI and P controllers.

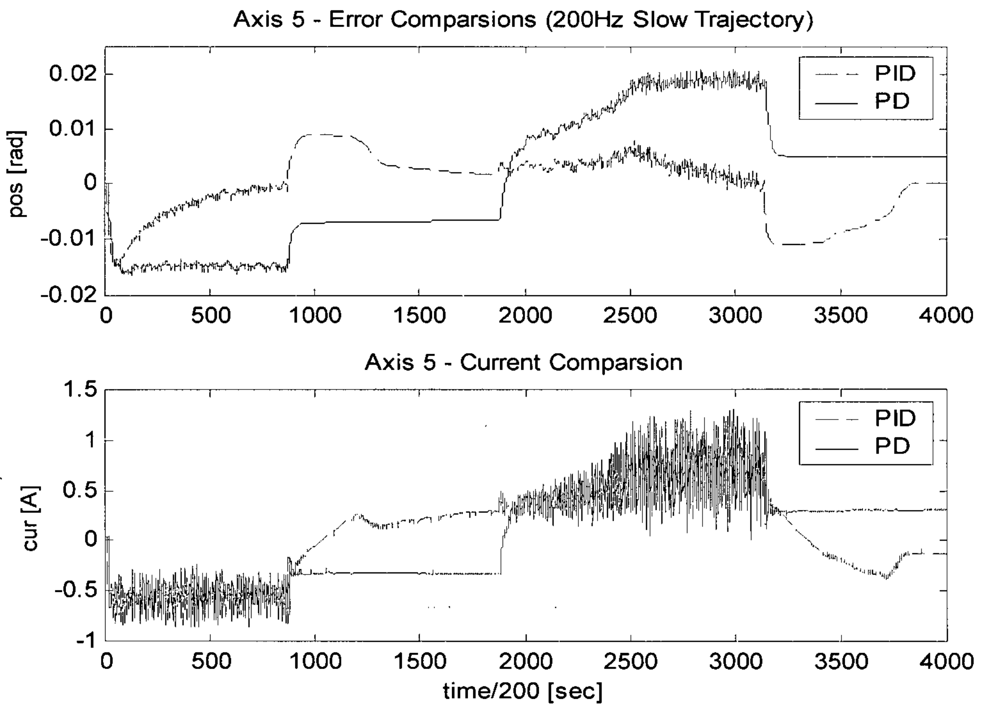


Figure D.10. The fifth-axis plots for slow robot motion with PID and PD controllers.

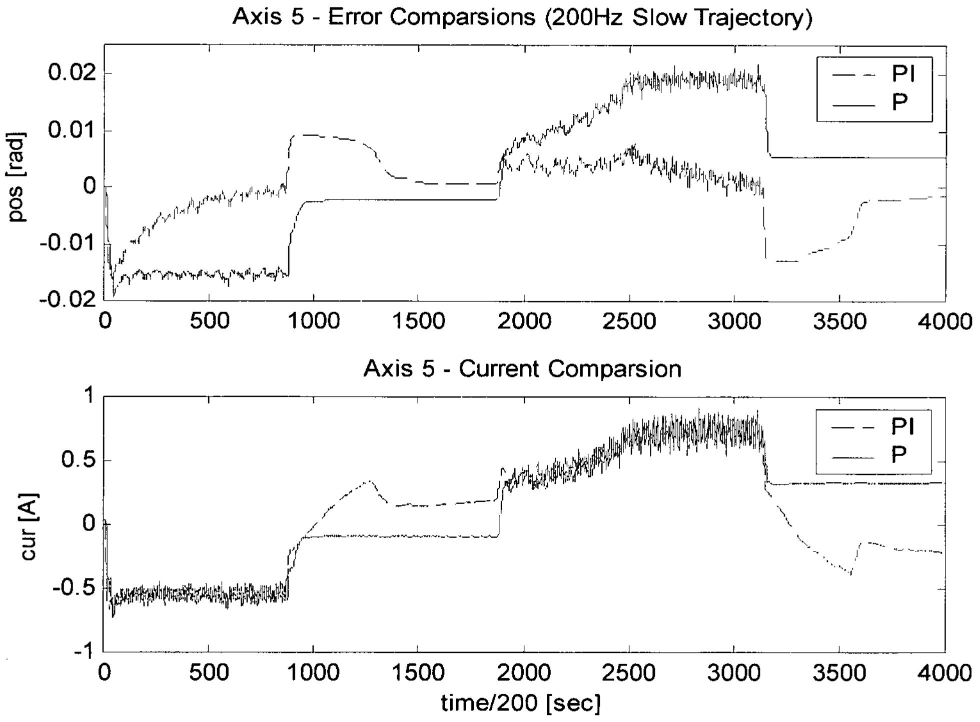


Figure D.11. The fifth-axis plots for slow robot motion with PI and P controllers.

D.2 Point-to-Point Motion Task with the Payload

The results are given for simple point-to-point slow robot motion trajectory where the payload is concentrated at the tip of end-effector. The trajectory is given by:

```
PTP_traj( [0.4; -0.2; 0.1], [0.4; 0.4; 0.1], 2, 0.005, 60, 5 )
```

D.2.1 Applied Control Law

The block diagram of utilized architecture is presented on Figure D.12. The zero-order hold is embedded into the architecture of ORTS. The low-pass filter is designed as second order system, and the applied control law is presented below.

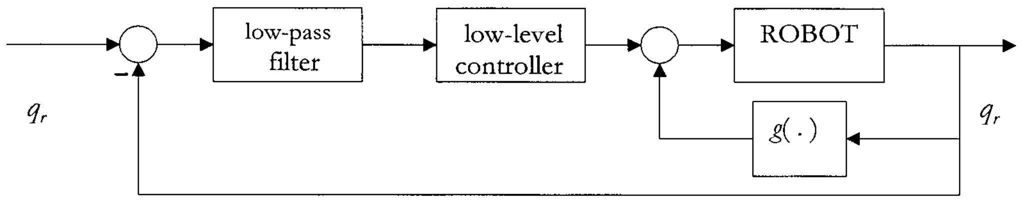


Figure D.12. Independent joint controller with PD+G controller.

The transfer function of low-pass filter represented in discrete time domain is taken as,

$$W_F(z) = \frac{b_2 z^2 + b_1 z + b_0}{z^2 + a_1 z + a_0}, \tag{D - 8}$$

and the PD+G control law in time domain is calculated as,

$$u(t) = g(.) + K_c[e(t) + T_d \frac{de(t)}{dt}], \tag{D - 9}$$

where the parameters of digital PD controller are calculated as:

$$K_P = K_{PC}, \qquad K_D = K_{DC}/T, \tag{D – 10}$$

and K_{PC} , and K_{DC} are predefined PD controller gains in continuos time domain. The sampling period T is defined as $T = 1/Freq$ and in this case control architecture operates on frequency $Freq = \omega_S = 200$ Hz.

The gravitational terms already divided by the motor parameters for the first three links are:

$$g[0] = 0, \qquad \text{(D - 11)}$$

$$g[1] = (-3.2373\cos(q[1]) - 5.15\cos(q[1] + q[2])) / 1.7677, \qquad \text{(D - 12)}$$

$$g[2] = (-5.15\cos(q[1] + q[2])) / 6.6269, \qquad \text{(D - 13)}$$

where joint angles are:

$$\theta_1 = q[0], \qquad \theta_2 = q[1], \qquad \theta_3 = q[2]. \qquad \text{(D - 14)}$$

The digital filter parameters are calculated as:

$$b[k] = K_P + K_D, \qquad a[k] = -(1 + dpole),$$

$$b[k-1] = -((dpole + 1)K_P + 2K_D, \qquad a[k-1] = dpole,$$

$$b[k-2] = dpoleK_P + K_D, \qquad \text{(D – 15)}$$

where discrete-time pole is defined as,

$$dpole = (1/(2\pi CuttOffFreq))/(T + (1/(2\pi CuttOffFreq))). \qquad \text{(D - 16)}$$

According to Shannon's theorem, the cut-off frequency is chosen not to be larger than half of the sampling frequency:

$$|X(j\omega)| = s, \qquad \forall \omega \geq \frac{\omega_S}{2}. \qquad \text{(D - 17)}$$

Thus, the digital PD + G control signals for the first three axis are calculated as

$$u[0] = g[0] + pd[0], \tag{D-18}$$

$$u[1] = g[1] + pd[1], \tag{D-19}$$

$$u[2] = g[2] + pd[2]. \tag{D-20}$$

The plots are shown in Figures D.13 – D.17.

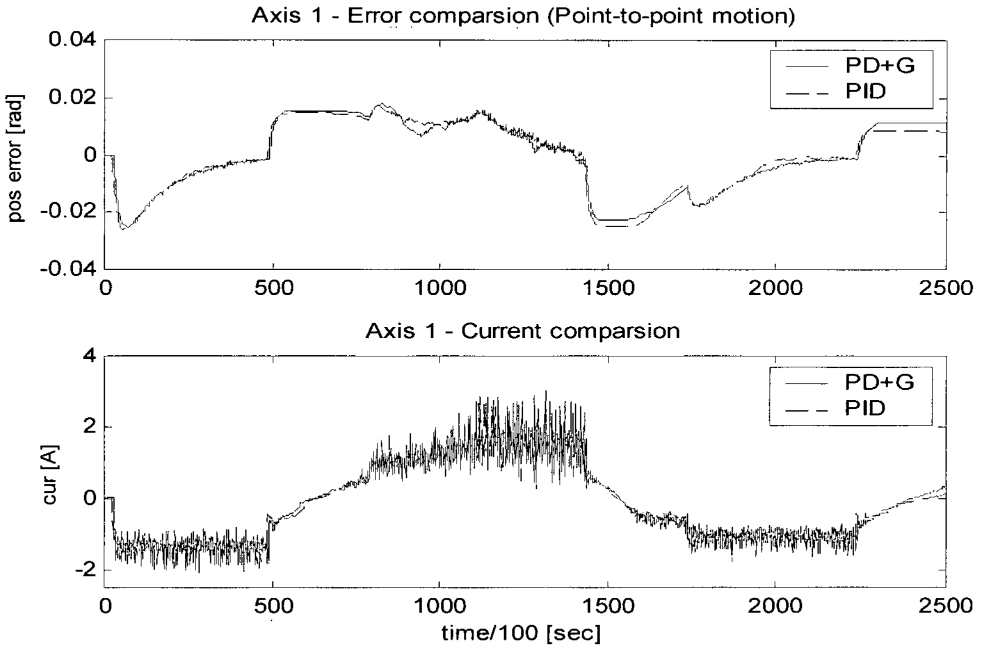


Figure D.13. The first-axis plots for slow robot motion with PD+G and PID controllers.

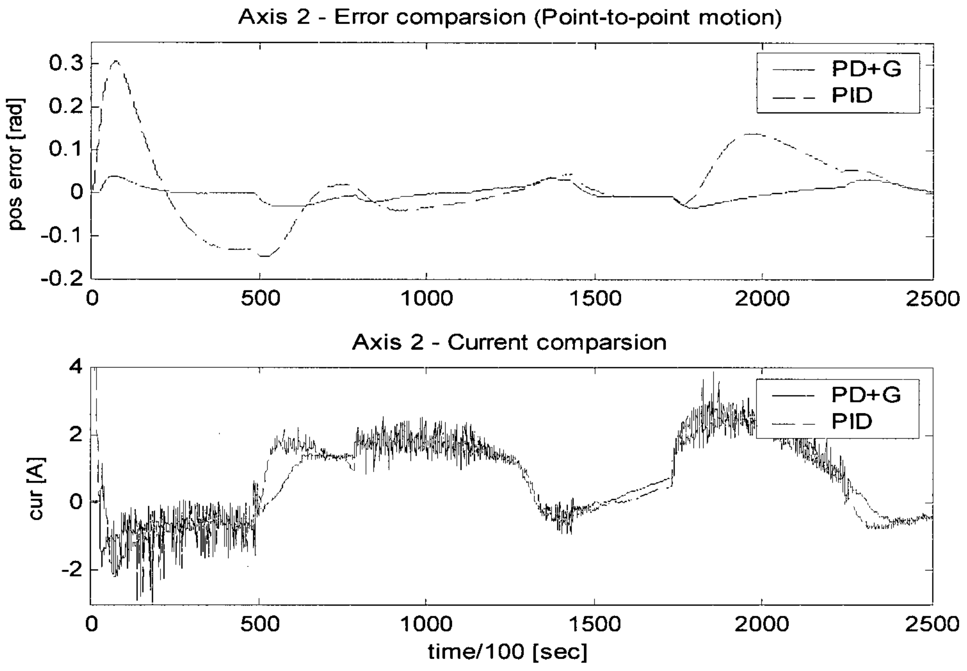


Figure D.14. The second-axis plots for slow robot motion with PD+G and PID controllers.

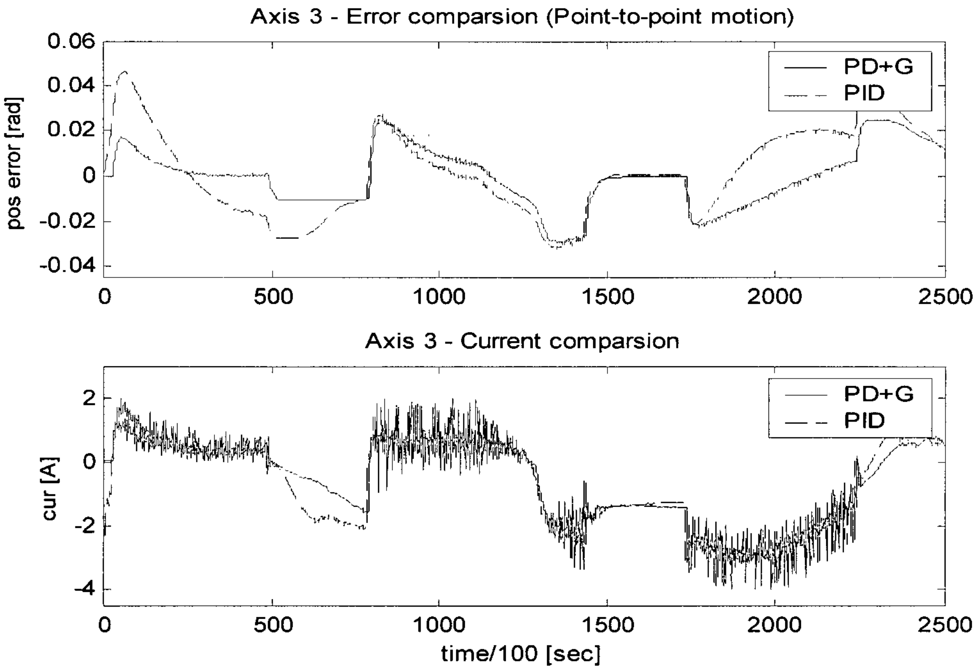


Figure D.15. The third-axis plots for slow robot motion with PD+G and PID controllers.

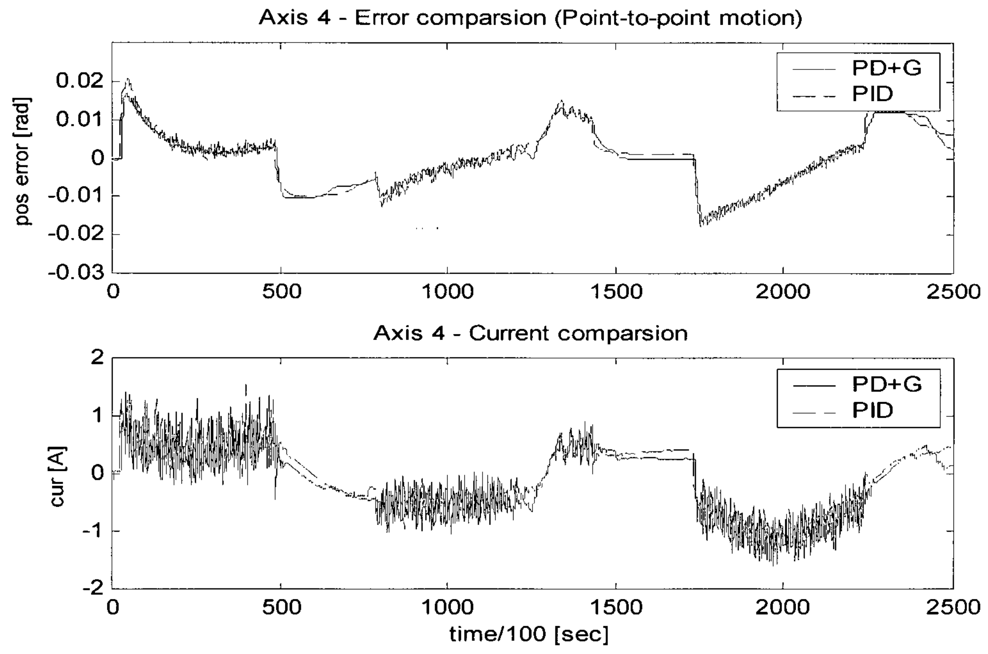


Figure D.16. The fourth-axis plots for slow robot motion with PD+G and PID controllers.

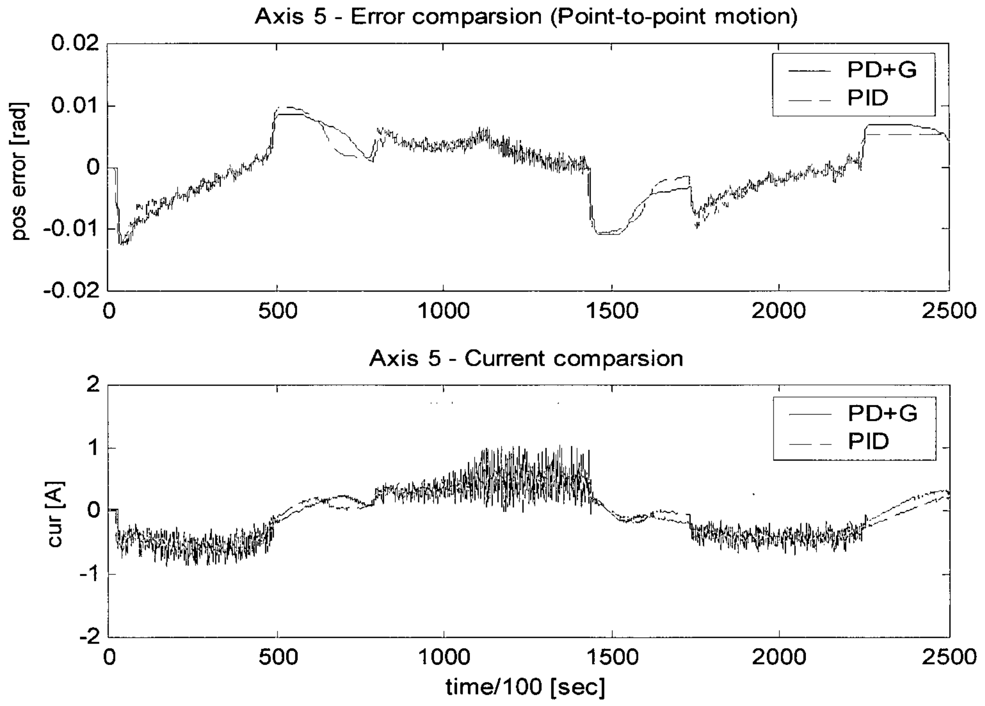


Figure D.17. The fifth-axis plots for slow robot motion with PD+G and PID controllers.

D.3 The Point-to-Point Motion Task with High-Speed Trajectory

The results are given for simple point-to-point robot motion where the high-speed trajectory is given by:

```
PTP_traj( [0.4; -0.2; 0.1], [0.4; 0.4; 0.1], 1, 0.005, 60, 5 )
```

D.3.1 Applied Control Law

The block diagram of computer torque architecture utilized for this task is presented on Figure D.18. The zero-order hold is embedded into the architecture of ORTS. Due to the lack of velocity sensor information the signal from encoder is sent through the designed first order filter and the approximation of velocity signal is made. The low-pass filter is designed as second order system, and the applied control law is presented below.

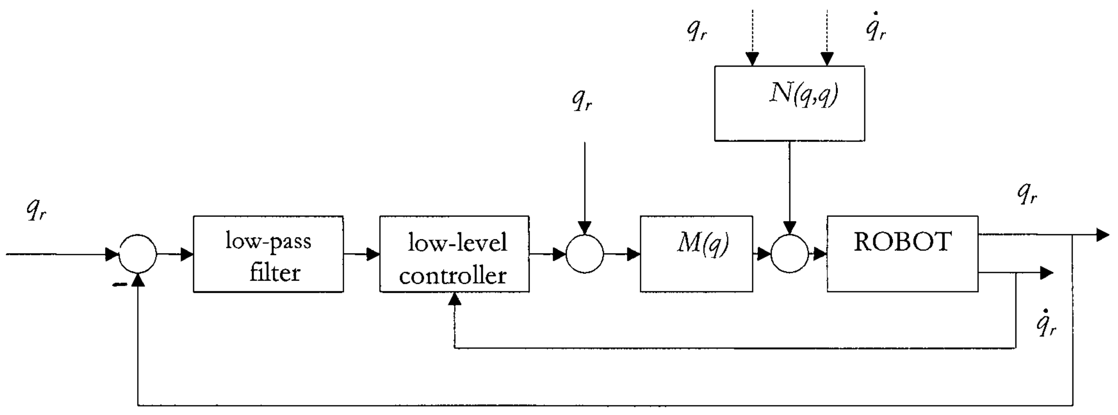


Figure D.18. Computer torque with PID controller.

For this task only first three links of manipulator are considered and controlled. The control signals for first three links are given, respectively:

$$u[0] = n[0] + 0.5657(m[0][0]\ddot{q}_d[0] + m[0][1]\ddot{q}_d[1] + m[0][2]\ddot{q}_d[2]) \\ + m[0][0]pid[0] + m[0][1]pid[1] + 3.7488m[0][1]pid[2], \qquad \text{(D - 21)}$$

$$u[1] = n[1] + 0.5657(m[1][0]\ddot{q}_d[0] + m[1][1]\ddot{q}_d[1] + m[1][2]\ddot{q}_d[2]) \\ + m[1][0]pid[0] + m[1][1]pid[1] + 3.7488m[1][1]pid[2], \qquad \text{(D - 22)}$$

$$u[2] = n[2] + 0.5657(m[2][0]\ddot{q}_d[0] + m[2][1]\ddot{q}_d[1] + m[2][2]\ddot{q}_d[2]) \\ + m[2][0]pid[0] + m[2][1]pid[1] + 3.7488m[2][1]pid[2], \qquad \text{(D - 23)}$$

where nonlinear terms are calculated as:

$$n[0] = (g[0] + temp2)/1.7677, \qquad \text{(D - 24)}$$
$$n[1] = (g[1] + temp2)/1.7677, \qquad \text{(D - 25)}$$
$$n[2] = (g[2] + temp2)/6.6269, \qquad \text{(D - 26)}$$

where

$$temp2 = \dot{q}[0][0]temp[0][0] + \dot{q}[0][1]temp[1][0] + \dot{q}[0][2]temp[2][0], \qquad \text{(D - 27)}$$

and $temp[i][j]$ are temporary files for matrix calculations.

The joint velocity estimates are calculated from position measurements using the extended Euler approximation as

$$v_k = vv_{k-1} + (q_k - q_{q-1})/T, \qquad \text{(D - 28)}$$

that leads to

$$temp\,\dot{q}[i][0] = 0.1temp\,\dot{q}[i][1] + ((tempq[i][0] - temp[i][1])/T). \qquad \text{(D - 29)}$$

The plots are given on Figures D.19 – D.21.

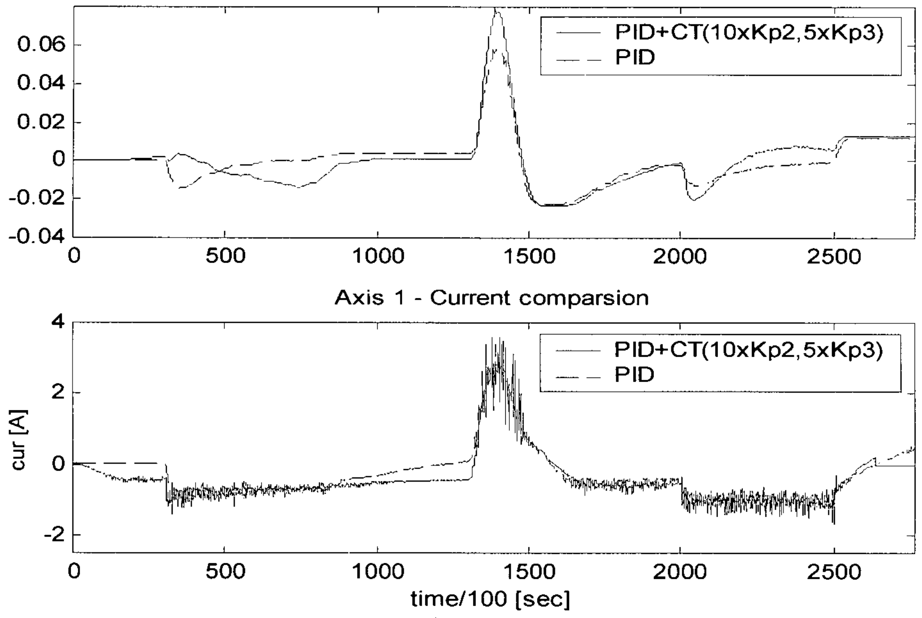


Figure D.19. The first-axis plots for fast robot motion with PID+CT and PID controllers.

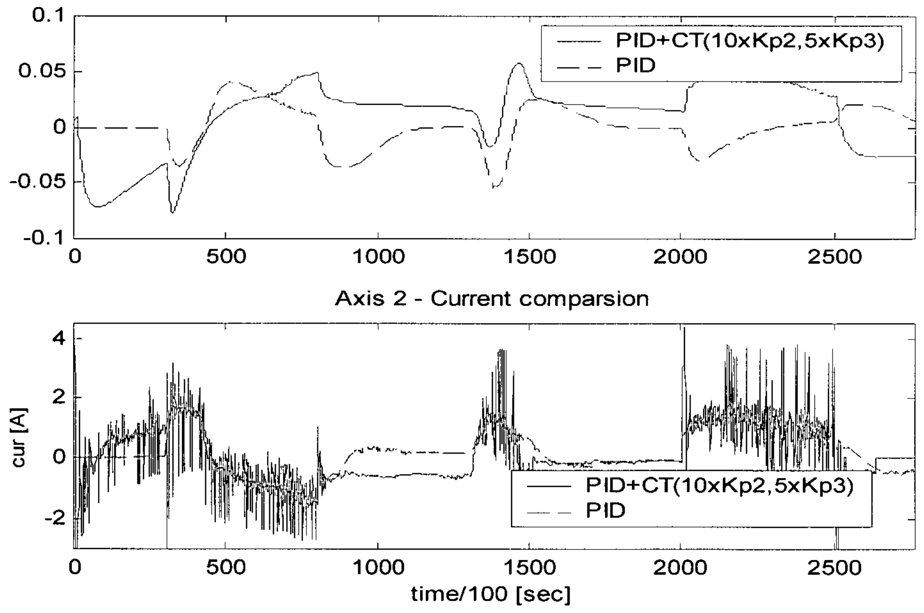


Figure D.20. The second-axis plots for fast robot motion with PID+CT and PID controllers.

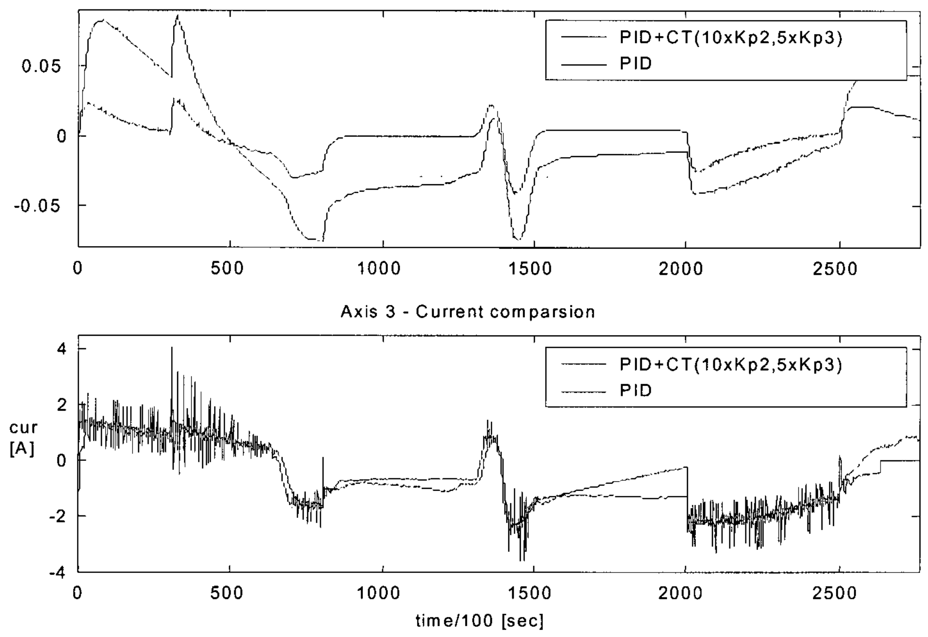


Figure D.21. The third-axis plots for fast robot motion with PID+CT and PID controllers.

Appendix E

Proposed Tuning System

In this appendix, a controller tuning system that can be implemented on-line is proposed. The system is a gain tuner for PID controller. However, using this approach, gain tuners for other commonly used controllers can be implemented.

E.1 Criteria for Tuning Module

The **tuning module** is a controller gain-tuner, which is designed as a fuzzy-logic-based controller that monitors robot joint response characteristics and simultaneously modifies the gains to provide better responses for large deviations of the monitored quantities. Separate tuning modules are indicated depending upon the type of controller selected. For example, through the design of, the PID controller gains, one might take into consideration the following issues:

- larger values of gains are needed to improve error convergence
- too large gain values may cause resonance oscillations
- feedback gains amplify not only error signals, but noise as well
- too large integral gain can produce a big overshoot in response (undesirable for precise positioning).

Thus, the actual controller design is a compromise solution that is a consequence of different criteria imposed by user as well as system feasibility to maintain the stability. The main idea, in accordance with system response, is that only one of the local gains (e.g. proportional gain) is directly changed, and the values of derivative and integral gains (if exist), are changed simultaneously so that stability of the whole system preserved.

For this system, simple criteria for synthesizing gain-tuning-rules are used:

- if the response errors are large and do not show a tendency to decrease, the proportional gain is increased to make faster error convergence

- if the response errors are large but show significant convergence to desired level, the proportional gain is gradually decreased to the value that is appropriate for small-error conditions [17].

- if the errors are small, the proportional gain is decreased to prevent resonance oscillations and attenuate negative noise effects.

E.2 Membership Functions

In order to express these criteria that characterize this module for gain tuning mathematically, triangle-shaped membership functions are adopted with fuzzy intervals, labeled as *NEG, ZER,* and *POS,* Figure E.1. The fuzzy set comprises 27 rules and is 'open' in form so it can be adapted according to users criteria.

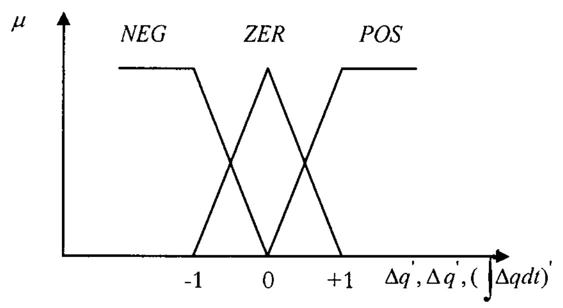


Figure E.1. Fuzzy sets for PID tuning.

For example, the rule:

$$\textbf{if } \Delta q' \textbf{ is } NEG \textbf{ and } \Delta \dot{q}' \textbf{ is } POS \textbf{ and } (\int \Delta q dt)' \textbf{ is } POS \textbf{ then } \xi \textbf{ is } ZER$$

has the effect of changing the proportional gain as a function of response errors. The inputs $\Delta q, \Delta \dot{q}$, and $(\int \Delta q dt)$ are subjected to a nonlinear transformation to obtained normalized values $\Delta q', \Delta \dot{q}', (\int \Delta q dt)'$, in the closed interval [-1, +1], and ξ is a normalized rate of change in proportional gain.

As outlined in the recommendations, implementation of this tuner is the next step to implementation to the TREX project.