République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département de Génie Electrique
Laboratoire de **C**ommande des **P**rocessus

Spécialité : **Automatique**
Option : **Systèmes Intelligents de Commande et Robotique**

Mémoire de Magister

Présenté par : **OUBBATI FARID**
Ingénieur d'Etat en Electronique de l'Université de Boumerdes

THEME

Commande d'un bras de Robot Manipulateur à l'aide d'une carte dSPACE - ds1104

Devant le Jury:

Président :	F.BOUDJEMA	Professeur, ENP
Rapporteur :	O.STIHI	Chargé de Cours, ENP
Examinateurs :	M.TADJINE	Maître de Conférences, ENP
	H.CHEKIREB	Maître de Conférences, ENP
	R.ILLOUL	Chargé de Cours, ENP

Mémoire préparée au sein du **L**aboratoire de **C**ommande des **P**rocessus de l'ENP
10, avenue Hassen Badi BP 182 El-Harrach 16200 Alger

DEDICATION

For my family, for my friends.

ACKNOWLEDGMENTS

I would like to thank my adviser, Mr **Omar Stihi**, for his support and for his patience in continuously supervising me to accomplish this work.

I would also like to thank, Mr R.ILLOUL, Mr M.TADJINE and Mr H.CHEKIREB for accepting to be members of my examination committee and Professor F.BOUDJEMA for accepting to be its chairman and finally my gratitude goes to all the staff of the **P**rocesses **C**ontrol **L**aboratory.

ABSTRACT

ملخص

الهدف من هذا العمل كان تصميم و استعمال نظم تحكم في اطاربطاقة آلية للتحكم DSP للسيطرة على ذراع آلي صنع خصيصا لهذا و بمحركات تعمل بالتيار الكهربائي المستمر . العمل بدا بإيجاد نموذج رياضي للذراع الآلي ، تركيب و دراسة البطاقة الآلية للتحكم dSPACE ds1104 الحديثة الصنع, و في النهاية تجربة نظم التحكم المختارة (نظام التحكم التناسبي المشتق - PD، نظام التحكم التناسبي المشتق المكمل -PID، نظام التحكم بالمزدوجة المحسوبة - CTC) مع الذراع الآلي.

كلمات المفتاح : نظم تحكم بالذراع الآلي، البطاقة الآلية للتحكم DSP، PD/PID، نظام التحكم بالمزدوجة المحسوبة CTC.

Résumé

L'objective de ce travail était d'implémenter et tester des algorithmes de commande en utilisant une carte de commande DSP en un manipulateur de type puma construit spécialement pour ça et motorisé par des moteurs a courant continus. Le travail a commencé par modéliser la structure mécanique du manipulateur construit, installer et étudier la carte de commande dSPACE ds1104 qui est un matériel nouvellement acquis, et finalement, implémenter et tester les algorithmes de commande sélectionnés (PD, PID, Commande du Couple Calculé) avec notre manipulateur.

Mots clés : commande de robot manipulateur, Contrôleur DSP, PD/PID, Commande du Couple Calculé.

Abstract

The objective of this work was to implement and test a DSP based control algorithms on a puma type manipulator mechanical structure built especially for this purpose and derived by DC Motors. The work Started by modelling the built manipulator, installing and studying the dSPACE ds1104 Controller Board which is a newly acquired material, and finally, implementing and testing the chosen control algorithms (PD, PID, Computed Torque Control) with our manipulator.

Key words: Robot Manipulator Control, DSP Controller, PD/PID, Computed Torque Control.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

GENERAL INTRODUCTION

The Robotics Institute of America defines a robot as follows:

A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.

The robot can be defined more practically as a computer-controlled device that combines the technology of digital computers with the technology of servo-control of articulated chains. It should be easily reprogrammed to perform a variety of tasks, and must have sensors that enable it to react and adapt to changing conditions. Most industrial robots satisfy this definition. They basically serve to eliminate the need of high cost, specialized equipment in the manufacturing industry.

Robots are extensively used in the manufacturing of automobiles, electronic goods and semiconductors. The automotive industry "employs" 20% of all robots. Robots are also used for small volume flexible manufacturing systems for video-cameras, walkmans, and personal computers. However, there are applications in hostile environments in which it is necessary to use robots (for example, in space, nuclear plants) or it is too dangerous to use humans (for example, military operations). There are others where the physical task demands skills that humans simply do not have (for example, surgery).

The objective of the work presented here was first to study, design and build a mechanical structure representing a 3dof manipulator having the capability to perform 3D trajectories "imposed and free" after that, the job consisted of designing and implementing some control algorithms that will be tested on our manipulator. The control algorithms implementation support is a newly acquired DSP Card "ds1104" which is one of the new generation controller cards characterized mainly by a huge computation performance, the ease of use, and the wide range of applications on which it may be applied.

In chapter I, we have introduced the basic theoretical background and definitions necessary to understand study and control robotic mechanisms and which will be the base in the modelling and the development of our mechanical structure and its control.

In chapter II, since the DSP Card is a new acquired card and it has not been used before, the work first was to install it and test its functioning with simple real time applications in order to be familiar with the development environment offered by the DSP Card and the development as well as the implementation steps necessary to follow in order to use efficiently the DSP Controller Card. The first part of chapter II illustrates the hardware parts composing the DSP Card and there characteristics, the second part introduces the software platform consisting of a seamless combination of Matlab "Simulink and dSPACE Real Time Interface (RTI)" on which we design our applications, the Real Time Workshop (RTW) which will enable the compilation of the simulink models into a **C** code and the download of these applications to the DSP Card for real time running. This process is extremely time-efficient since it allows the user to use a high level block diagram simulation program like Simulink instead of spending hours on coding and debugging in a lower level language. Visualization and analysis of results can be conducted in dSPACE ControlDesk "graphical user interface software" that allows graphing, data capturing and real-time parameter tuning. Another advantage of ControlDesk is the compatibility of its captured data with Matlab.

Chapter III, based on the theory given in chapter I we have derived the manipulator model from the homogenous transforms to the equations of motion and on which we will test and simulate the control algorithms.

In chapter IV, in the first part we will present our simulation results of the tested control algorithms on the manipulator model including the motors dynamics and using the same simulation parameters settings supported by the DSP Card. In these simulations; various types of trajectories will be used with different time lengths, different sampling periods and for some simulations we will use a voltage control signal in a PWM form to be as close as possible to the real time results. In the second part, we will present the real time application results of some control algorithms, with chosen “free and imposed” trajectories, captured directly from the DSP Card and treated under Matlab.

At the end of the thesis, two appendices will be attached; the first one represents some mathematical theory necessary for chapter I, the second one is the structural diagram of the hardware used for the real time application and contains also the websites where we can find their datasheets.

CHAPTER I
ROBOTIC MECHANISMS

I.1 Overview of Robotic Mechanisms

A robot system generally consists of three subsystems **Figure 1.1**:

- The Motion subsystem: the physical structure that carries out desired motions.
- The recognition subsystem: it uses various sensors to gather information about any objects being acted upon, about the robot itself, and about the environment.
- The control subsystem: it influences the motion subsystem to achieve a given task using the information from the recognition subsystem [1].

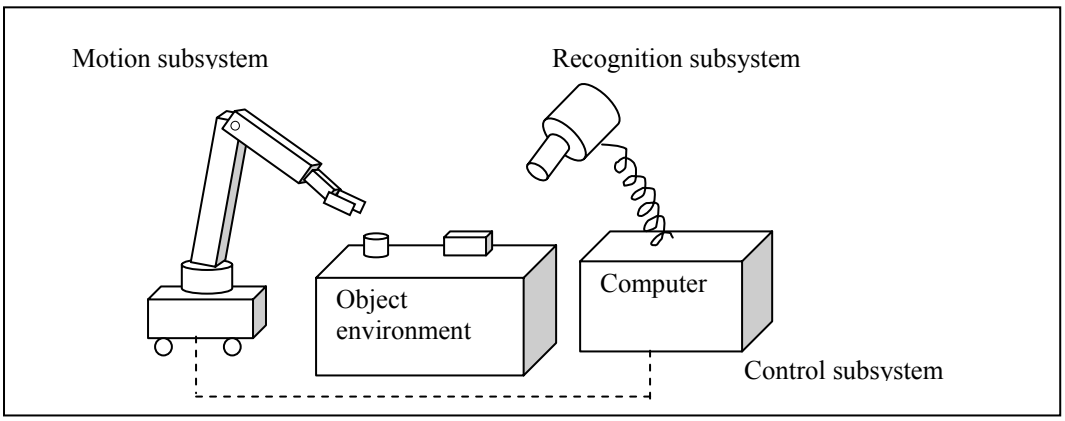


Figure 1.1: A Robot system.

Most robot manipulators can be regarded as open–loop link mechanisms consisting of several links connected together by joints. Typical joints can be **Revolute joints** or **Prismatic joints** which are represented by the symbols shown in **Figure 1.2**. Driving these joints with appropriate actuators moves the end point of the mechanism. A manipulator can usually be divided into an arm portion, a wrist portion, and a hand portion.

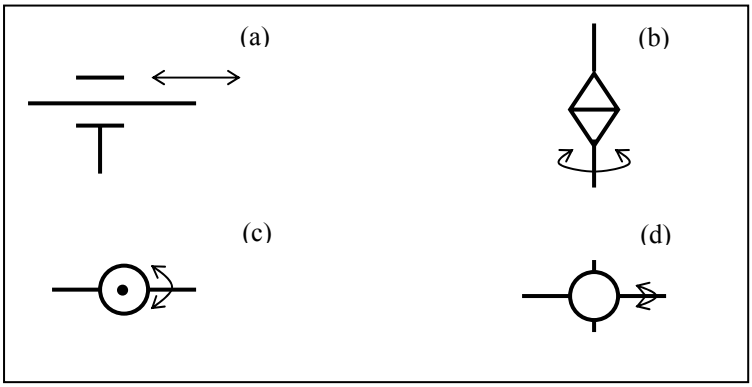


Figure 1.2: Symbols of joints (arrows show direction of motion):
(a) Prismatic joint **(b)** Revolute joint **(c)** Up-and-down rotation
(d) Back-and-forth rotation.

The two basic and typical mechanisms that exist and from which almost all the other mechanisms can be derived are: the **PUMA** type also called (the vertical multi-joint type) and the **SCARA** type also called (the horizontal multi-joint type) manipulators **Figures 1.3a** and **1.3b**. Each of the mechanisms in **Figure 1.3** has three degrees of freedom, which is the

minimum number of degrees of freedom needed for placing the end-point of the arm at an arbitrary point in the three-dimensional space. Here the **degree of freedom** is defined as the minimal number of position variables necessary for completely specifying the configuration of a mechanism.

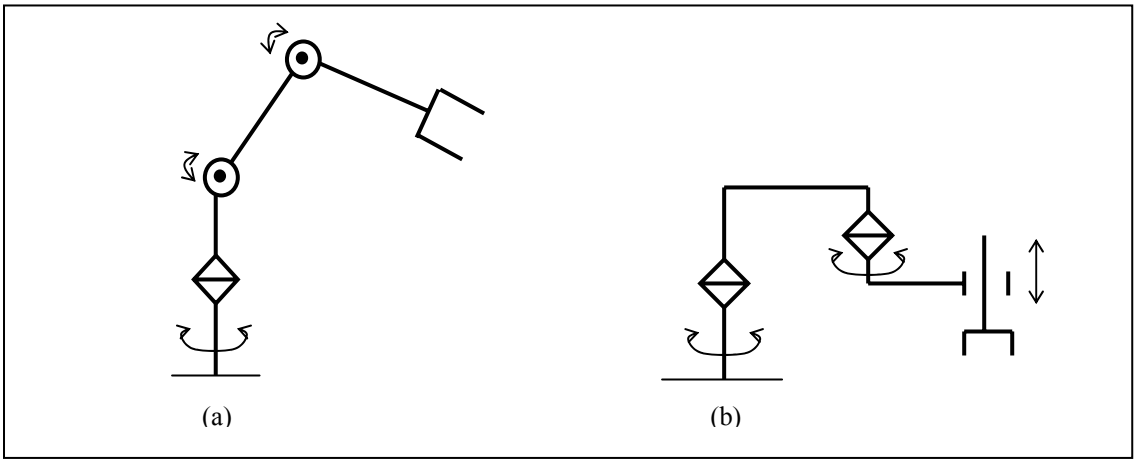


Figure 1.3: Two different mechanisms. (a) Puma type manipulator (b) Scara type manipulator.

The wrist is connected to the end of the arm portion. The main role of the wrist is to change the orientation of the hand. Examples of wrist mechanisms are shown in **Figure 1.4**.

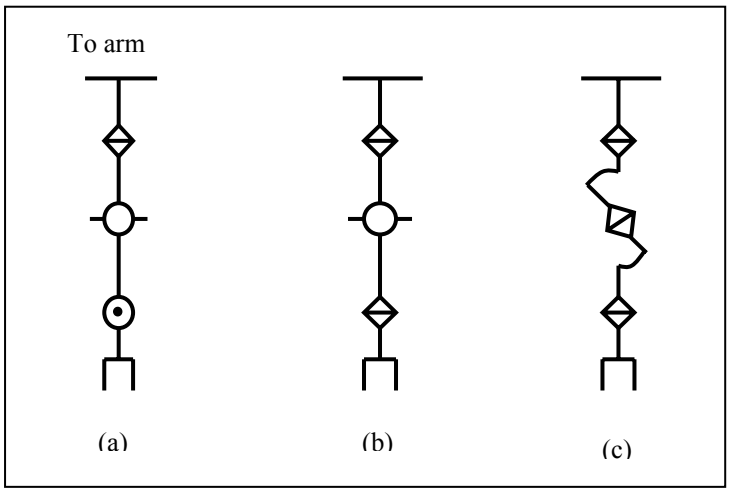


Figure 1.4 : Wrist mechanisms.

I.2 Kinematics

Kinematics of robot manipulators means studying geometrically the motion of the manipulator links and/or objects related to the manipulation task in terms of position, velocity, and acceleration [1].

I.2.1 The Position and Orientation

The first thing we have to do in order to analyse a manipulator is to represent mathematically the position and orientation of the manipulator it self (its joints), the tool it holds, and the objects on which the robot works in the three dimensional space.

To express the position and orientation of an object with respect to a given **reference frame *(A)*** we attach a frame to this object the **object frame *(B)*** so that the object position is the position of the origin of the object frame and the object orientation is the direction of the axes of the object frame all this with respect to the reference frame, **Figure 1.5**.

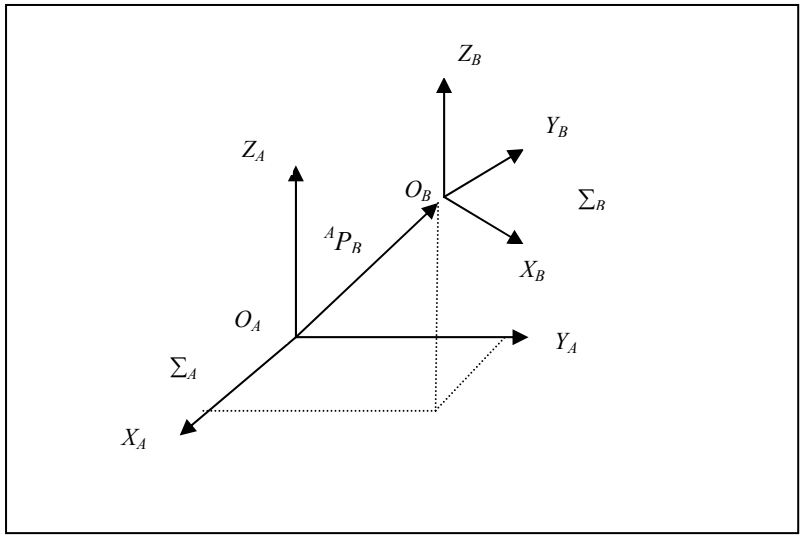


Figure 1.5: Reference frame and object frame.

${}^A p_B$ is the vector from O_A to O_B (i.e., **the position vector** of O_B relative to O_A) expressed in [1]$\sum_A$. the unit vectors in the directions of X_B,Y_B, and Z_B, expressed in $\sum_A$ are denoted ${}^A x_B$, ${}^A y_B$, and ${}^A z_B$. Then, the position of the object *(B)* is represented by ${}^A p_B$, and its orientation is represented by $\{{}^A x_B, {}^A y_B, {}^A z_B\}$.The subscript A on the left means that the vectors are expressed in the frame $\sum_A$.

The orientation is usually defined as a matrix: ${}^A R_B = [{}^A x_B \; {}^A y_B \; {}^A z_B]$ and can be regarded as describing the rotational part of the relative displacement of frame $\sum_B$ from frame $\sum_A$, this matrix is called the **Rotation matrix**.

Other ways exist for representing the orientation of an object like the **Euler angles** and **Roll-pitch-yaw angles** which regard the orientation as a result of three sequential rotations about some fixed axes from the reference frame $\sum_A$, and represent the orientation of frame $\sum_B$ by a set of three rotational angles.

I.2.2 Coordinate Transformation

I.2.2.1 The Homogenous Transform

Suppose that the relation between two coordinate frames $\sum_A$ and $\sum_B$ is given by the position vector ${}^A p_B$ and the rotation matrix ${}^A R_B$ of $\sum_B$ with respect to $\sum_A$. then the relation between the

[1] Hereafter, all the frames denoted by "$\sum$ " are right-hand orthogonal coordinate frames.

position expressions, ${}^{A}r$ and ${}^{B}r$, of a point in space by (respectively) frame $\sum_A$ and $\sum_B$, **Figure 1.6**, is:

$$ {}^{A}r = {}^{A}R_B \, {}^{B}r + {}^{A}p_B \qquad (I\text{-}1) $$

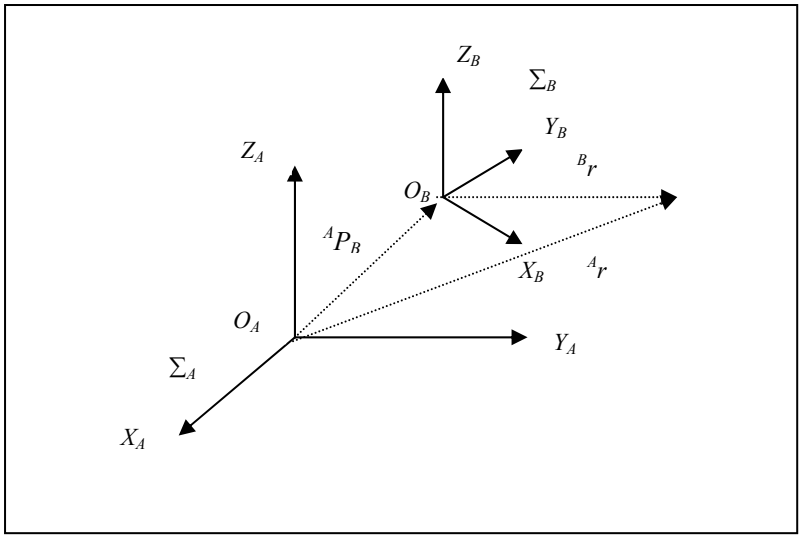


Figure 1.6: Frame $\sum_A$ and $\sum_B$.

equation *(I-1)* can also be expressed as:

$$ \begin{bmatrix} {}^{A}r \\ \hline 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}R_B & & {}^{A}p_B \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{A}r \\ \hline 1 \end{bmatrix} = {}^{A}T_B \begin{bmatrix} {}^{B}r \\ \hline 1 \end{bmatrix} \qquad (I\text{-}2) $$

In this new expression, the three-dimensional vectors ${}^{A}r$ and ${}^{B}r$ must be represented by the four-dimensional vectors obtained by adding the element 1 at the bottom of the original vectors. In return for this, the expression obtained is simpler in the sense that the transformation from the expression of a vector with respect to $\sum_B$ to that with respect to $\sum_A$ is done by the multiplication of just one matrix: ${}^{A}T_B$, this 4x4 matrix is called the **Homogenous Transform**. The vector
$[{}^{A}r^{T} \ , \ 1]^{T}$ may also be written as ${}^{A}r$ when there is no confusion; so equation *(I-2)* can be written as:

$$ {}^{A}r = {}^{A}T_B \ {}^{B}r \qquad (I\text{-}3) $$

The homogenous transform can also be used to describe the relation between two coordinate frames.

I.2.2.2 Product and Inverse of Homogenous Transform

Consider three frames, $\sum_A$, $\sum_B$, and $\sum_C$, and assume that the relations between $\sum_A$ and $\sum_B$ and between $\sum_B$ and $\sum_C$ are given by ${}^{A}T_B$ and ${}^{B}T_C$ respectively. Then the relation ${}^{A}T_C$ between $\sum_A$ and $\sum_C$ is given by the product of ${}^{A}T_B$ and ${}^{B}T_C$:

$$ {}^{A}T_C = {}^{A}T_B \ {}^{B}T_C \qquad (I\text{-}4) $$

the inverse of ${}^{A}T_{B}$ is:

$$ {}^{A}T_{B}^{-1} = {}^{B}T_{A} \qquad (I\text{-}5) $$

I.2.3 Joint Variables and the End-Effector Position

I.2.3.1 General Relations

The position of the end-effector of a manipulator with *(n)* degrees of freedom is determined by the rotational displacement of the revolute joints and the linear displacement of the prismatic joints composing its mechanism.

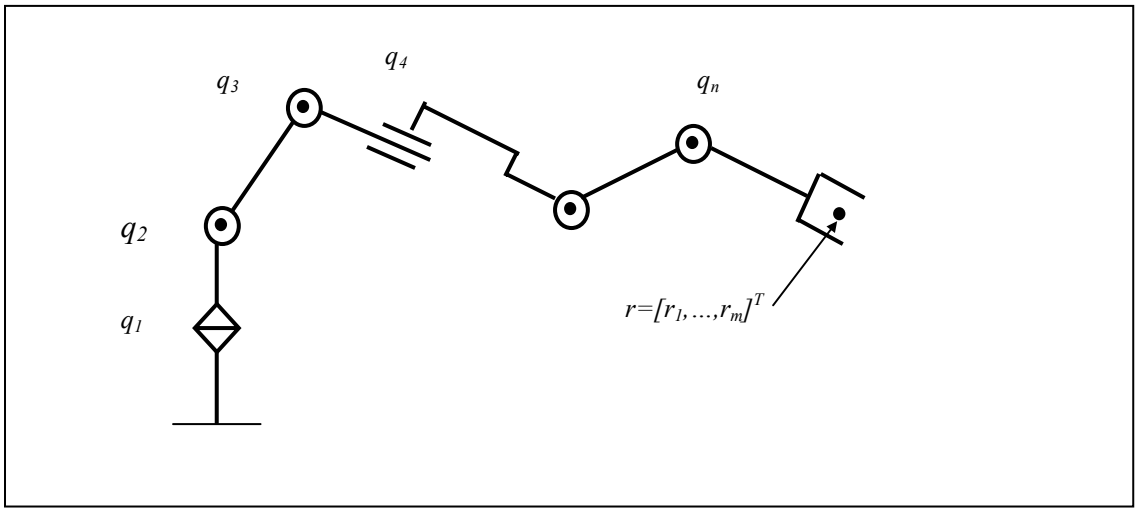


Figure 1.7: *n*-link manipulator.

In **Figure 1.7**, the joints are numbered "*1,2,...,n",* starting from the base of the manipulator. The displacement of joint *i* is denoted q_i and is called the **joint variable**. The collection of joint variables:

$$ \boldsymbol{q} = \left[q_1, q_2, \ldots, q_n \right]^T \qquad (I\text{-}6) $$

is called the **joint vector**. The position of the end-effector is denoted by the *m*-dimensional vector:

$$ \boldsymbol{r} = \left[r_1, r_2, \ldots, r_m \right]^T \qquad (I\text{-}7) $$

where $m \leq n$.

The relation between $\boldsymbol{r}$ and $\boldsymbol{q}$ is determined by the manipulator mechanism and is generally nonlinear and given by:

$$ \boldsymbol{r} = f_r(\boldsymbol{q}) \qquad (I\text{-}8) $$

this equation is called the **kinematics equation** of the manipulator.

In general, when some task is assigned to the manipulator, it is the end-effector position or the trajectory which is given and we have to calculate the joint vector $\boldsymbol{q}$ to realize this trajectory thus, we have to obtain $\boldsymbol{q}$ which satisfies the following:

$$ \boldsymbol{q} = f_r^{-1}(\boldsymbol{r}) \qquad (I\text{-}9) $$

Note that ***q*** may not exist or be not unique. The problem of obtaining ***r*** for a given ***q*** is called the **Direct Kinematics Problem (*DKP*)**, and that of obtaining ***q*** for a given ***r*** is called the **Inverse Kinematics Problem (*IKP*)**. It is rather easy to solve the direct kinematics problem and the inverse kinematics problem when we are dealing with small degrees of freedom manipulators; however, both problems become increasingly difficult as the number of degrees of freedom increases. One way to cope with this difficulty is to assign an appropriate coordinate frame to each link (which relates two consecutive joints) and to describe the relations among the links by the relations among these frames; this will be illustrated in the next subsection.

I.2.3.2 Link Parameters

Typical robots are **serial-link** manipulators comprising a set of bodies called links in a chain connected by joints. Each joint has one degree of freedom either translational or rotational.

For a manipulator with n joints numbered from *1* to n, there are $n+1$ links, numbered from *0* to n, link *0* is the base of the manipulator, generally fixed, and link n carries the end-effector, joint i connects links i and $i-1$.

For each joint i, the joint axes is defined as the rotational axis in the case of a revolute joint or as an arbitrary straight line parallel to the direction of translation in the case of a prismatic joint.

Now, a link can be specified by two numbers, the ***link length*** a_i of the common normal between joint axes i and $i+1$ (when the joints axes i and $i+1$ are parallel, the common normal is not unique, so it is selected arbitrarily) and the ***link twist angle*** α_i between the orthogonal projections of joint axes i and $i+1$ onto a plane normal to the common normal.

The relative positional relation between links $i-1$ and i at joint i can be described by the distance d_i between the feet of two common normals on the joint axis i, d_i is called the **joint length**. The angle θ_i between the orthogonal projection of these common normals to a plane normal to the joint axis i is called the **joint angle.** If joint i is revolute d_i is constant and θ_i expresses the rotational angle of the joint; if joint i is prismatic θ_i is constant and d_i expresses the translational distance of the joint. Hence, when joint i is revolute we adopt θ_i as the joint variable q_i, and when joint i is prismatic we adopt d_i. This way of describing link mechanisms using a_i, α_i, d_i, and θ_i is usually called the **Denavit-Hartenberg notation, Figure 1.8**. The link parameters for the first and last links are meaningless and are arbitrarily chosen to be *0*.

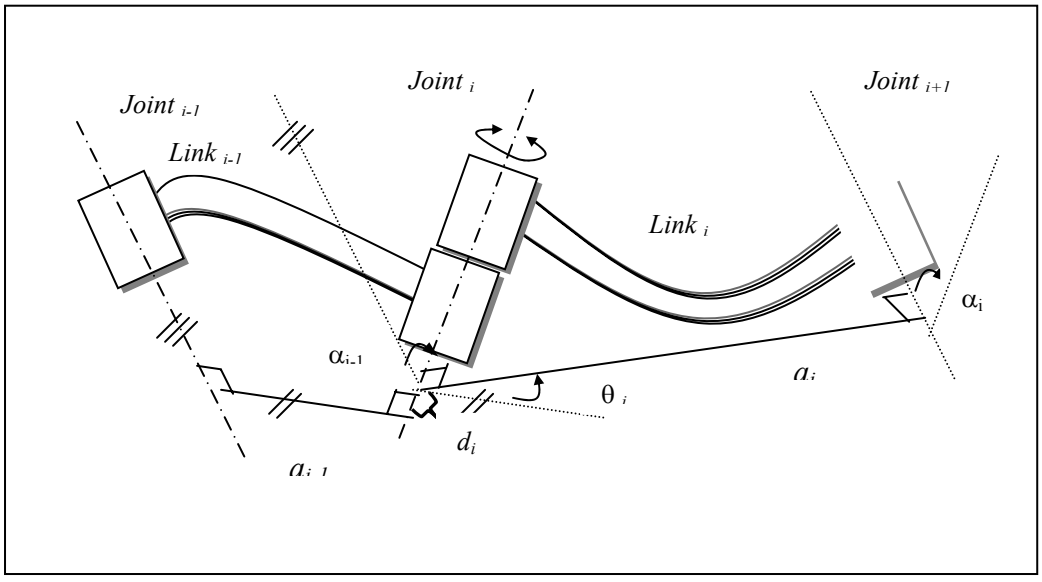


Figure 1.8: Joint axes and link parameters.

I.2.3.3 Link Frames

Now, after defining the link parameters, we will define the coordinate frames one attached to each link, the origin of the coordinate frame $\sum_i$ of link i is set at the end point of the common normal on joint axis i, the Z axis of $\sum_i$, denoted Z_i is selected to be in such a way that it aligns with joint axis i, the X axis of $\sum_i$, X_i is selected so that it is on the common normal and points from joint i to joint $i+1$, **Figure 1.9**, for link 0, the link frame $\sum_0$ is defined to be equal to $\sum_1$, for link n, the origin of $\sum_n$ is set at the end point of the common normal on joint axis n, the axis Z_n is aligned with the joint axis n, the axis X_n is aligned with X_{n-1}.

Now, once we define all the link frames $\sum_i$ of a mechanism, the four variables discussed earlier can expressed a follows:

a_i : the distance measured along the X_i axis from Z_i to Z_{i+1},

α_i : the angle measured clockwise about the X_i axis from Z_i to Z_{i+1},

d_i : the distance measured along the Z_i axis from X_{i-1} to X_i,

and

θ_i : the angle measured clockwise about the Z_i axis from X_{i-1} to X_i.

In other words, the frame $\sum_i$ can be obtained from $\sum_{i-1}$ by the following four transformations:

$$^{i-1}T_i = T_T(X_{i-1}, a_{i-1})\, T_R(X_{i-1}, \alpha_{i-1}) T_T(Z_i, d_i) T_R(Z_i, \theta_i) \qquad (1\text{-}10)$$

where $T_T(X, a)$ denotes a translation along the X axis for a distance a, and $T_R(X,\alpha)$ denotes a rotation about the X axis by an angleα.

Note that two differing methodologies have been established for assigning coordinate frames, each of which allows some freedom in the actual coordinate frame attachment:

- Frame i has its origin along the axis of joint $i+1$, as described by Paul's Robot Manipulators.
- The scheme described above commonly called the modified 'Denavit-Hartenberg' form.

In general, it is desirable to make as many link parameters zero as we can, since this will make the later analysis easier and will decrease the amount of computation necessary to solve the direct and inverse kinematics problems.

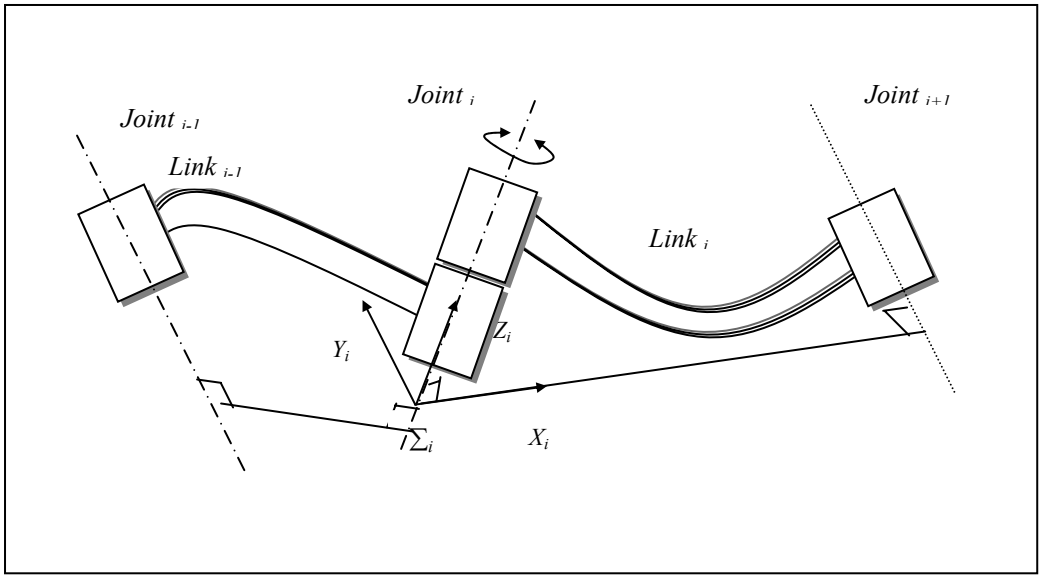


Figure 1.9: Link frame $\sum_i$.

I.2.4 Solution to the Direct Kinematics Problem

From equation *(*I-*4)*, the homogenous transform relating $\sum_n$ to$\sum_0$ is ${}^0T_n$ given by:

$$ {}^0T_n = {}^0T_1 \ {}^1T_2 \ \dots \ {}^{n-1}T_n \qquad (\text{I-}11) $$

When the values of all link parameters are given, ${}^{i-1}T_i$ is a function of q_i only; hence, ${}^0T_n$ is a function of the joint vector $\boldsymbol{q}$.

Let $\sum_E$ be a coordinate frame attached to the end-effector that is attached to link n, and let ${}^nT_E$ be the homogenous transform between the frames $\sum_E$ and $\sum_n$. let $\sum_R$ be the reference frame, and let ${}^RT_0$ describe the relation between $\sum_R$ and the base frame $\sum_0$, the transforms ${}^RT_0$ and ${}^nT_E$ are constant depending only on the location of the base with respect to the reference frame and the way the end-effector is attached to link n, so the relation between the end-effector and the reference frame is:

$$ {}^RT_E = {}^RT_0 \ {}^0T_n \ {}^nT_E \qquad (\text{I-}12) $$

Since the vector $\boldsymbol{r}$ of the end-effector is uniquely determined from ${}^RT_E$ and since ${}^0T_n$ is a function of $\boldsymbol{q}$, we can obtain $f_r(\boldsymbol{q})$ of equation *(*I-*8)* from equation *(*I-*12)* and the direct kinematics problem is solved, the illustration is given in the application.

I.2.5 Solution to the Inverse Kinematics Problem

That is the problem of finding the joint vector $\boldsymbol{q}$ that realizes a given value of the end-effector position vector $\boldsymbol{r}$.

Two major approaches have been developed to solve this problem and obtain a closed form solution; first, the algebraic approach which finds $\boldsymbol{q}$ by various algebraic transformations of equation *(*I-*12)* the second approach is a geometric approach where $\boldsymbol{q}$ is obtained by successive projection of the arm on the coordinate frames attached to its joints and forming equations to be solved. Note that we have to be careful to examine every possible solution.

Here we will consider only the algebraic approach: as an example, let's take six degrees of freedom manipulator whose direct kinematics equation is given by:

$$ {}^0r = {}^0T_6 \ {}^6r \qquad (\text{I-}13) $$

where ${}^0\boldsymbol{r}$ is the end-effector location (position and orientation) with respect to the reference frame and ${}^6\boldsymbol{r}$ is the location of the end-effector with respect to the last joint frame to which the end-effector is attached $\sum_6$. From equation *(*I-*11)* ${}^0T_6$ can be expressed as:

$$ {}^0T_6 = {}^0T_1 \ {}^1T_2 \ {}^2T_3 \ {}^3T_4 \ {}^4T_5 \ {}^5T_6 \qquad (\text{I-}14) $$

which is equivalent to twelve simultaneous algebraic equations with six unknown variables ($q_1, q_2, \dots, q_6$) that can be obtained after transforming equation *(*I-*14)* into:

$$ [\ {}^0T_1(q_1) \dots {}^{i-1}T_i(q_i)]^{-1} \ {}^0T_6 \ [\ {}^jT_{j+1}(q_{j+1}) \dots {}^5T_6(q_6)]^{-1} = {}^i T_{i+1}(q_{i+1}) \dots {}^{j-1}T_j(q_j),\ i < j \qquad (\text{I-}15) $$

multiplying both sides by ${}^6r$ and selecting simpler ones among these equations, we can obtain a solution rather easily; the illustration is given in the application.

I.2.6 The Jacobian Matrix

Using the same principle used to express the position and orientation of the object frame $\sum_B$ with respect to the reference frame $\sum_A$, we can express The translational velocity as the time derivative of the position vector ${}^A p_B$ of $\sum_B$ with respect to $\sum_A$ and is denoted ${}^A \dot{p}_B = d\,{}^A p_B / dt$.

For the rotational velocity of $\sum_B$ with respect to $\sum_A$, two methods exist for expressing it:

- The time derivative of the orientation vector ${}^A\phi_B$ which consists of three variables (e.g., Euler angles, or roll, pitch, and yaw angles), and is denoted ${}^A\dot{\phi}_B = d\,{}^A\phi_B / dt$.
- A vector ${}^A\omega_B$ having the same direction as the instantaneous axis of rotation and a magnitude proportional to the rotational speed about this axis.

The Jacobian matrix is used to express the relation between the end-effector velocity (translational and rotational velocity) and the joint velocity of a manipulator in a compact form.

Differentiating equation *(*I-*8)* with respect to time yields:

$$\dot{r} = J_r(q)\,\dot{q} \qquad (\text{I-}16)$$

where $J_r(\boldsymbol{q})$ is the Jacobian matrix of $\boldsymbol{r}$ with respect to $\boldsymbol{q}$ and is given by:

$$J_r(q) = \frac{\partial r}{\partial q^T} \qquad (\text{I-}17)$$

$J_r(\boldsymbol{q})$ will be written as J_r, for simplicity.

We replace $J_r(\boldsymbol{q})$ by $J_v(\boldsymbol{q})$ if we use method *2* to express the rotational velocity of the end-effector. A matrix relation between the two Jacobian matrices can be found.

I.2.7 General Expression of the Jacobian Matrix J_v

Here we will derive the general expression of the jacobian matrix J_v based on the relations among relative velocities of three moving frames $\sum_A$, $\sum_B$ and $\sum_C$ (translational and rotational velocities).

take $\sum_A$ as the reference then the following relations hold:

$${}^A p_C = {}^A p_B + {}^A R_B \; {}^B p_{CB} \qquad (\text{I-}18)$$

$${}^A\omega_C = {}^A\omega_B + {}^A R_B \; {}^B\omega_{CB} \qquad (\text{I-}19)$$

where ${}^B p_{CB}$ and ${}^B\omega_{CB}$ represent the position and the angular velocity of frame $\sum_C$ with respect to $\sum_B$ expressed in $\sum_B$. Differentiating equation *(18)* and after some transformation we will have:

$${}^A \dot{p}_C = {}^A \dot{p}_B + {}^A R_B \frac{d\,({}^B p_{CB})}{dt} + {}^A\omega_B \times ({}^A R_B \; {}^B p_{CB}) \qquad (\text{I-}20)$$

where ${}^A \dot{p}_B = d\;{}^A p_B / dt$. thus, the equation *(*I-*19)* is the relation between the rotational velocities of $\sum_B$ and $\sum_C$ with respect to $\sum_A$, and equation *(*I-*20)* is that between the translational velocities. (See reference [1] .p60 for a complete proof).

Now we will use these equations to derive the relations among the link velocities of the serial-link manipulator.

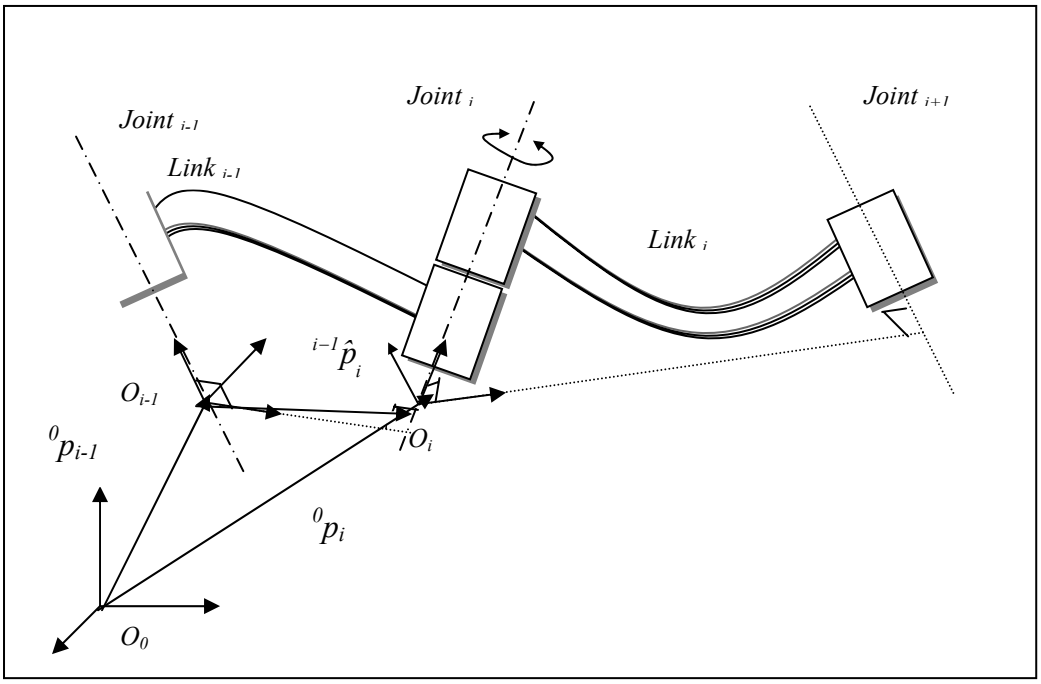


Figure 1.10: Vectors ${}^0p_i$ and ${}^{i-1}\hat{p}_i$.

As shown in **Figure 1.10**:
Consider the vectors: ${}^0p_i$ from $\sum_0$ to $\sum_i$ and ${}^{i-1}\hat{p}_i = [a_{i-1}, -\sin\alpha_{i-1}\ d_i, \cos\alpha_{i-1}\ d_i]^T$ a vector from $\sum_{i-1}$ to $\sum_i$, the following equations describing relative link velocities for the two cases of revolute and prismatic joints can be obtained using equations *(*I*-19)*, *(*I*-20)* after some transformations :

Revolute joint:

$$ {}^0\omega_i = {}^0\omega_{i-1} + {}^0R_i\ e_z\ \dot{q}_i \qquad (I\text{-}21) $$

$$ {}^0\dot{p}_i = {}^0\dot{p}_{i-1} + {}^0\omega_{i-1} \times ({}^0R_{i-1}\ {}^{i-1}\hat{p}_i) \qquad (I\text{-}22) $$

Prismatic joint:

$$ {}^0\omega_i = {}^0\omega_{i-1} \qquad (I\text{-}23) $$

$$ {}^0\dot{p}_i = {}^0\dot{p}_{i-1} + {}^0R_i\ e_z\ \dot{q}_i + {}^0\omega_{i-1} \times ({}^0R_{i-1}\ {}^{i-1}\hat{p}_i) \qquad (I\text{-}24) $$

where ${}^0\omega_i$ and ${}^0\dot{p}_i = d({}^0p_i)/dt$ are the rotational and translation velocities of link i respectively; and $e_z=[0,0,1]^T$, (see reference [1] .p62 for a complete proof).
Based on the previous results, consider the end-effector frame $\sum_E$ and the link frame $\sum_n$ both fixed to link n, using equations *(*I*-19)*, *(*I*-20)* and by the correspondence of $\sum_A \longleftrightarrow \sum_0$, $\sum_B \longleftrightarrow \sum_n$, and $\sum_C \longleftrightarrow \sum_E$ we get :

$$ {}^0\omega_E = {}^0\omega_n, \qquad (I\text{-}25) $$

$$ {}^0\dot{p}_E = {}^0\dot{p}_n + {}^0\omega_n \times ({}^0R_n\ {}^n\hat{p}_E) \qquad (I\text{-}26) $$

from equations *(*I*-21)* and *(*I*-26)* and ${}^0\omega_0 = {}^0\dot{p}_0 = 0$, we can express ${}^0\omega_E$ and ${}^0\dot{p}_E$ as a linear function of $\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n$. From these linear functions we can obtain an expression for J_v. for example, if all n joints are revolute, we have from equations *(*I*-25)*, *(*I*-26)*, *(*I*-21)* and *(*I*-22)*:

$$ {}^0\omega_E = \sum_{i=1}^{n} {}^0R_i \, e_z \dot{q}_i \qquad (I\text{-}27) $$

$$ {}^0\dot{p}_E = \sum_{j=1}^{n} \left[({}^0R_j \, e_z \, \dot{q}_j) \times \left(\sum_{i=j}^{n} {}^0R_i \; {}^i\hat{p}_{i+1} \right) \right] \qquad (I\text{-}28) $$

where ${}^n\hat{p}_{n+1} = {}^n\hat{p}_E$ for notational convenience.
defining ${}^0z_i$ and ${}^0p_{E,j}$ by :

$$ {}^0z_i \triangleq {}^0R_i \; e_z \qquad (I\text{-}29) $$

and

$$ {}^0p_{E,j} = \sum_{i=j}^{n} {}^0R_i \; {}^i\hat{p}_{i+1} = {}^0p_{E,\,j+1} + {}^0R_j \; {}^j\hat{p}_{j+1} \qquad (I\text{-}30) $$

we obtain :

$$ J_v = \begin{bmatrix} {}^0z_1 \times {}^0p_{E,1} & {}^0z_2 \times {}^0p_{E,2} & \ldots & {}^0z_n \times {}^0p_{E,n} \\ {}^0z_1 & {}^0z_2 & .... & {}^0z_n \end{bmatrix} \qquad (I\text{-}31) $$

${}^0z_i$ represents the joint axis *i*, and ${}^0p_{E,i}$ represents the vector from a point on joint axis *i* to the end-effector. For more general manipulators containing both revolute and prismatic joints we have:

$$ J_v = \left[J_{v1}, \, J_{v2}, \, \ldots, \, J_{vn} \right] \qquad (I\text{-}32) $$

where

$$ J_{vi} = \begin{cases} \begin{bmatrix} {}^0z_i \times {}^0p_{E,\,i} \\ {}^0z_i \end{bmatrix} & \text{If joint } i \text{ is revolute,} \\ \begin{bmatrix} {}^0z_i \\ 0 \end{bmatrix} & \text{If joint } i \text{ is prismatic.} \end{cases} \qquad (I\text{-}33) $$

hence we can calculate J_v from ${}^0T_i$ by using the following relations :

$$ {}^0T_i = \begin{bmatrix} {}^0x_i & {}^0y_i & {}^0z_i & | & {}^0p_i \\ ..... & ..... & ..... & ..... & ..... \\ 0 & 0 & 0 & | & 1 \end{bmatrix}, \; i = 1,2, \ldots, n+1, \qquad (I\text{-}34) $$

$$ {}^0p_{E,i} = {}^0p_E - {}^0p_i, \qquad (I\text{-}35) $$

I.2.8 The Singular Configurations

Now, we will consider the problem of obtaining the joint velocity vector $\dot{q}$ needed to realize a given end-effector velocity vector *v*:
for non redundant manipulators and when J_v is non-singular:

$$ \dot{q} = J_v^{-1} \; v \qquad (I\text{-}36) $$

for the case of redundant manipulators $n > 7$ and the rank of $J_v = 6$ we have:

$$\dot{q} = J_v^+ \, v + (I - J_v^+ \, J_v) \, k \qquad (I\text{-}37)$$

where J_v^+ is the pseudo-inverse of J_v *(Appendix 1)* and k is an arbitrary n-dimensional constant
vector.

There are configurations $\boldsymbol{q}$ at which the jacobian matrix becomes singular and thus we cannot obtain its inverse, those configurations are called singular configurations. Generalizing this situation, *we define the singular configurations of a general n-link manipulator as those for which the end-effector velocities (translational, rotational, or their combinations) in a certain direction cannot be realized by any finite joint velocity* [1].

More mathematically speaking, let n' be an integer given by $n' = \max_q \text{rank } J_v(q)$.
If a configuration $q=q_s$ satisfies rank $J_v(q) < n'$ such that if n>6 we have $n'=6$, and if n≤6 we have $n'=n$ then we define q_s as the singular configuration.

Note that if n'=n=6, a necessary and sufficient condition for q to be a singular configuration is:

$$\det J_v(q) = 0 \qquad (I\text{-}38)$$

det J denotes the determinant of a matrix J, therefore, $\dot{q}$ cannot be calculated by the equations *(I-36)* and *(I-37)* at singular configurations. Moreover, we have to ovoid to be too close to a singular configuration since the calculated $\dot{q}$ may become so excessively large that it cannot be realized.

Note that J_v can be replaced by J_r in the definition above but we have to check whether the singular configuration is real or representational and the condition $detJ_r(q)=0$ becomes a necessary condition only.

There are two approaches for coping with the performance deterioration due to singular configurations. One is simply to ovoid using the singular configurations and their neighbouring region by planning the given task carefully. Another is to change the mechanism design. We can, for example, move the singular configuration to an unimportant area of the reachable region by some sophisticated mechanism, or we can add some extra degrees of freedom to the manipulator so that the singular configurations can be avoided by means of the added redundancy.

In addition to that the jacobian matrix is used to express the relation between the end-effector velocity (translational and rotational velocity) and the joint velocity of a manipulator, it can be used to express the relation between joint driving force $\tau = [\tau_1, \tau_2, \ldots, \tau_n]$ and the force ${}^0f_E$ and the moment ${}^0n_E$ applied to the origin of the end-effector frame $\sum_E$ by the relation :

$$\tau = J_v^T \begin{bmatrix} {}^0f_E \\ {}^0n_E \end{bmatrix} \qquad (I\text{-}39)$$

I.2.9 The Jacobian Inverse

Several solutions have been proposed in order to obtain the jacobian inverse:

- The numerical solutions: they can be used to treat the singular as well as the redundant cases but they need a large amount of computation.

- The analytic solutions may be used: it needs a less amount of computation but we have to consider all the singular cases.

There exist solutions which take advantage of the particular form the jacobian matrix in order to reduce the amount of computation, for example: if the jacobian matrix is non-singular of order *n* and has the form:

$$J = \begin{bmatrix} A & 0 \\ B & C \end{bmatrix} \qquad (I\text{-}40)$$

where the matrices A and C are square non-singular matrices, the inverse of the jacobian J can be expressed by:

$$J^{-1} = \begin{bmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{bmatrix} \qquad (I\text{-}41)$$

and the problem reduces to a much more simpler problem, the inversion of smaller order matrices.

I.2.10 The Manipulability

The manipulability of a robot manipulator can be defined as: *the ease of arbitrary changing the position and orientation of the end-effector in a given joint configuration* [1]. And it is an important factor among others like the positioning accuracy, speed, cost …, considered in evaluating manipulators.

I.2.10.1 Manipulability Ellipsoid

Using the jacobian matrix $J_v(q)$ relating the end-effector velocity vector v and the joint velocity $\dot{q}$ (equation *(*I-*16)*).

consider the set of all end-effector v which are realizable by joint velocities such that the Euclidean norm of $\dot{q}$:

$$\|\dot{q}\| = (\dot{q}_1^2 + \dot{q}_2^2 + \ldots + \dot{q}_n^2)^{1/2} \qquad (I\text{-}42)$$

satisfies $\|\dot{q}\| \leq 1$, This set is an ellipsoid in the *m*-dimensional Euclidean space. In the direction of the major axis of the ellipsoid, the end-effector can move at high speed, on the other hand, in the direction of the minor axis, the end-effector can move only at low speed. If the ellipsoid is almost a sphere, the end-effector can move in all directions uniformly. Also, the larger the ellipsoid is, the faster the end-effector can move. This ellipsoid is called the *manipulability ellipsoid*.

we can show that the manipulability ellipsoid is given by the set of all v satisfying:

$$v^T (J^+)^T J^+ v \leq 1 \qquad (I\text{-}43)$$

where J^+ is the pseudo inverse of J.

the principal axes of the manipulability ellipsoid can be found by using the singular value decomposition of J (appendix 1) given by :

$$J = U \sum V^T \qquad (I\text{-}44)$$

where $U(m \times m)$ and $V\ (n \times n)$ are orthogonal matrices and where $\sum$ is an $(m \times n)$ matrix. The principal axes of the manipulability are: $\sigma_1 u_1$, $\sigma_2 u_2, ..., \sigma_m u_m$, where σ_i is the *ith singular value* of *J* and the radius in the direction if u_i ; and u_i is the *ith* column vector of *U* **Figure 1.11**.

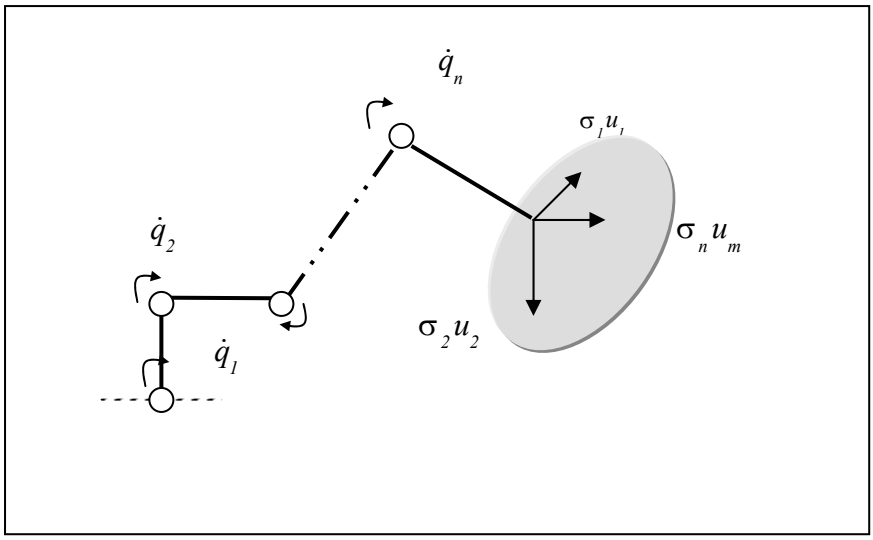


Figure 1.11: Manipulability ellipsoid.

I.2.10.2 Manipulability Measure

One of the representative measures for the ability of manipulation derived from the manipulability ellipsoid is the volume of the ellipsoid. This is given by $c_m w$ where:

$$w = \sigma_i \sigma_2 \ldots \sigma_m, \qquad (I\text{-}45)$$

$$c_m = \begin{cases} (2\pi)^{m/2} / \left[2 \,.4.6.....(m-2)\,.\ m\right] & \text{If m is even} \qquad (I\text{-}46a) \\ 2(2\,\pi)^{(m-1)/2} / \left[1\,.3.5.....(m-2).\ m\right] & \text{If m is odd} \qquad (I\text{-}46b) \end{cases}$$

w is called *the manipulability measure* for a manipulator at a configuration ***q***.
the manipulability measure *w* has the following properties :

a) $w = \sqrt{\det\ ((J)\ J^T(q))}$ (I-*47)*

b) when non redundant manipulators are considered *(m = n)* :
$w = \left|\det(J(q))\right|$ (I-*48)*

c) generally $w \geq 0$, and $w = 0$ only if the manipulator is in a singular configuration (*rank(J(q)) <m)* from this fact we can regard the manipulability measure as a kind of distance of the manipulator configuration from singular ones.

Another kind manipulability more precise and more detailed which take into consideration the dynamics of the manipulator can be estimated with its dynamic-manipulability ellipsoid, and dynamic-manipulability measure, this will not be considered here.

I.3 Robot Dynamics

It is necessary to analyse the dynamic characteristics of a manipulator in order to control it, to simulate it, and to evaluate its performance. Two methods exist for obtaining the equations of motion: the **Lagrangian** and the **Newton-Euler** formulations.

The Lagrangian formulation uses the concept of the Lagrangian, which is related to the kinetic energy. It results in simple equations of motion suitable to examine the effects of various parameters on the motion of the manipulator. It has been the standard one since the 1970s.

Recently, as the need for more rapid and accurate operations of manipulators and for real-time computation of the dynamics equation has increased, the Newton-Euler formulation has been found to be superior to the Lagrangian formulation for the purpose of fast calculation.

I.3.1 The Lagrangian Formulation

One of the merits of the Lagrange's equation of motion is that any variable can be used instead of the orthogonal coordinates as long as they can uniquely specify the positions of objects; these variables are called *generalized coordinates*. Generally when dealing with manipulators of n degrees of freedom, we choose the joint variables q_i as the generalized coordinate, and calculate the kinetic energy K_i and the potential energy P_i of each link i separately, and after that, we calculate the Lagrangian function given by:

$$L = \sum_{i=1}^{n} K_i - \sum_{i=1}^{n} P_i \qquad (I\text{-}49)$$

and after that, we obtain the Q_i which is the generalized force term corresponding to the joint variable q_i which is the joint driving torque τ_i :

$$Q_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \qquad (I\text{-}50)$$

equation *(*I-*49)* is called *Lagrange's equation of motion of link i.* Finally, we obtain the general equation of motion of the manipulator:

$$\tau = M(q)\,\ddot{q} + h(q, \dot{q}) + g(q) \qquad (I\text{-}51)$$

where $M(q)\,\ddot{q}$ is the inertial force term, $h(q,\dot{q})$ is an n- dimensional vector representing the centrifugal and coriolis force term, and $g(q)$ is an n- dimensional vector representing the gravity force term. $M(q)$ is an $(n \times n)$ symmetric matrix called the inertia matrix in joint coordinate.

The equation of motion of a general n-link manipulator can derived directly from the homogenous transforms among link frames by deriving each term of it:

M_{ij} is the *(i,j)* element of $M(q)$ and is given by:

$$M_{ij} = \sum_{k=\max(i,j)}^{n} tr\left(\frac{\partial\, {}^0T_k}{\partial q_j}\,\hat{H}_k\,\frac{\partial\, {}^0T_k^T}{\partial q_i}\right) \qquad (I\text{-}52)$$

h_i is the *ith* element of $h(q,\dot{q})$ and is given by :

$$h_i = \sum_{j=1}^{n} \sum_{m=1}^{n} \sum_{k=\max(i,j,m)}^{n} tr\left(\frac{\partial^2 \ {}^0T_k}{\partial q_j \partial q_m} \hat{H}_k \frac{\partial \ {}^0T_k^T}{\partial q_i} \right) \dot{q}_j \ \dot{q}_m \qquad (I\text{-}53)$$

g_i is the *ith* element of $g(q)$ and is given by:

$$g_i = -\sum_{j=1}^{n} m_j \ \tilde{g}^T \ \frac{\partial \ {}^0T_j}{\partial q_i} \ {}^j\hat{s}_j \qquad (I\text{-}54)$$

such that *tr*(.) denotes the trace of a matrix, $\tilde{g} = [\tilde{g}_x, \tilde{g}_y, \tilde{g}_z, 0]^T$ is the gravitational acceleration vector in $\sum_0$, ${}^i\hat{s}_i = [\hat{s}_{ix}, \ \hat{s}_{iy}, \ \hat{s}_{iz}]^T$ is a vector from 0_i of frame $\sum_i$ to the centre of mass of link *i*
expressed in $\sum_i$, and $\hat{H}_i$ is defined as the pseudo inertia matrix of link *i*. (see reference [1] .p91 for a complete illustration).

I.3.2 The Newton-Euler Formulation

We can consider it as a way to calculate the joint driving force for realizing a given trajectory of joint vector ***q***. suppose the present values of joint displacements q_i and joint velocities $\dot{q}_i$ and the desired values of joint accelerations $\ddot{q}_i$ are given for all links, also suppose that the force f_{load} and moment n_{load} exerted on the end link by the environment or a grasped object are given. The procedure consists of:

First, calculating the angular velocity ω_i, the angular acceleration $\dot{\omega}_i$, the linear velocity $\dot{p}_i$, and the linear acceleration $\ddot{p}_i$ of link *i* with respect to a reference frame, starting from the base to the end link.

Second, using the Newton's and Euler's equations (see reference [1] .p82-85), we calculate the force $\hat{f}_i$ and the moment $\hat{n}_i$ that must be applied to the centre of mass of link *i* to realize such motion.

Third, we calculate the force f_i and the moment n_i that must be applied at joint *i* to produce $\hat{f}_i$ and $\hat{n}_i$, starting from the end link and moving inward to the base, with the given values of f_{load} and n_{load} as boundary conditions.

Finally, we calculate the joint driving force τ_i necessary at each joint *i*. The set of equations needed to realize the above calculations corresponds to the Newton's Euler formulation of the equation of motion.

The relation among the link accelerations can be derived by differentiating equations *(*I*-19)*, *(*I*-20)*, and in order to make the calculations easier, all the vectors related to link *i* will be expressed with respect to the link frame $\sum_i$, resulting on the following equations:

$${}^i\omega_i = \begin{cases} {}^{i-1}R_i^T \ {}^{i-1}\omega_{i-1} + e_z \dot{q}_i & \text{if R} \qquad (I\text{-}55a) \\ {}^{i-1}R_i^T \ {}^{i-1}\omega_{i-1} & \text{if P} \qquad (I\text{-}55b) \end{cases}$$

$${}^i\dot{\omega}_i = \begin{cases} {}^{i-1}R_i^T \ {}^{i-1}\dot{\omega}_{i-1} + e_z \ddot{q}_i + ({}^{i-1}R_i^T \ {}^{i-1}\omega_{i-1}) \times e_z \ \dot{q}_i & \text{if R} \qquad (I\text{-}56a) \\ {}^{i-1}R_i^T \ {}^{i-1}\dot{\omega}_{i-1} & \text{if P} \qquad (I\text{-}56b) \end{cases}$$

$$ {}^{i}\ddot{p}_i = \begin{cases} {}^{i-1}R_i^T\left[{}^{i-1}\ddot{p}_{i-1} + {}^{i-1}\dot{\omega}_{i-1}\times {}^{i-1}\hat{p}_i + {}^{i-1}\omega_{i-1}\times({}^{i-1}\omega_{i-1}\times {}^{i-1}\hat{p}_i)\right] & \text{if R} \quad (I\text{-}57a) \\ {}^{i-1}R_i^T\left[{}^{i-1}\ddot{p}_{i-1}+{}^{i-1}\dot{\omega}_{i-1}\times {}^{i-1}\hat{p}_i+{}^{i-1}\omega_{i-1}\times({}^{i-1}\omega_{i-1}\times {}^{i-1}\hat{p}_i)\right]+2\,({}^{i-1}R_i^{T}\,{}^{i-1}\omega_{i-1})\times(e_z\dot{q}_i)+e_z\ddot{q}_i & \text{if P} \quad (I\text{-}57b) \end{cases} $$

$$ {}^{i}\ddot{s}_i = {}^{i}\ddot{p}_i + {}^{i}\dot{\omega}_i\times {}^{i}\hat{s}_i + {}^{i}\omega_i\times({}^{i}\omega_i\times {}^{i}\hat{s}_i) \qquad (I\text{-}58) $$

$$ {}^{i}\hat{f}_i = m_i\,{}^{i}\ddot{s}_i \qquad (I\text{-}59) $$

$$ {}^{i}\hat{n}_i = {}^{i}I_i\,{}^{i}\dot{\omega}_i + {}^{i}\omega_i\times({}^{i}I_i\,{}^{i}\omega_i) \qquad (I\text{-}60) $$

$$ {}^{i}f_i = {}^{i}R_{i+1}\,{}^{i+1}f_{i+1} + {}^{i}\hat{f}_i \qquad (I\text{-}61) $$

$$ {}^{i}n_i = {}^{i}R_{i+1}\,{}^{i+1}n_{i+1} + {}^{i}\hat{n}_i + {}^{i}\hat{s}_i\times {}^{i}\hat{f}_i + {}^{i}\hat{p}_{i+1}\times({}^{i}R_{i+1}\,{}^{i+1}f_{i+1}) \qquad (I\text{-}62) $$

$$ \tau_i = \begin{cases} e_z^T\,{}^{i}n_i & \text{if R} \quad (I\text{-}63a) \\ e_z^T\,{}^{i}f_i & \text{if P} \quad (I\text{-}63b) \end{cases} $$

${}^{i}I_i$ is the inertia tensor expressed in frame $\sum_i$ and $e_z = [0,\ 0,\ 1]^T$, if it is necessary to consider gravity, we only have to set: ${}^{0}\ddot{p}_0 = -\tilde{g} = -[\tilde{g}_x,\ \tilde{g}_y,\ \tilde{g}_z]^T$, if we have to consider friction at joints, we may include a friction term γ_{Fi} to equation *(*I-*63)*:

$$ \tau_i = \begin{cases} e_z^T\,{}^{i}n_i + \gamma_{Fi} & \text{if R} \quad (I\text{-}64a) \\ e_z^T\,{}^{i}f_i + \gamma_{Fi} & \text{if P} \quad (I\text{-}64b) \end{cases} $$

The two formulations will be used to derive the equation of motion of the manipulator used in the application and it can be seen that they give exactly the same result.

I.3.3 Identification Problem of Manipulators

When we wish to use the dynamics equation of motion of a manipulator, the values of various parameters in the equation must be known. These parameters may be kinematics parameters typically derived directly from the link parameters, or they may be dynamic parameters appearing only in the dynamic equations consisting mainly of inertial parameters and friction constants.

These parameters can be calculated **off-line** by measuring the size and weight of each part of the manipulator, but this method does not produce very accurate results, the most practical method is to estimate the dynamic parameters by some identification method based on data taken during certain test motions of the manipulator and since the structure of a general manipulator is well known and not very complex the problem is relatively easily solved; however, since the dynamics change when a manipulator grasps an object or perform some task, it is desirable to identify this changes and to use the results for control. This process requires an **on-line** identification method. . (see reference [1] .p113. For an identification scheme taking these points into consideration).

I.4 Trajectory Generation

The basic problem is how to select a trajectory between a given initial position ($\boldsymbol{r}_0$) and a final position ($\boldsymbol{r}_f$) of the end-effector with a time interval ($\boldsymbol{t}_f$) allowed for moving between them. Various methods for solving this problem have been proposed. A simple method based on polynomial functions of time will be presented here. This method can be divided into two schemes, one in which the trajectory is considered in terms of joint variables and one in which it is considered in terms of position variables of the end-effector.

I.4.1 The Joint Variables Scheme

Suppose we are given the initial end-effector position vector $\boldsymbol{r}_0$ and the final desired end-effector position vector $\boldsymbol{r}_f$, we first , by solving the inverse kinematics problem, determine the initial and final joint vectors $\boldsymbol{q}_0$ and $\boldsymbol{q}_f$ respectively, we will now apply the generation algorithm on one joint variables q_i and represent it by ξ . We assume that the value of ξ at the initial time *(0)* is ξ_0 and that the value at the final time *(t_f)* is ξ_f ;

$$\xi\ (0)\ = \xi_0\ ,\ \xi\ (t_f)\ = \xi_f \qquad \text{(I-65)}$$

we further assume that the velocity and the acceleration of ξ must satisfy the following boundary conditions:

$$\dot{\xi}\ (0) = \xi_0\ ,\quad \dot{\xi}\ (t_f) = \dot{\xi}_f, \qquad \text{(I-66)}$$

$$\ddot{\xi}\ (0) = \ddot{\xi}_0\ ,\quad \ddot{\xi}(t_f) = \ddot{\xi}_f \qquad \text{(I-67)}$$

There exist several functions that can satisfy these conditions, we select polynomials of time *t* because of their ease of computation. The polynomials of the lowest order that satisfy these constraints are of fifth order (six conditions – six equations – six unknowns); thus, we express ξ*(t)* as:

$$\xi\ (t)\ =\ a_0 + a_1 t + a_2 t^2 + a_3\, t^3 + a_4 t^4 + a_5\ t^5 \qquad \text{(I-68)}$$

Then the unknown coefficients $a_1 – a_5$ that satisfy the conditions are:

$$a_0 = \xi_0\ , \qquad \text{(I-69a)}$$

$$a_1 = \dot{\xi}_0\ , \qquad \text{(I-69b)}$$

$$a_2 = \tfrac{1}{2}\,\ddot{\xi}_0\ , \qquad \text{(I-69c)}$$

$$a_3 = \frac{1}{2t_f^3}\left[20\xi_f\ -\ 20\xi_0\ -\ (8\dot{\xi}_f\ +\ 12\ \dot{\xi}_0\ )\ t_f\ -\ (3\ddot{\xi}_0\ -\ddot{\xi}_f\ )\ t_f^2\ \right], \qquad \text{(I-69d)}$$

$$a_4 = \frac{1}{2\ t_f^4}\left[30\xi_0\ -\ 30\xi_f\ +\ (14\ \dot{\xi}_f\ +\ 16\ \dot{\xi}_0\ )\ t_f + (3\ \ddot{\xi}_0\ -\ 2\ddot{\xi}_f\ )\ t_f^2\ \right], \qquad \text{(I-69e)}$$

$$a_5 = \frac{1}{2t_f^5}\left[12\xi_f\ -\ 12\ \xi_0\ -\ (6\dot{\xi}_f\ +\ 6\dot{\xi}_0\ )t_f - (\ddot{\xi}_0\ -\ddot{\xi}_f\ )\ t_f^2\right], \qquad \text{(I-69f)}$$

if we consider that $\ddot{\xi}_0 = \ddot{\xi}_f = 0$ and we can show that :

$$\xi_f\ -\ \xi_0 = \frac{t_f}{2}\ (\dot{\xi}_0\ +\ \dot{\xi}_f) \qquad \text{(I-70)}$$

then $a_5 = 0$, which implies that only a fourth-order polynomial is required for $\xi(t)$.

The trajectory is generated by using combinations of fourth-order polynomials and straight lines.

It starts from rest at ξ_0, passes through the phases of acceleration, constant velocity, and deceleration, and finally comes to complete stop at ξ_f. applying the following procedure:

A. choose the value of parameter Δ that denotes one half the acceleration and deceleration period,
B. second, determine the auxiliary points ξ_{02} and ξ_{f1} by :
    a. taking $\xi_{01}=\xi_0$ *at* $t=\Delta$ $\xi_{f2}=\xi_f$ *at* $t=t_f-\Delta$
    b. ξ_{01} and ξ_{f2} are connected by a straight line and ξ_{02} and ξ_{f1} are determined on the straight line at times $t=2\Delta$ and $t=t_f-2\Delta$ respectively.

We determine the trajectory segments between ξ_0 and ξ_{02}, and between ξ_{f1} and ξ_f by fourth-order polynomials such that their velocities satisfy relation *(*I-*70)* and their accelerations are zero at this boundary points, **Figure 1.12**.

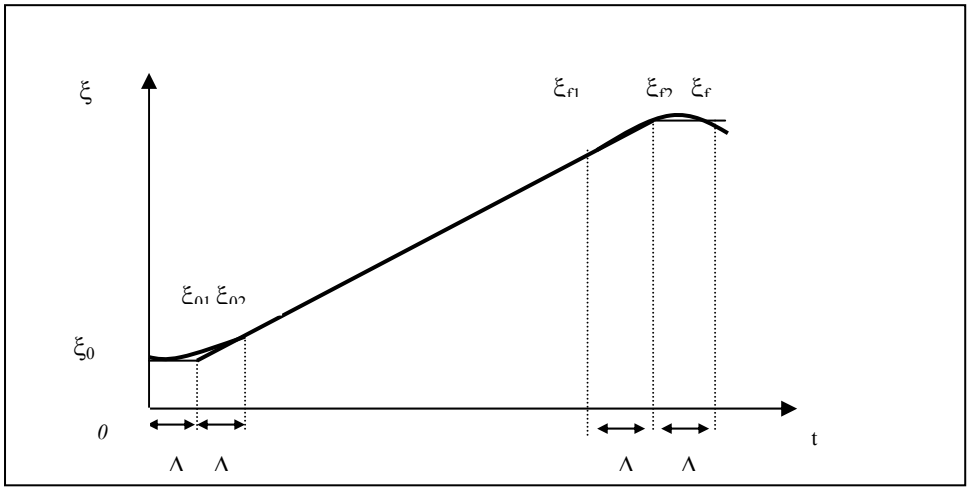


Figure 1.12: The Trajectory with acceleration, constant velocity, and deceleration phases.

This same procedure can be used to generate other types of trajectories where we are supposed to pass through intermediate points exactly or not, just by considering these intermediate points as temporary end point of a previous trajectory segment and the start point of the next segment.

I.4.2 The End-Effector Position Variables Scheme

When the trajectory between $\boldsymbol{r}_0$ and $\boldsymbol{r}_f$ is generated by the joint variable scheme, it is sometimes difficult to predict the motion of the end-effector in space. There are also cases where we want the end-effector to follow a specific trajectory (ex: arc welding). In such cases we wish to generate a trajectory between $\boldsymbol{r}_0$ and $\boldsymbol{r}_f$ in terms of the position variables of the end-effector. The general procedure is to, first, obtain the trajectory in terms of the position variables (a circle, a sinusoid, a straight line….), second, using the inverse kinematics problem, obtain the corresponding trajectory of the joint vector $\boldsymbol{q}$. And here, we have to verify if the joint variables trajectories can be realized or not.

I.5 The Controller

I.5.1 The Concept

The fundamental elements of tasks performed by robot manipulators are:

A. To move the end-effector, with or without a load, along a desired trajectory.
B. To exert a desired force on an object when the end-effector is in contact with it.

The former is called **Position Control** and the later **Force control**.

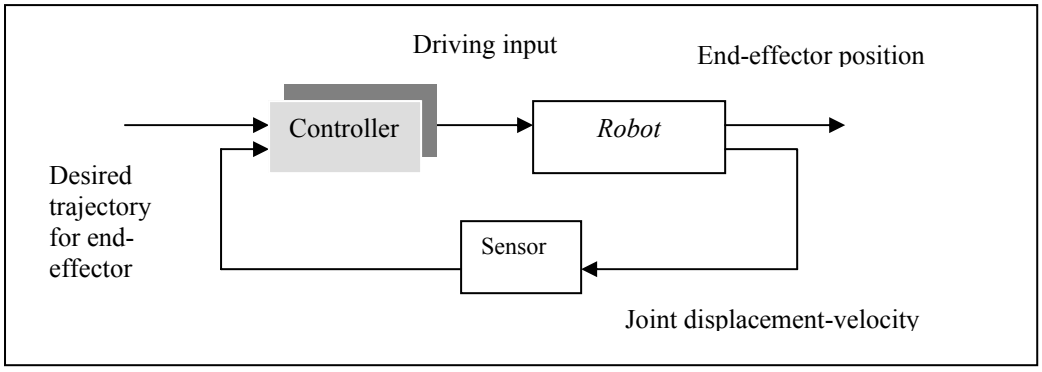


Figure 1.13: Position control system.

Figure 1.13 is a rough sketch of a typical controller for position control, joints positions and velocities are generally measured by joints sensors, such as potentiometers, tachometer-generators, and/or encoders.

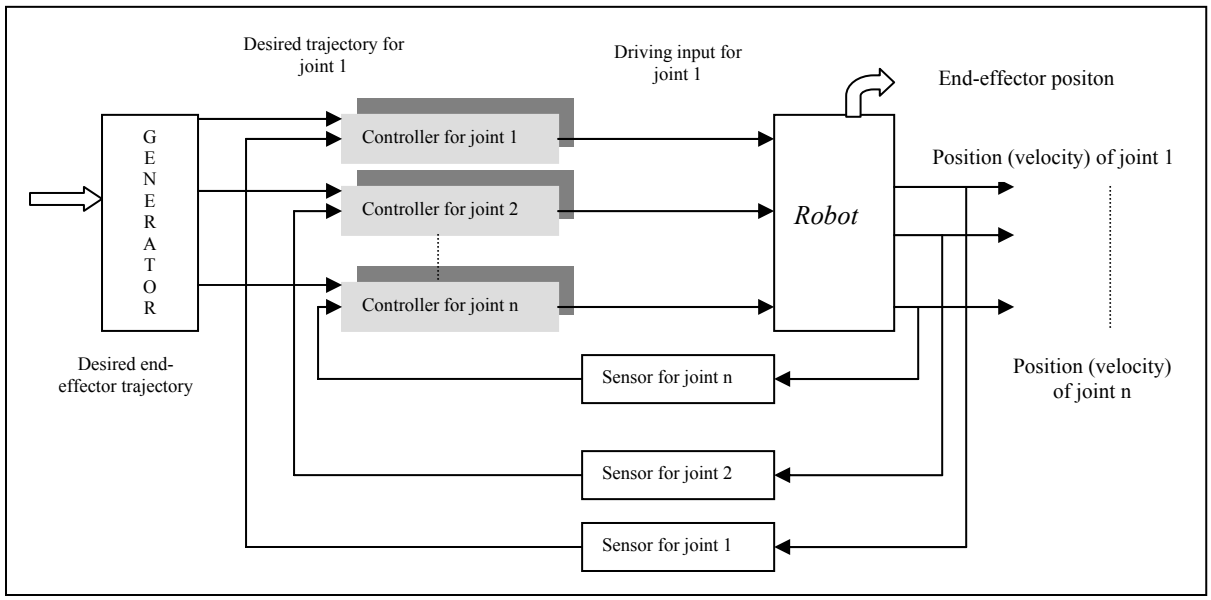


Figure 1.14: Example of a controller.

Figure 1.14 is a detailed structure of a position controller; the outputs on the right of the figure are the joints positions of the manipulator. The joint-trajectory generator determines the desired joint trajectory from the desired trajectory of the end-effector. The desired joints trajectories are then given to the joints controllers. Each joint controller is a servomechanism **Figure 1.15** that uses the position and velocity feedback with constant feedback gains ex: **PD**, **PID**; this kind of controllers has often been used for general industrial manipulators. Generally, changes in dynamics are due to changes in the manipulator configuration; there is also interaction among the joints. The above controller design figure.1.14 implicitly assumes

the possibility of coping with such changes in the manipulator dynamics and joint interaction by regarding them as disturbances. However, when there are severe demands for fast and accurate positioning, the tracking performance of this type of controller is no longer adequate and more complex control algorithms must be used ex: **Computed torque controller**.

In force control, it is generally necessary to measure the forces driving the joints or/and the contact force between the end-effector and the object by force sensors, and to feed these signals back to the controller.

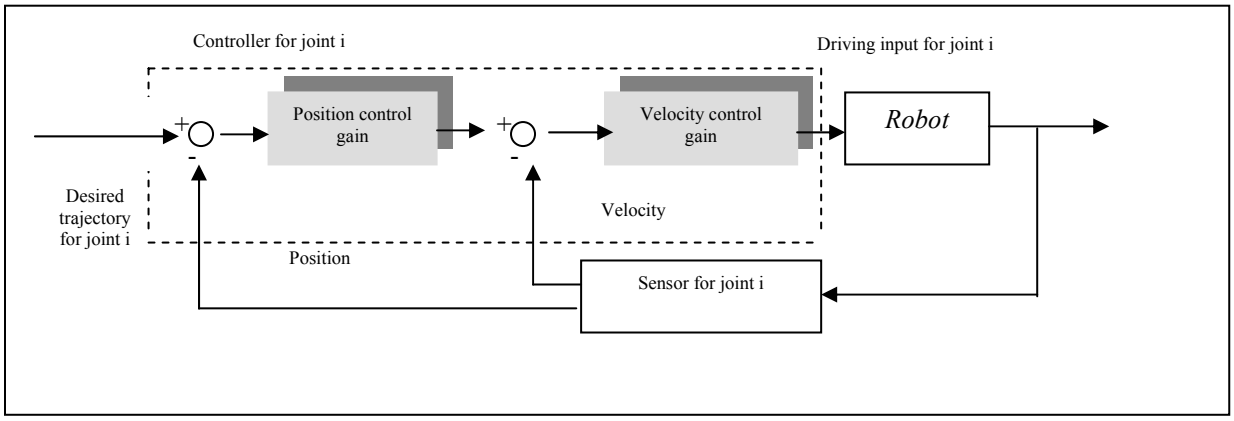


Figure 1.15: Position –velocity feedback servo-system.

I.5.2 Linear Feed-Back Control

Most manipulators in practical use today have a controller consisting of position and velocity feedback loops which are independent for each joint, and in spite of inherent non-linear coupling among the joints of a manipulator, such a controller is usually fairly effective.
The dynamic equation of a manipulator with a friction term added is given by:

$$\tau = M(q)\,\ddot{q} + h(q, \dot{q}) + V\dot{q} + g\,(q) \qquad (I\text{-}71)$$

where V is the joint friction coefficient matrix and $M(q)$ satisfying $M(q)=M^T(q) \; >0$ that is $M(q)$ is a positive definite matrix . Denoting the reduction ratio between the actuator displacement vector q_a and the joint vector q by a non-singular diagonal matrix with constant coefficient G_r; we have:

$$G_r \; q \; = \; q_a \qquad (I\text{-}72)$$

we assume that the dynamic equation of the actuator is given by:

$$\tau_m \; = \; M_a \;\; \ddot{q}_a + V_a \dot{q}_a + \tau_a \qquad (I\text{-}73)$$

M_a : inertia matrix of the actuator,
V_a : the frictional coefficient matrix,
τ_m :the generated force by the actuator,
τ_a :the joint deriving force transmitted to the link mechanism from the actuator.
the relation between τ_a and τ is :

$$\tau = G_r \tau_a \qquad (I\text{-}74)$$

combining equations *(*I-*71)*, *(*I-*72)* and *(*I-*74)* we get the dynamics equation for the whole system of the actuators and the link mechanism:

$$G_r \tau_m = [G_r M_a G_r + M(q)]\ddot{q} + h(q, \dot{q}) + [G_r V_a G_r + V]\dot{q} + g(q) \qquad (I\text{-}75)$$

if we apply a PD feedback controller law:

$$\tau_m = G_r^{-1} \{-K_d \dot{q}(t) + K_p [q_d(t) - q(t)]\} \qquad (I\text{-}76)$$

where $q_d(t)$ is the desired trajectory for $q(t)$.

If we assume a large reduction ratio for each joint actuator and large values for the diagonal elements of K_p and K_d , then the terms $M(q), h(q, \dot{q}), V\dot{q}$ and $g(q)$ are small relative to other terms, which means neglecting the manipulator dynamics and considering it as disturbances. Combining *(*I-*75)* and *(*I-*76)* we get the new dynamics equation of the closed-loop system:

$$[G_r M_a G_r]\ddot{q} + [G_r V_a G_r + K_d]\dot{q} + K_p [q - q_d] = 0 \qquad (I\text{-}77)$$

This represents a set of independent second-order system for each joint. The response characteristics are adjusted by properly selecting the gains K_p and K_d.

If the above assumptions are not valid, we have to take into account the manipulator dynamics.

I.5.3 The Two-Stage Control by Linearization and Servo Compensation

Also called the "Computed Torque control". Modifying the equation *(*I-*71)* we get:

$$\tau = M(q)\ddot{q} + h_N(q, \dot{q}) \qquad (I\text{-}78)$$

where

$$h_N(q, \dot{q}) = h(q, \dot{q}) + V\dot{q} + g(q) \qquad (I\text{-}79)$$

taking $[q^T, \dot{q}^T]^T$ as state variables and considering the non-linear state feedback compensation described by:

$$\tau = h_N(q, \dot{q}) + M(q)u_q \qquad (I\text{-}80)$$

where $u_q = \ddot{q}$ is a new input vector. Then this resulting closed-loop system is linear and decoupled with respect to the joint variables. If we assume an exact model of the manipulator without external disturbances to the system, then it is sufficient to put the desired acceleration $\ddot{q}_d = u_q$ to obtain $q(t) = q_d(t)$ which means that the desired trajectory is perfectly achieved. In practice, however, some modelling errors and some external disturbances are inevitable. The basic idea here is to reduce their effect on the control performance by installing a servo compensator to the linearized system. **Figure 1.16** is a schematic diagram of the control system.

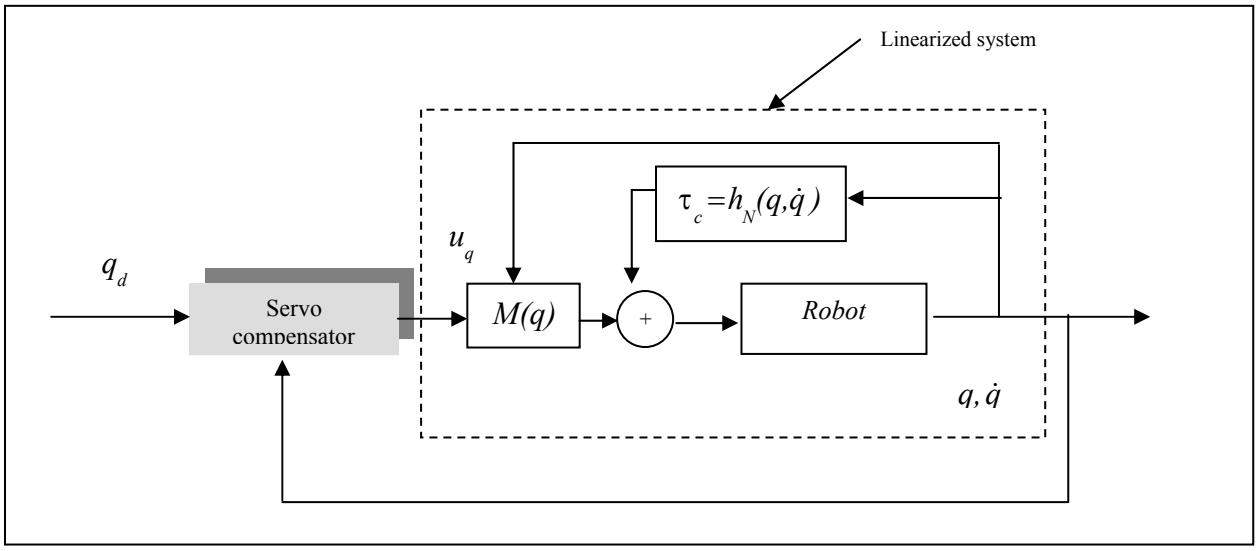


Figure 1.16: Computed torque controller.

taking the servo compensator described by:

$$u_q = \ddot{q}_d + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q) \qquad (I\text{-}81)$$

and if we set :

$$K_p = diag(\omega_c^2) \qquad (I\text{-}82a)$$

$$K_d = dig(2\zeta\omega_c) \qquad (I\text{-}82b)$$

where ω_c and ζ represent the natural frequency and the damping coefficient respectively for each control stage corresponding to each joint, such that $0<\zeta \leq 1$ and $\omega_c > 0$.

The same approach of linearization and compensation can be applied in terms of variables which are more related to the manipulator tasks, such as the position and orientation of the end-effector.

Hence we consider the *n* dimensional output vector position ***r*** (position, and orientation) and using equations *(*I-*8)* and *(*I-*16)* we get:

$$\ddot{r} = J_v(q)\ddot{q} + \dot{J}_v(q)\dot{q} \qquad (I\text{-}83)$$

which implies:

$$\ddot{q} = J_v^{-1}(q)\left[\ddot{r} - \dot{J}_v(q)\,\dot{q}\right] \qquad (I\text{-}84)$$

substituting *(*I-*84)* in *(*I-*80)* we get the non-linear state feedback compensation described by:

$$\tau = h_N(q, \dot{q}) + M(q)J_v^{-1}(q)\left[-\dot{J}_v(q)\dot{q} + u_r\right] \qquad (I\text{-}85)$$

where $u_r = \ddot{r}$ is a new input vector. Then this resulting closed-loop system is linear and decoupled with respect to the vector position ***r*** and as for the previous approach and for the

same reasons, installing a servo compensation is necessary and we obtain the control system in **Figure 1.17**:

$$u_r = \ddot{r}_d + K_d(\dot{r}_d - \dot{r}) + K_p(r_d - r) \qquad (I\text{-}86)$$

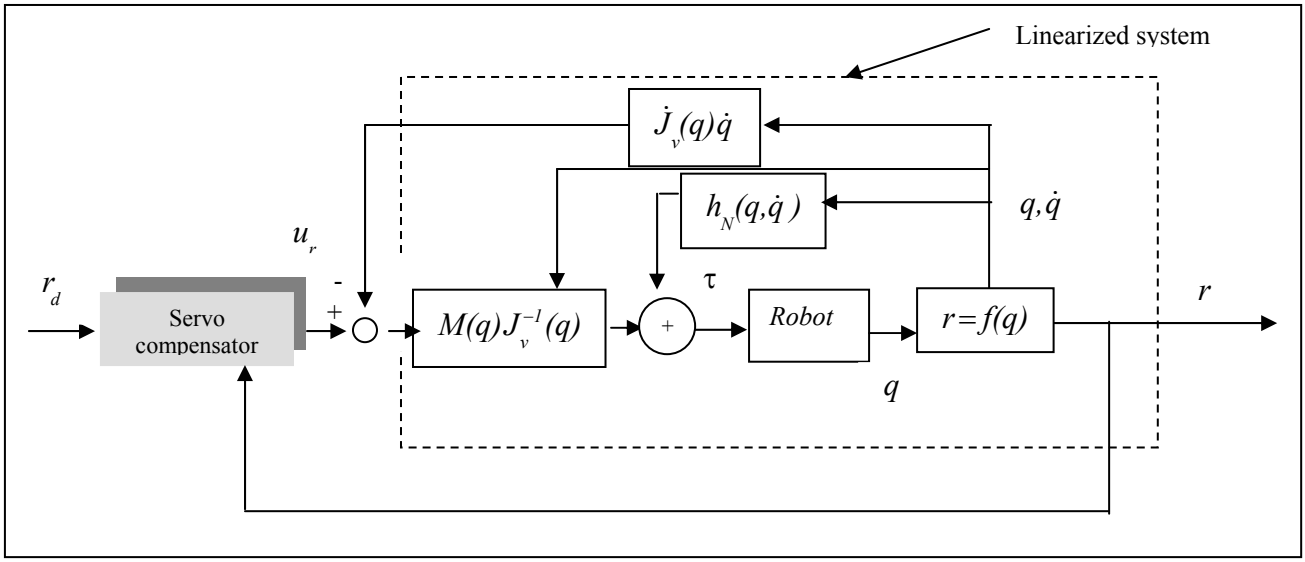


Figure 1.17: Computed torque controller using the vector position.

One of the major problems that may face us in implementing this control scheme is the complexity of the algorithm itself, which requires usually a powerful digital computer for real time applications. Two solutions to this problem have been proposed:

A. The use of the Newton-Euler formulation for computing τ for a given $q,\dot{q},$ and $u_{q\text{-}r}$ in equations *(*I-*80)* and *(*I-*85)* since it can be shown that the Newton-Euler formulation requires less amount of computations than lagrangian formulation, however, it is more difficult to implement and program.
B. The linearization of the dynamics using the desired trajectory, which means calculating τ with q_d, $\dot{q}_d$ and $u_{q\text{-}r}$ in equations (I-*80)* and *(*I-*85)*, but also here, the linearization is less accurate because of the difference between q and q_d, resulting in a degradation of the control performance. If the improvement in performance due to a shorter sampling period is larger than the above degradation, this approach can give good results.
C. The last proposed solution is to distinguish the sampling period for the linearization and that for the servo compensation, such that the linearization, which requires a large amount of computation, may have a large sampling period provided that the servo compensation has a small enough sampling period to cover the error caused by the large sampling period of the linearization.

I.6 Conclusion

In this chapter, we have introduced and explained almost all the aspects required to design, analyse, and control robot manipulators, starting by the introduction of some common manipulator mechanisms, after that we have illustrated the methodology used to analyse cinematically a robot mechanism and as a final step in the analysis, the derivation of the dynamics equation of movement that must be based on the kinematics analysis as well as on the dynamics one. After the analysis steps, come the controllers design and implementation which are closely related to the analysis done previously.

From this chapter, it can be clearly seen that the analysis can easily be done when dealing with small degrees of freedom, but the difficulties will increase exponentially when adding extra degrees of freedom and also we can note that the analysis steps are related and the parameters found in one step are used in the next step starting from the homogenous transforms this is why we have to be careful in the parameters derivation from the beginning and try to always choose the easiest way to place the link frames on the structure so that to get ride of the maximum number of link parameters and as a consequence, decrease the amount of calculations required for the following steps.

In the next chapter, we will introduce the support on which we will perform our real time application the “dSPACE ds1104” DSP card and we will illustrate all the hardware and software topics associated with it and the steps to be followed in order to program it and use it.

CHAPTER II
THE CONTROLLER BOARD "dSPACE ds1104"

II.1 Digital Signal Processing

Digital signal processing (**DSP**) involves the manipulation of digital signals in order to extract useful information from them and there are many reasons why one would want to process an analog signal in a digital fashion by converting it into a digital signal.

The main reasons for that are:

- Digital processing allows programmability: The same DSP hardware can be used for many different applications simply by changing the code residing in memory.
- The possibility to realize functions that cannot be or at least with difficulties realized in analog environment.
- The existence of simulation tools….

The use of these digital signal processing techniques has been widely enlarged to various fields, among them:

- Signal processing (filtering - coding - compression …),
- Medical applications (biomedical imaging …),
- Digital Control applications (DC and AC Motors ….).

The satisfaction of the two major requirements -numerical calculations and real time application - necessary for the implementation of complex algorithms with time constraints which will become crucial when working in real time, has led to the apparition of special purpose processors with a special architecture oriented to performance rather then to extended functionalities and programmers comfort.

The first **Digital Signal Processor -DSP-** appeared at the beginning of the 80th equipped with Analog to Digital (ADC) and Digital to Analog (DAC) converters and with an internal memory to permit the realization of autonomous systems and in 1982 **Texas Instruments** introduces the TMS32010 DSP, the first member of what will be the most famous and widely used DSP family, other constructors exist like **Hitachi**, **Motorola** but essentially the architectures and the functionalities are quite the same and they are continuously developed and improved.

One of the most recent applications of Digital signal processors is the Digital control mainly applied to AC and DC motors.

II.2 Digital Control Systems

The availability of inexpensive processors has led to their extensive use in digital control systems. Although there is a degradation of performance by the introduction of sampling, this is more then overcome by the greater range of compensators that can be achieved digitally. The flexibility that is provided by microprocessors in synthesizing digital compensators yields greatly improved closed-loop performance.

In addition of using the digital computers in designing control systems, Digital Processors can be used to perform the control function. The size and cost advantages associated with these Digital Processors make there use as the compensator or the controller economically and physically feasible. Using such a digital processor with a plant which operates in the continuous time domain requires that its input signal be discrete, which in turn requires the sampling of the signal used by the controller, such sampling may be an inherent characteristic of the system (radar system), or it may be achieved by the use of analog to digital data converters (ADC) that must be incorporated in the control system. The output of the controller must be converted from discrete form into an analog signal by a digital to analog converter (DAC) **Figure 2.1** [3].

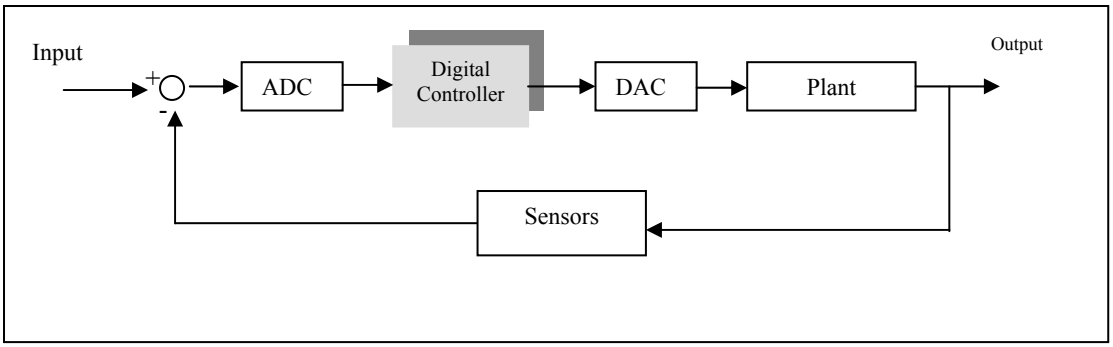


Figure 2.1: Digital control system.

II.3 The dSPACE ds1104 Controller Board

Now we will introduce the Digital Signal Processor "ds1104" that will be used in the real time application and illustrate the procedure to programme it and use it for control purposes, we will consider two aspects:

II.3.1 The Hardware Platform

The ds1104 Controller Board is a single-board system and the hardware package contains:

- One PCI-slot board with a bracket including a 100-pin I/O connector.
- CP1104 Connector Panel with adapter cable to the I/O connector of the board.

The ds1104 Controller Board is a standard board that can be plugged into a PCI slot of a PC. It is a complete real-time control system based on a 603 PowerPC floating-point processor running at 250 MHz. For advanced I/O purposes, the board includes a slave-DSP subsystem based on the TMS320F240 DSP microcontroller.

The CP1104 Connector Panel provides an easy-to-use connection between the ds1104 Controller Board and devices to be connected to it. Devices can be individually connected, disconnected or interchanged via BNC connectors and Sub-D connectors without soldering. This simplifies system construction, testing and troubleshooting.

The **Figure 2.2** gives an illustration of the architecture and the functional units of the ds1104.

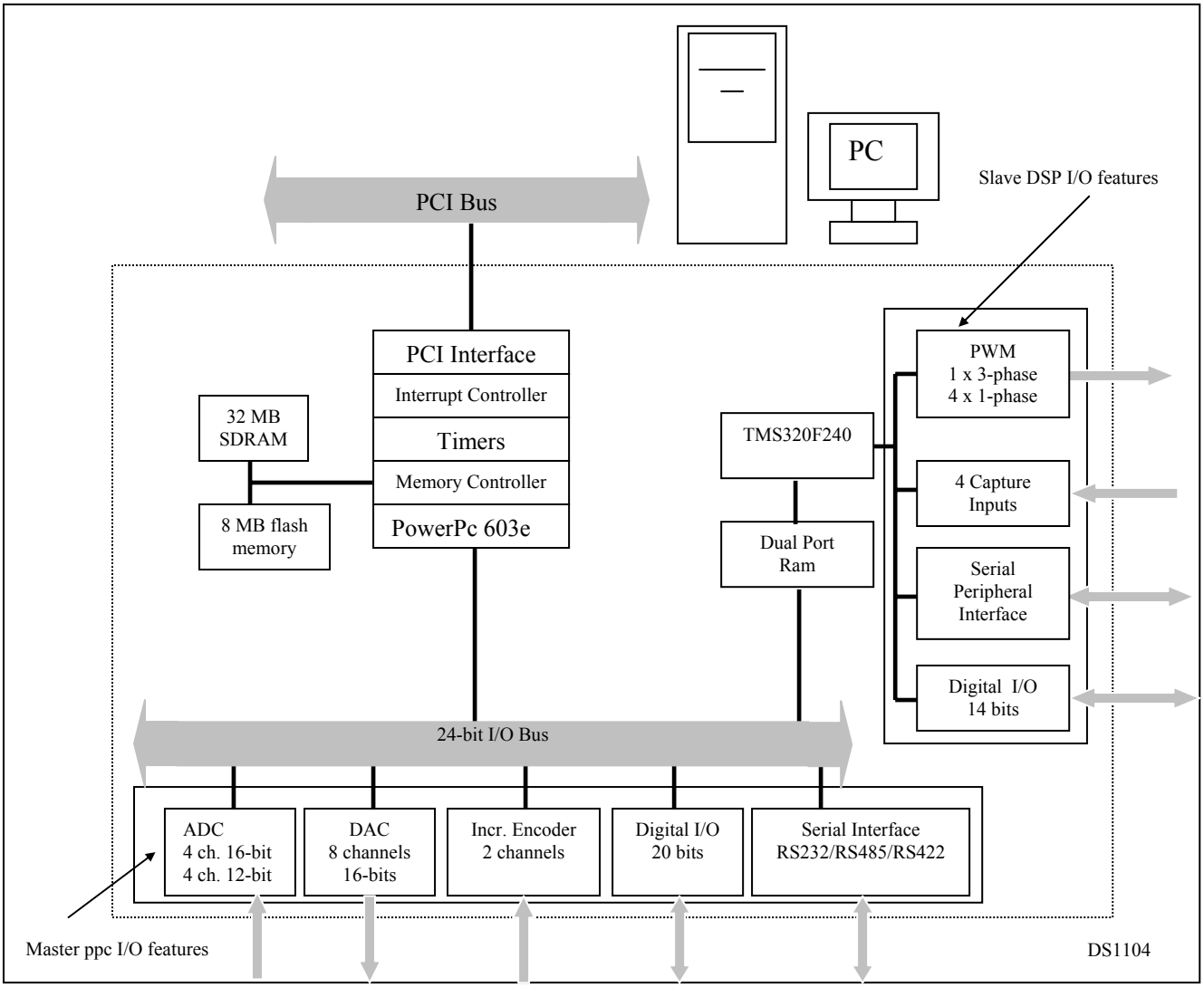


Figure 2.2: The architecture and the functional units of the DS1104.

II.3.1.1 The MPC8240 Computing Unit

It is the ds1104's main processing unit representing the computing power of the board, and featuring several I/O units. It consists of:

- A 64-bit floating-point processor PowerPC 603e microprocessor (the master PPC) on which your control models will be implemented:
  - Running at 250 MHz (CPU clock).
  - Containing a 16-KByte L1 data cache.
  - Containing a 16-KByte L1 instruction cache.
- An interrupt controller: The ds1104 provides access to various hardware interrupts - originating either from on-board devices such as timers, ADC "end of conversion"…, or from external devices connected to the board such as Host PC interrupts, user interrupts…. The interrupt controller of the master PPC samples the interrupts originating from outside the master PPC at a frequency of BCLK/ 64.
- A synchronous DRAM controller: The ds1104 is equipped with two memory sections:
  - Global memory:
    - 32-MByte synchronous DRAM (SDRAM) for applications and data
    - Fully cached (L1 cache)

  - Flash memory:
    - 8 MByte, divided into 4 blocks of 2 MByte each
    - 6.5 MByte can be used for a user-specific application
    - 1.5 MByte are reserved for the boot firmware
    - 8-bit read / write access by master PPC
    - At least 100,000 erase cycles possible

An application loaded to the flash memory will be started automatically after reboot.

- Timers: The ds1104 board is equipped with 6 timer devices. The timers are driven by the bus clock whose frequency is referred to as BCLK , they can be divided into:
  - Time Base Counter:
    - Free-running 64-bit up counter driven by BCLK/4
    - Is used for measurement of relative and absolute times (for execution time measurement and delays)
    - Is used for time-stamping
  - Timers 0 ... 3:
    - 32-bit down counters driven by BCLK/8
    - Selectable period for each timer
    - Used as a trigger for periodic tasks
    - Generation of a timer interrupt when counter reaches 0, and automatic reload.
  - Decrementer:
    - 32-bit down counter driven by BCLK/4
    - Selectable period
    - Used as a trigger for periodic tasks
    - Generation of a timer interrupt when counter reaches 0, and automatic reload.
- A PCI interface (5 V, 32 bit, 33 MHz): The ds1104 provides a PCI interface requiring a single 5 V PCI slot. The interface has the following characteristics:
  - Access from/to the host PC via 33 MHz-PCI interface: The interface serves the board setup, program downloads and runtime data transfers from/to the host PC.
  - Interrupt line: The host interface provides a bidirectional interrupt line: Via this line, the host PC can send interrupt requests to the master PPC and vice versa. Both the host PC and the master PPC can monitor the state of the interrupt line to detect when the corresponding interrupt service is finished.

II.3.1.2 Master PPC I/O Features

Before talking about the I/O features provided by The DSP Card, let's introduce two of the main important operations required in digital signal processing:

II.3.1.2.1 The Analog to Digital Conversion (ADC)

Analog-to-Digital Converters (ADC) are devices which convert a voltage level to a digital word for use in the computer. The process of analog-to-digital signal conversion consists of converting a continuous time and amplitude signal into discrete time and amplitude values. Sample, hold and quantization constitute the steps needed to achieve analog-to-digital signal conversion [4].

II.3.1.2.1.1 Sample and Hold

Sampling is the process of generating discrete time samples from an analog signal [4], it is done by taking samples at periodic instant of time (T_s) called the sampling period and whose amplitude corresponds to the value of the signal at this instant **Figure 2.3**. A loss of information is inevitable since the discrete signal is not defined between the sampling periods which explain the need of a holding circuit. Sampling frequency must be greater than twice the maximum frequency of the analog signal $f_s = 1/T_s \geq 2\, f_{max}$ *'shanon theorem'*, after filtering the analog signal and removing the high frequencies (noise and interferences) that are present in most analog signals. The hold step consists of holding the output amplitude constant at the value of the impulse for the duration of the sampling period T_s.

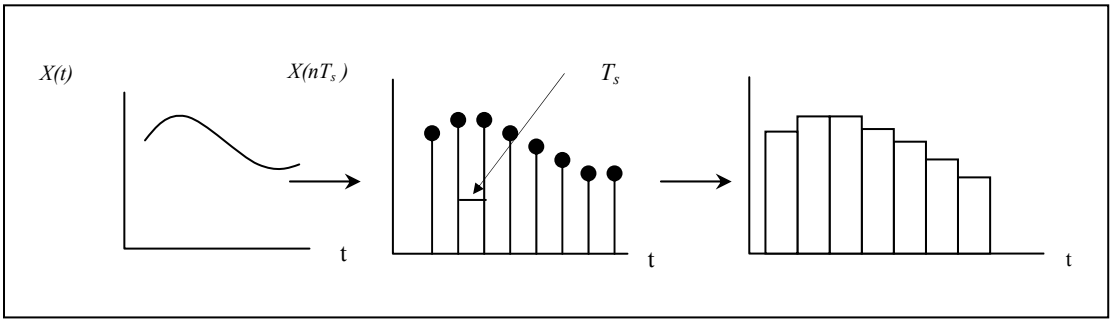


Figure 2.3: Sample and hold process.

II.3.1.2.1.2 Quantization

An ADC converter has a finite number of bits (or **bit resolution**), as a result, continuous amplitude values get represented or approximated by discrete amplitude levels. The process of converting continuous amplitude into discrete amplitude levels is called **quantization Figure 2.4**. This approximation leads to an error called **quantization noise** uniformly distributed over *-0.5LSB* and *0.5LSB* (*LSB* stands for Least Significant Bit, $LSB = V_{ref}/2^N$ such that V_{ref}: Input reference voltage and N: is the converter bit resolution).other performance metrics exist to evaluate an ADC converter like:

- **The Signal to Noise Ratio (SNR)** which is the ratio of the signal power to noise power at the ADC output.
- **Offset Error**: ideally, the analog input to an ADC converter should be *0 V* to get an output word 0, but practically, there exists always an offset.
- **Gain Error**: it can be explained by the fact that the *(LSB)* steps width is not exactly the same from on step to another which introduces automatically an error in the conversion.

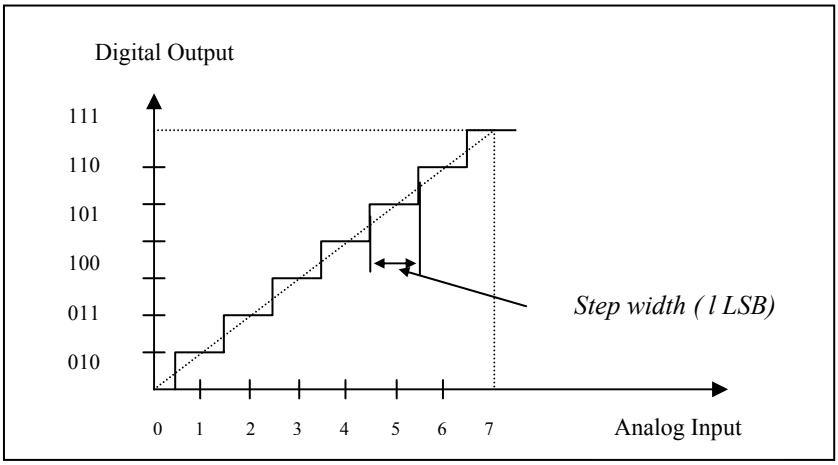


Figure 2.4: I/O Characteristic of a 3-bit ADC converter.

II.3.1.2.2 Digital-to-Analog Signal Conversion DAC

After a signal has been digitally treated and conditioned by the processing unit (Digital Controller) of **Figure 2.1**, a DAC converter is used to convert the sampled binary information back in to an analog signal. The conversion is called a zero order hold type where each output sample level is a function of its binary weight value and is held until the next sample arrives [8].

A DAC represents a limited number of discrete digital input codes by a corresponding number of discrete analog output values. A DAC can be thought of as a digitally controlled potentiometer whose output is a fraction of the full-scale analog voltage determined by the digital input code **Figure 2.5**. The same performance metrics used to evaluate the ADC converters exist also for the DAC converters, the only difference is that here the DAC inputs are words of bits and the output is an analog signal.

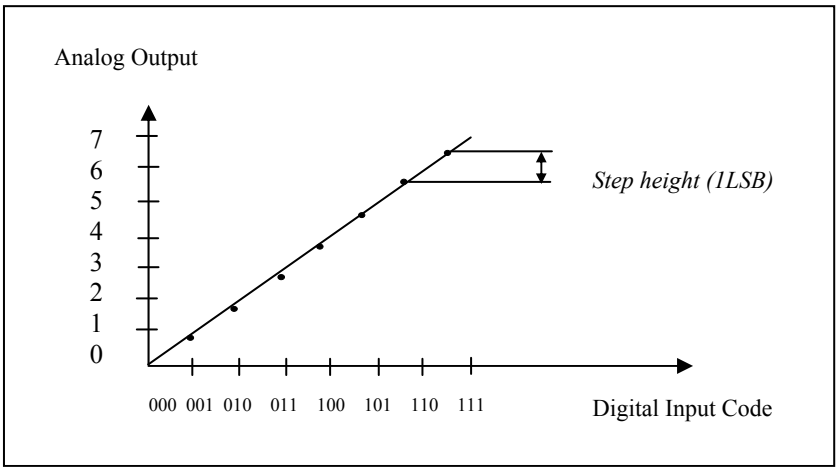


Figure 2.5: I/O Characteristic of a 3-bit DAC converter.

The master PPC controls the following I/O features of the ds1104:

- ADC unit featuring two different types of ADC converters:
  - 1 ADC converter (ADC1) multiplexed to four channels (signals ADCH1 ... ADCH4). The input signals of the converter are selected by a 4:1 input multiplexer.
  - The ADC converter has the following characteristics:
    - 16-bit resolution
    - ±10 V input voltage range
    - ± 5 mV offset error
    - ± 0.25% gain error
    - > 80 dB signal-to-noise ratio (SNR)
  - 4 parallel ADC converters (ADC2 ... ADC5) with one channel each (signals ADCH5 ... ADCH8). The ADC converters have the following characteristics:
    - 12-bit resolution
    - ±10 V input voltage range
    - ± 5 mV offset error
    - ± 0.5% gain error
    - > 70 dB signal-to-noise ratio (SNR)
- DAC unit featuring:
  - 8 parallel DAC channels (signals DACH1 ... DACH8) with the following characteristics:

  - 16-bit resolution
  - ±10 V output voltage range
  - ± 1 mV offset error
  - ± 0.1% gain error
  - > 80 dB signal-to-noise ratio (SNR)
  - Transparent (converted value output immediately) and latched mode (converted value output after a strobe signal).
- A bit I/O unit with the following characteristics:
  - 20-bit digital I/O
  - Direction selectable for each channel individually
  - ±5 mA maximum output current
  - TTL voltage range for input and output
- An incremental encoder interface. It has the following characteristics:
  - Input channels for two digital incremental encoders
  - Support of single-ended TTL and differential RS422 signals
  - 24-bit position counter
  - 1.65 MHz maximum encoder line count frequency.
- A universal asynchronous receiver and transmitter (UART) to perform serial asynchronous communication with external devices, the UART interface has the following characteristics:
  - Selectable transceiver mode (RS232, RS422, RS485).
  - Baudrates of up to
    - 115.2 kBd (RS232)
    - 1 MBd (RS422/ RS485)
  - Selectable number of data bits, parity bits and stop bits
  - Software FIFO buffer of selectable size

II.3.1.3 The Slave DSP I/O Features

The ds1104's slave DSP subsystem consists of:

- A 16-bit fixed-point processor Texas Instruments TMS320F240 DSP :
  - Running at 20 MHz
  - 4Kx16 bit dual-port memory (DPMEM) used for communication with the master PPC.

The slave DSP provides the following I/O features:

- A bit I/O unit with the following characteristics:
  - A 14-bit digital I/O.
  - Direction selectable for each channel individually
  - ±13 mA maximum output current
  - TTL voltage range for input and output
- A Slave DSP Timing I/O Unit to :
  - PWM Signal Generation:
    - Single phase PWM signal generation : The PWM signal generation has the following characteristics:
      - The generation of up to four PWM signals of a period *Tp* each and with variable Duty cycles (*Thigh* (Time where PWM signal not equal to zero) /*Tp*) for each of the channels.
      - Wide range of PWM frequencies.
      - Polarity (Active high or Active low)
      - Symmetric or asymmetric PWM mode

        - Non-inverted and inverted outputs for 3-phase PWM signal generation (PWM3) with variable :
            - Duty cycles (*Thigh*/*Tp*) for each of the three phases.
            - Wide range of PWM frequencies
            - Deadband: For the three PWM duty cycles of PWM3 and PWMSV, you can specify one deadband value. This is the time gap between the rising and falling edges of the non-inverted and inverted PWM signals. The deadband introduces a time delay that allows complete turning off of one power transistor before the turning on of the other power transistor.
        - Non-inverted and inverted outputs for the generation of 3-phase space vector PWM signals (PWMSV) with variable :
            - Values *T1* and *T2* of the space vector
            - Sector of the space vector
            - Wide range of PWM frequencies
            - Deadband
    - Square-wave signal generation (D2F): The square-wave signal generation (D2F) provides outputs for the generation of up to four square-wave signals with variable frequencies.
    - Signal measurement:
        - PWM signal measurement (PWM2D): The slave DSP provides input channels for the measurement of the duty cycles and PWM periods *Tp* of up to four PWM signals. The PWM period length *Tp* that can be measured depends on the number of channels used for (PWM2D).
        - Square-wave signal measurement (F2D): The slave DSP provides input channels for the measurement of the frequencies of up to four square-wave signals. You can specify a minimum input frequency in the range “5 mHz ... 150 Hz” below which the measurement will return a value of 0 Hz. The maximum frequency that can be measured depends on the number of channels used for F2D.
- A Serial Peripheral Interface (SPI): The slave DSP of the ds1104 features a serial peripheral interface (SPI). The SPI can be used to perform high-speed synchronous communication with devices connected to the ds1104, such as an external ADC converter.

  The SPI transfers serial bit streams of selectable length (1 ... 8 bits) with a transfer rate (78,125 Baud ... 1.25 MBaud (slave mode) or 2.5 MBaud (master mode)) from and to external devices. The slave mode is when the transfer rate is specified (via a special clock signal) by the external device and the master mode is when the transfer rate is specified by the SPI. The data transfer can be done on either the rising or the falling edge of the clock signal, with or without delay.

Notes:

- Due to the board's limited number of I/O pins, the pins used to provide the bit I/O signals of the slave DSP are shared with other I/O signals of the slave DSP.
- Avoid any connection of power supply to the DSP board especially when it is off.
- Verify the signals ranges before feeding them to the card (ex: for the ADC…).

For more detailed information about the different options and practical functionalities consult the ds1104 DSP Card Manual.

II.3.1.4 The PWM Signal Generation

PWM signal generation is crucial to many motor and motion control applications. PWM signals are pulse trains with fixed frequency and magnitude and variable pulse width. There is one pulse of fixed magnitude in every PWM period *(Tp)*, however, the width of the pulses changes from period to period according to a modulating signal. When a PWM signal is applied to the gate of a power transistor, it causes the turn-on/turn-off intervals of the transistor to change from one PWM period to another, according to the same modulating signal. The frequency of a PWM signal is usually much higher than that of the modulating signal, or the fundamental frequency, so that the energy delivered to the motor and its load depends mainly on the modulating signal. Concerning the 3-phase and space vector PWM signals generation, these will not considered here, **Figure 2.6**.

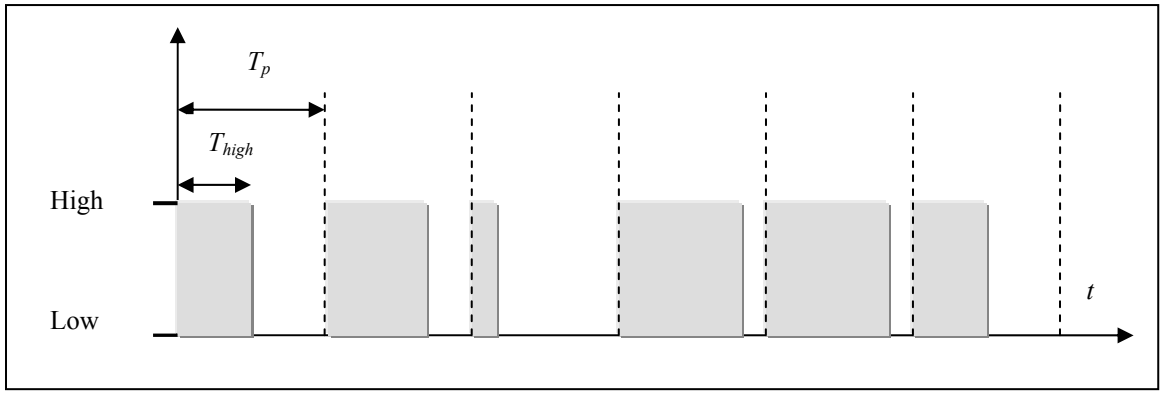


Figure 2.6: Asymmetric PWM signal (active high).

II.3.2 The Software Platform

Here we will illustrate the development steps to be followed in implementing an application (Control Algorithm, Digital Signal Processing application…), download it to the DSP card and experiment with it in real time.

II.3.2.1 The Development Steps

Before starting implementing applications, the following conditions must be satisfied:

- The ds1104 DSP Card must be installed properly in a PCI slot inside The PC (Read and ensure compliance with the safety precautions stated in the manual).
- The Matlab, Simulink, and Real-Time Workshop software with the proper versions according to the DSP Card software package must be properly installed before the installation of the DSP Card software package (Compatibility software list is available in the DSP Card CD).
- A C compiler compatible with the DSP card is required to be installed (the ds1104 uses The Microtec PowerPC C Compiler which is provided with the DSP Card software package).

Install the ds1104 Software package following these steps:

- To activate dSPACE licenses, plug the execution key (dongle) into the parallel port of the host PC.
- Insert the Key-Disk into the floppy-disk drive
- Start the installation and follow the directives.

After completing successfully these steps, we are ready to implement, build and experiment with applications.

II.3.2.1.1 The Implementation

The first step is to implement your control algorithm, two methods are provided to do this:

- Develop your application directly in Simulink as usual, and when needed, embed the blocks provided by the dSPACE's Real Time Interface (RTI) which are found in a library in Simulink which is automatically added after the installation of DSP Card Software package, **Figure 2.7** shows the RTI library.

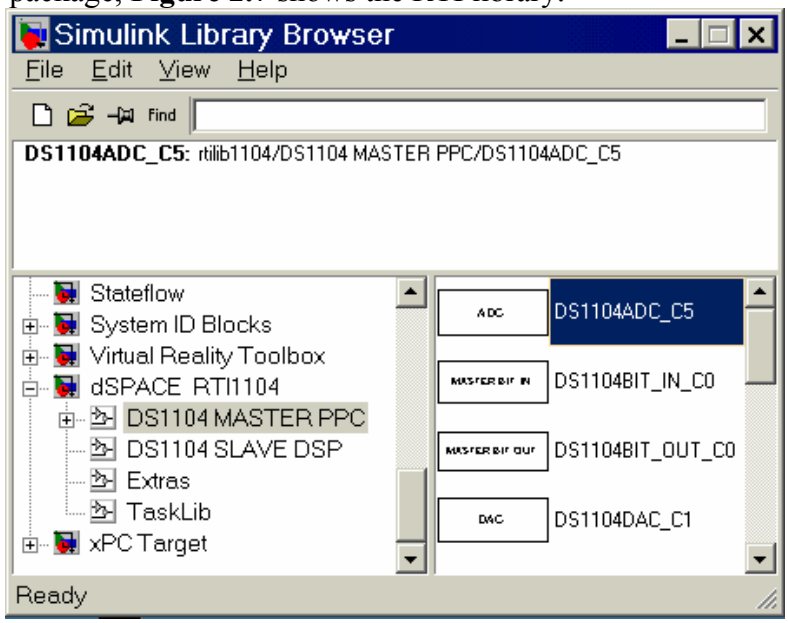


Figure 2.7: The RTI library.

The RTI Simulink blocks permit us to access most of the functions and all the inputs and the outputs existing and provided by DSP Card ex:

- Get the input of a DAC converter of the DSP Card **Figure 2.8A**.
- Give the output to an ADC converter **Figure 2.8B**.
- Generate a PWM signal by providing its DutyCycle **Figure 2.8C** …

Block options like choosing the ADC number in the ADC block or the frequency of the PWM signal in the PWM generation block can be accessed and specified by clicking twice on it, **Figure 2.9** shows the settings of a DAC converter.

We can implement applications that don't use the RTI blocks and test it in the DSP Card.

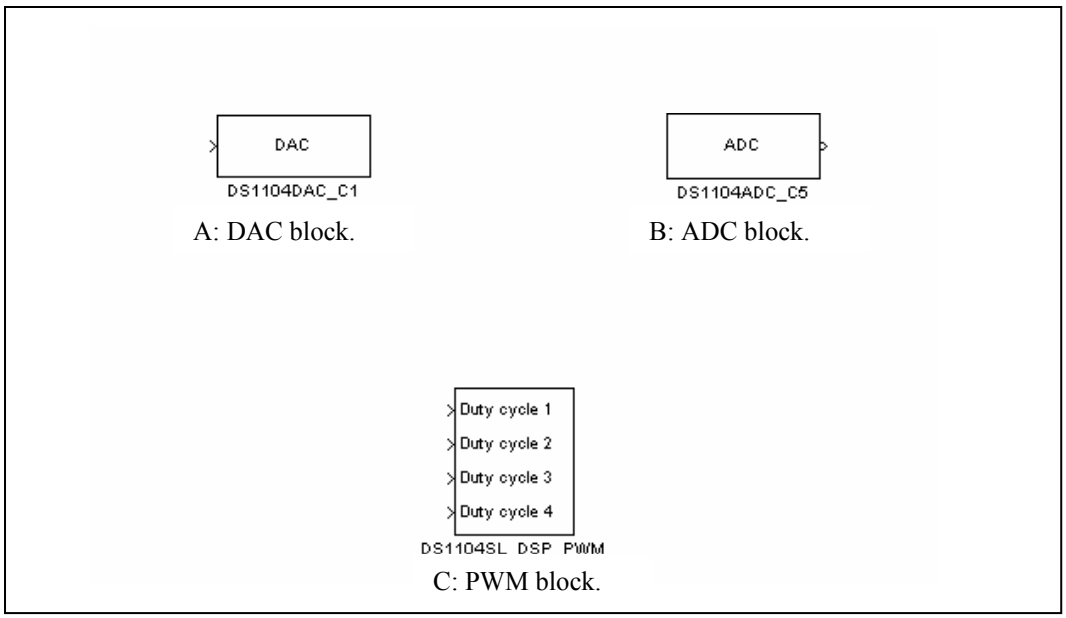


Figure 2.8: The RTI blocks.

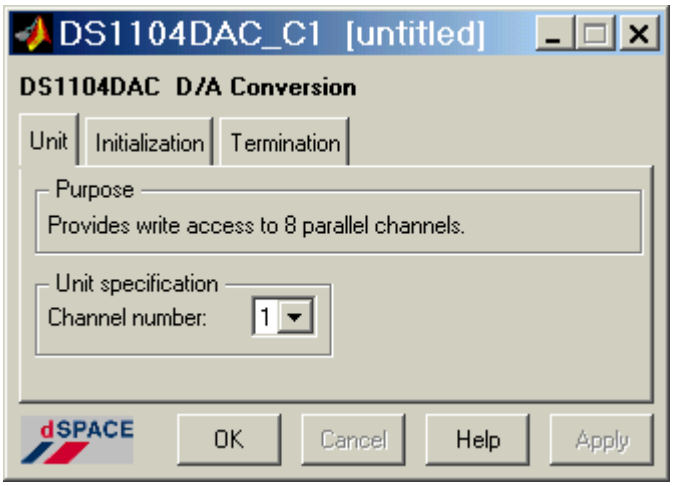


Figure 2.9: The DAC Settings dialog box.

- Develop your application using RTLib's functions and handcode your application directly in C and by the use of special instructions like “*down1104 -file_name-”* you can compile, link and download the application to the DSP Card and experiment with it. This method will not be considered here.

After we have completed the design process of our model in Simulink and we are ready to download it to the DSP Card, before that, we have to set the simulation parameters and verify the Real Time Workshop configuration in the simulation parameters dialog box, as shown in the **Table 2.1** in the Solver page:

Simulation time frame	
Start time	0
Stop time	inf
Solver options frame	
Type	Fixed-step "ode1"
Mode	"SingleTasking"
Fixed step size	0.001 ... 0.01

Table 2.1: Simulation parameters.

In the advanced page, we have to set -*off* - the Block reduction option.

For the RTW page, we check if the system target file is *rti1104.tlc*, the template makefile is *rti1104.tmf*, and the make command is *make_rti* are specified which are the default settings when the DSP software is first installed. If necessary, change the entries.

Now, we are ready to build and download the application to the DSP Card, Click the Build button.

II.3.2.1.2 The Build Process

When you initiate the build process of a simulink model “<*model*>.*mdl”*, a number of utilities are invoked sequentially to build the real-time program and download it to the hardware. The build process can be divided into two separate phases:

- Code generation phase: In this phase all the C code needed for the real-time application is generated. The *make_rti* command is started and performs the following steps sequentially:

a) The Real-Time Workshop generates an intermediate file, <*model*>.*rtw*.
b) The Target Language Compiler (TLC) is invoked to generate the C code for the model <*model*>.*rtw* and the dSPACE I/O blocks used in the model.
c) A makefile called <*model*>.*mk* is created and it holds all the commands necessary to build the real-time program from the Simulink model and download it to the hardware.

- Compilation phase: In this phase the C sources are translated into the final application, which can be executed on the dSPACE real-time hardware. The <*model*>.*mk* makefile is invoked by the *dsmake* make utility and this utility calls :
  a) The compiler/linker, which compiles the C sources of the model and links the object files and libraries into an executable file.
  b) The dSPACE loader program, which downloads the executable file to the real-time processor and starts its execution.

Several files are created during the build process in the “<*model*>.*mdl”* directory, some of them are needed only during the build process, and others are needed by the controlDesk for the experimentation among them:

- <*model*>.*TRC* variable description file: provides information on the available variables in the model and how they are grouped (subsystems…)
- <*model*>.*MAP* file: maps symbolic names to physical addresses.
- <*model*>.*SDF* System description file: specifies which executable file and the corresponding variable description file <*model*>.*TRC* is downloaded onto the processor memory.
- <*model*>.*ppc* compiled object files.

The <*model*>.*SDF* and <*model*>.*ppc* files are of a special importance, since they can be loaded directly onto the processor causing the hole application to be loaded and started as it will be shown after, so we can start the real time simulation of a simulink model just by loading its *sdf* or *ppc* file as long as it has not been changed.

II.3.2.1.3 The Experimentation

Experiment and Test is the phase that follows the implementation of your control algorithm on your dSPACE real-time board. ControlDesk, the dSPACE's well-established experiment software, provides all the functions to control, monitor and automate experiments and make the development of controllers more efficient.

Here we will illustrate the essential steps in order to use efficiently ControlDesk to experiment with real time applications.

The ControlDesk window contains three main regions in the default settings, **Figure 2.10**, The Navigator (upper left corner), the general work area (upper right area) and the Tool window (bottom area).

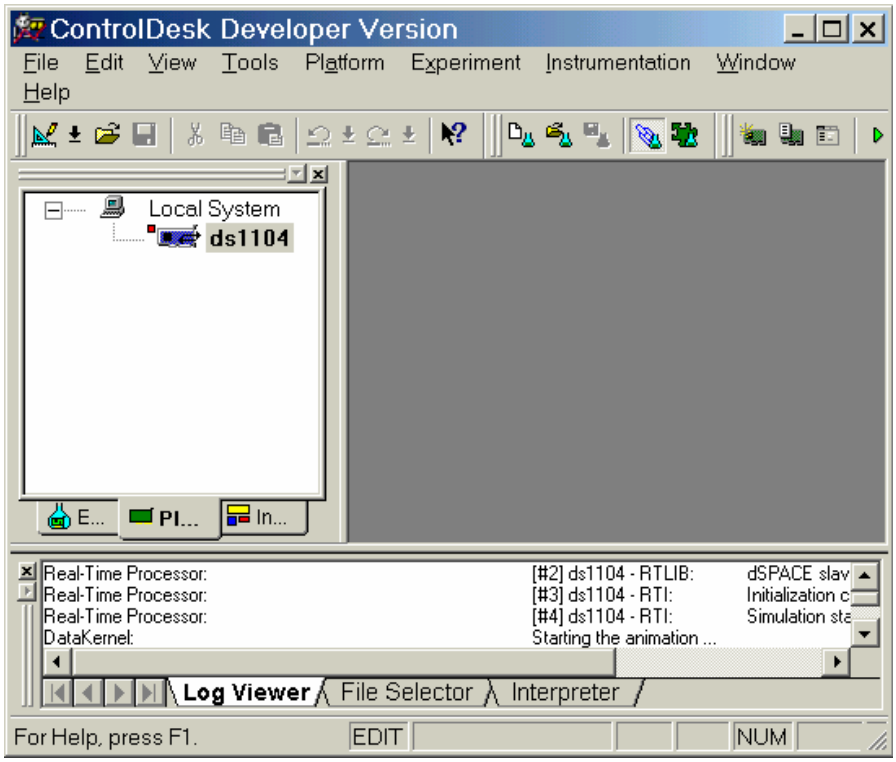


Figure 2.10: The ControlDesk window.

The Navigator has three tabs: Experiment, Instrumentation and Platform.

- The platform tab shows the platforms to which ControlDesk is connected.
- The instrumentation tab shows open “Layouts” (graphical user interfaces created by the user) and the graphical instruments associated with those layouts.
- The Experiment tab shows the current opened experiment and the associated files.

The Tool window has several tabs, the most important being the file selector tab that allows you to browse files and drag and drop applications to the platform tab. The extensions shown in the file selector are **.mdl* (Simulink model files), **.ppc* (compiled object files), **.sdf* (system description files) and **.trc* (variable description files). The files that can be dragged and dropped are the **.ppc* or **.sdf* and they have the same effect.

The general work area is used to display the layouts.

When we want to experiment with a real time application we have to:

a) Create a new experiment: To create a new experiment, click on the file menu in ControlDesk and choose “New Experiment”. You will see the window in **Figure 2.11**. Fill the required fields like the name and the working directory and the optional documentation fields if needed. The experiment will be saved with a *.cdx extension. We will associate all the files that we work with to this experiment so that when we save the experiment and close it, all the layouts (graphical user interfaces) and compiled files from Simulink will be linked to the experiment and loaded with it if we choose to open it later.

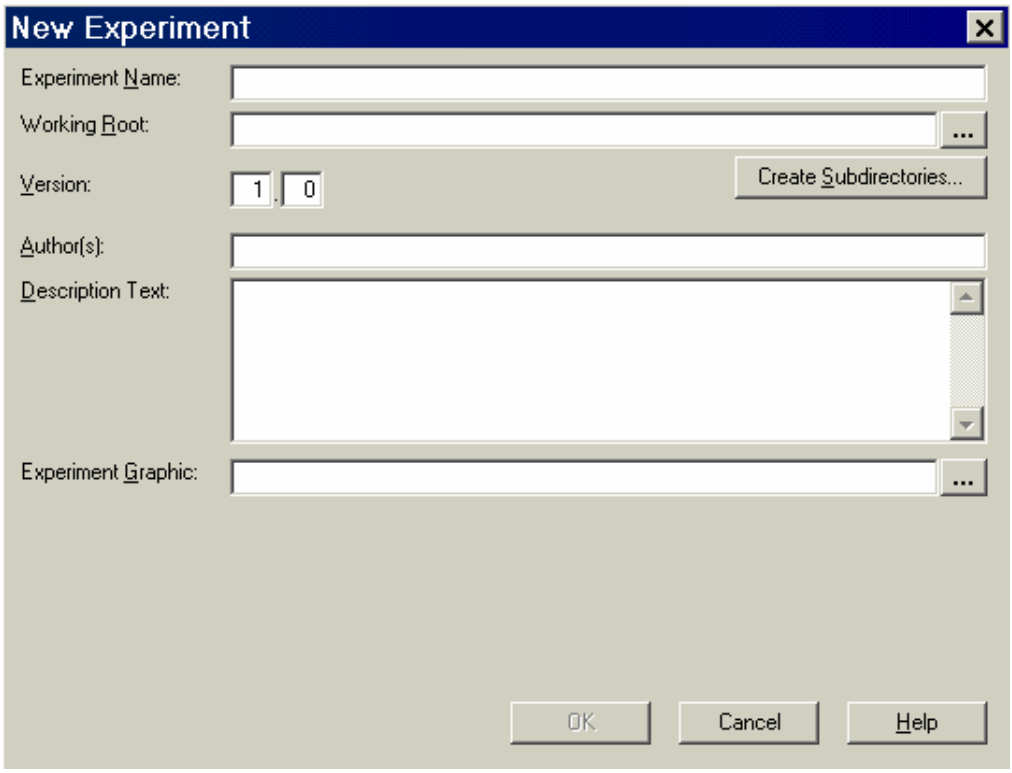


Figure 2.11: New Experiment dialog box.

When you first download an application to hardware, a tab for this application will appear in the tool window tabs, allowing you to access to the application files, variables, parameters…which are needed in experimentation.

b) Create a Graphical User Interface: In order to visualize the real-time experiment and make it interactive, it is necessary to build an instrument panel in a layout and associate it with the experiment. In order to do so, click on “File” in ControlDesk, then click on “New” and select “Layout”. The general work area of ControlDesk will appear with a new window as shown in **Figure 2.12**.

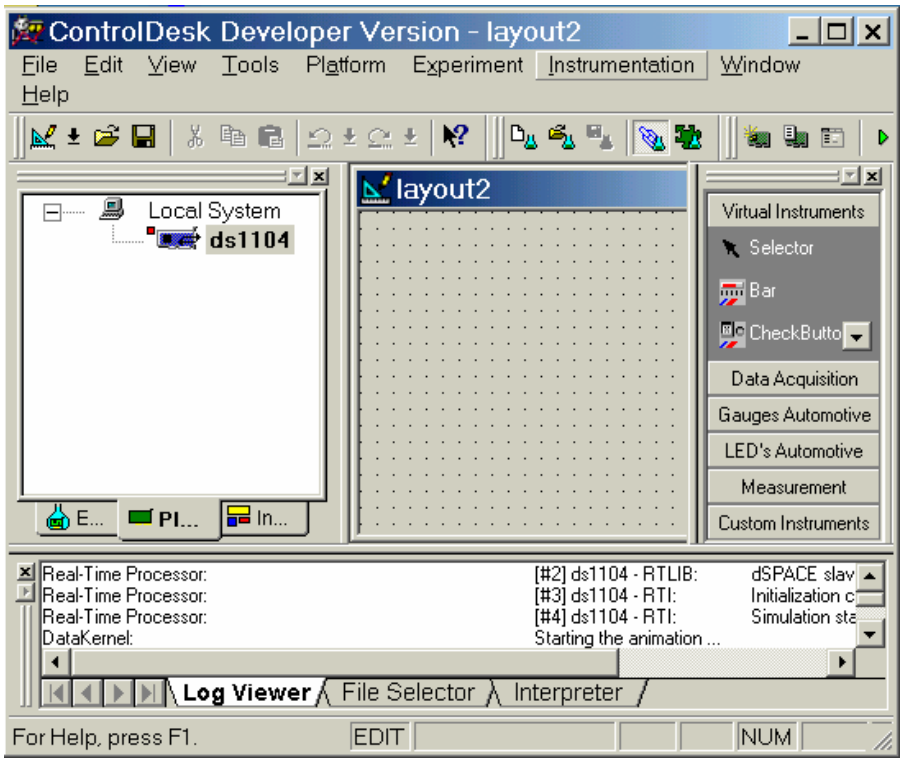


Figure 2.12: The ControlDesk GUI.

Note the Virtual Instruments panel on the right. It allows you to select from various instruments and then draw the instruments box, arrange and align them in the Layout area. ControlDesk Standard provides a set of powerful instruments. They are designed to monitor and/or control simulator variables interactively and display data captures. They are grouped according to their functionalities (Virtual instruments: used to display or enter values to the connected variables - data acquisition instruments: used to display captured simulation data …) **Figure 2.13**.

Figure 2.13: The Virtual Instruments.

c) Connecting instruments to simulator variables: It is done in order to visualize the signals and control the parameters of real-time applications and Simulink simulations. When you draw an instrument in the layout, it will have a red border indicating that no variable have yet been assigned to it. To assign a variable to an instrument:
   - Double-click on "Model Root" in the left half of the Tool window.
   - Double-click on "the block name" under "Model Root" which we want to see the output "Out1"or input "In1". The variables that can be changed in real time are associated with a variable called **P** which must be dragged and dropped on the instruments ex: "Numeric Input" an instrument in the virtual instruments group.
   - Labelled signals in simulink are displayed under the Model Root directly.
   - You can access the properties of an instrument by double clicking on it.

ControlDesk provides three operating modes:

- Edit Mode (the default mode under which we create and edit layouts and capture data properties),
- Test Mode (used for testing the current layouts) ,
- Animation Mode (used to visualize the real-time progression of the experiment)

Switching between modes is done by clicking on one of the icons in **Figure 2.14**.

Figure 2.14: The Operating modes.

Before running the experiment, we have to verify the Capture settings in the Capture settings window **Figure 2.15** which can be accessed by a small button in the tool window. The Capture settings window allows us to modify many options; the following are the most important for us:

- Length: manages the data capture "display" in the instruments by selecting the length of the data capture "acquisition".
- Auto Repeat: Select this checkbox to repeat the capture automatically. If this checkbox is not selected, only one capture is taken at a time.
- Auto start with animation: Select this checkbox to automatically start the capture when the animation is started.
- Capture Variables: to save real time variables data into a **.mat* files in order to be analysed and compared to a reference data under matlab. The variables can be added in order to be saved by a drag and drop operation on this field.

And many other options exist; see the manual to consult them.

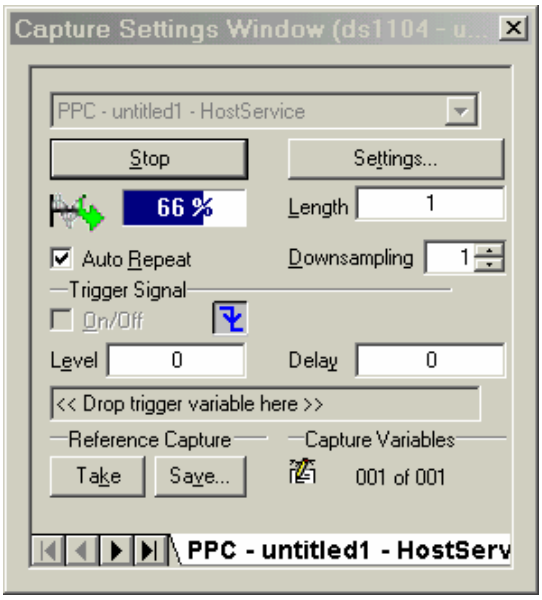


Figure 2.15: The Capture settings window.

After creating the instruments panels in the layout, connecting the variables to them and setting the data capture options, we are ready to run the experiment and pass to the animation mode to observe the real time application progression. Click on the “Play” button and immediately switch to the animation mode and watch the signals and acts on them.

When the experiment terminates, the last step is to add all the opened layouts, data connections, application files, capture settings, instruments properties… to the experiment created at the beginning so that all these files will be associated with it and will be loaded with it next time:

Click on File and select “Add all opened files”.

Click on File and select “Save Experiment”.

II.3.3 ControlDesk Automation

ControlDesk provides an important feature which permits us to perform time-consuming tasks automatically or to automate real time actions by programming them. The most important applications of this feature can be illustrated as follows:

- Automating the testing of control algorithms which are increasing in complexity continuously by generating stimulus test signals to observe the behaviour of the control algorithms (simulations or Hardware in the Loop applications).
- Analysing the results under ControlDesk with the possibility of using Matlab commands.
- Exchange data with Matlab.
- Generate test reports using Microsoft Excel and Microsoft Word.
- The possibility to use the DSP Card as a diagnosis aided tool for Industrial Computer networks containing ECUs (Electronic Control Units), PLCs (Programmable Logic Controllers), Sensors, PCs, Motors… by using the capacity to communicate with the diagnosis lines provided by these components via the DSP Card Serial Interface ports, or to simply, analyse the signals exchanged in the networks.

This Automation can be programmed with the Python programming language which is an interpreted, object-oriented, high-level programming language with these characteristics:

- The Python programs are written in the form of scripts executed under ControlDesk.

- These scripts are also used to program the interaction between the layout Instruments and between the layouts by applying specific events.

Test Automation Module is optional and it is not available with our DSP Card Software package.

II.4 Conclusion

In this chapter, first, we have illustrated the essential principals of digital control on which the DSP Card functioning is based, after that, we have introduced the hardware components composing the DSP card and all the functionalities that can be used, and finally, we have shown the procedure to be followed in order to install the DSP Card, implement the real time application, download it to the DSP processor, and experiment with it.

The ds1104 is a very useful tool for different applications like testing control algorithms on external plants ex: DC/AC motors, or the possibility to experiment with different digital signal processing algorithms including operations like filtering, convolution…, and in addition to that, the communication instruments provided with the ds1104 permit the use of the DSP Card as a communication unit that can be plugged in a computer network and used as a diagnosis or monitoring tool.

In the next chapter, we will use the theory introduced in chapter I to model the mechanical structure built especially for this work.

CHAPTER III
THE APPLICATION

III.1 The Three Degrees of Freedom (3dof) Manipulator

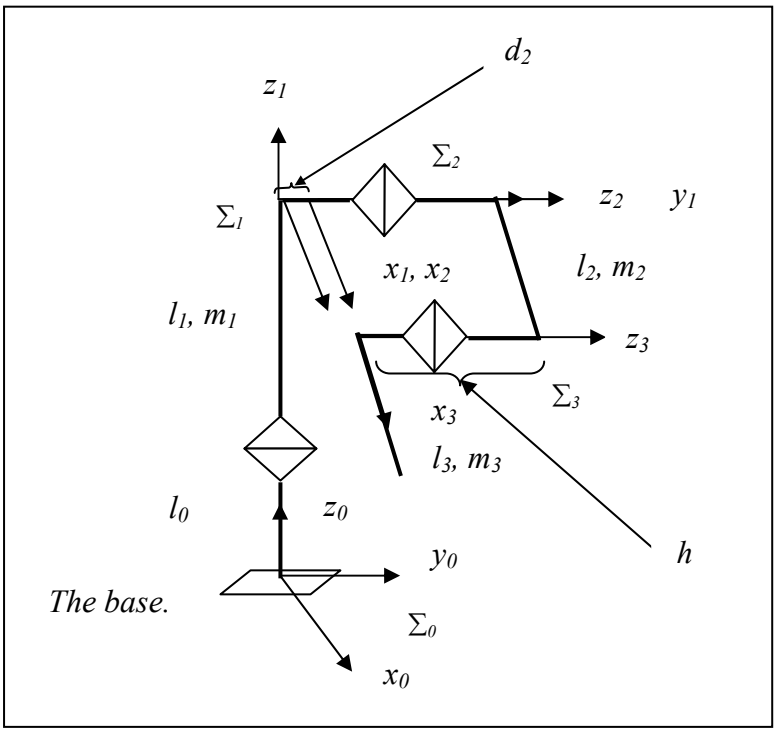


Figure 3.1: Link structure and frames.

III.1.1 The Link Parameters

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	*0*	*0*	l_0+l_1	θ_1
2	*0*	*-90°*	d_2	θ_2
3	l_2	*0*	*0*	θ_3

Table 3.1: The link parameters.

As shown in figure 3.1 the following will be considered:

θ_i : the joint angle of joint *i*,
m_i : the mass of link *i*,
I_i :the moment of inertia of link *i* about the axis that passes through the center of mass and is :parallel to the *z* axis,
l_i :the length of link *i*,
lg_i :the distance between joint *i* and the center of mass of link *i* (the center of mass is assumed to be on the straight line connecting the two joints).

III.1.2 Kinematics
a) The Homogenous Transforms

$$^{*}\ {}^{0}T_1 = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & l_0 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (III\text{-}1a)$$

$${}^{1}T_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -S_2 & -C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (III\text{-}1b)$$

$${}^{2}T_3 = \begin{bmatrix} C_3 & -S_3 & 0 & l_2 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (III\text{-}1c)$$

b) DKP of the Manipulator

Using equations *(*I-*11)* and *(*I-*12)* and knowing that ${}^{3}r = {}^{3}p_E = [l_3,\ 0,\ 0,\ 1]^T$:

$$rx = \frac{1}{2}\, l_3\, C_{1-2-3} + \frac{1}{2}\, l_3\, C_{123} + \frac{1}{2}\, l_2\ C_{1-2} + \frac{1}{2}\, l_2\, C_{12} - S_1\, d_2 \qquad (III\text{-}2a)$$

$$ry = \frac{1}{2}\, l_3\ S_{123} + \frac{1}{2}\, l_3\ S_{1-2-3} + \frac{1}{2}\, l_2\ S_{12} + \frac{1}{2}\, l_2\, S_{1-2} + C_1\, d_2 \qquad (III\text{-}2b)$$

$$rz = -\, S_{23}\, l_3 - S_2\, l_2 + l_1 + l_0 \qquad (III\text{-}2c)$$

where *rx, ry* and *rz* are the position coordinates of the end-effector expressed in frame $\sum_0$.

c) IKP of the Manipulator

Using the algebraic approach explained in section *(***I.2.5***)* and introducing the vector position ${}^{0}\boldsymbol{r} = [rx,\ ry,\ rz]^T = {}^{0}p_E$, we obtain the following two sets of equations:

$$({}^{0}T_2)^{-1}\ {}^{0}r = {}^{2}T_3\ {}^{3}r \qquad (III\text{-}3)$$

$$C_1\ C_2\ rx + S_1\, C_2\ ry - S_2\ rz + S_2\, l_1 + S_2\, l_0 = C_3\, l_3 + l_2 \qquad (III\text{-}4a)$$

$$-\, C_1\, S_2\ rx - S_1\, S_2\ ry - C_2\ rz + C_2 l_1 + C_2 l_0 = S_3\, l_3 \qquad (III\text{-}4b)$$

$$-\, S_1\ rx + C_1\ ry - d_2 = 0 \qquad (III\text{-}4c)$$

and if we further consider :

$$({}^{0}T_1)^{-1}\ {}^{0}r = {}^{1}T_3\ \ {}^{3}r \qquad (III\text{-}5)$$

$$C_1\ rx + S_1 ry = C_{23}\, l_3 + C_2\, l_2 \qquad (III\text{-}6a)$$

$$-\, S_1\ rx + C_1\ ry = d_2 \qquad (III\text{-}6b)$$

---

* In this chapter, we will use the following notation:
$C_1 = \cos(\theta_1),\ S_1 = \sin(\theta_1),\ C_{12} = \cos(\theta_1 + \theta_2),\ and\ \ S_{2\times 23} = \sin(2\theta_2 + \theta_3)\ and\ so\ on.$

$$rz - l_1 - l_0 = -S_{23}\, l_3 - S_2\, l_2 \qquad (III\text{-}6c)$$

since the variables θ_i are fairly well separated in equations (III-*4)* and (III-*6)* they can be easily found. For example we can obtain θ_1 from equations (III-*4c)* or equation (III-*6b)*:

$$\theta_1 = a\tan 2(-rx,\, ry) \pm a\tan 2\,(\sqrt{rx^2 + ry^2 - d_2^2}\,,\, d_2) \qquad (III\text{-}7)$$

summing the squares of equations (III-*6a)*, (III-*6b)* and (III-*6c)* we get :

$$2l_2\, l_3\, C_3 + l_3^2 + l_2^2 + d_2^2 = rx^2 + ry^2 + rz^2 - 2\, rz\, l_1 - 2\, rz\, l_0 + l_1^2 + 2\, l_1\, l_0 + l_0^2 \qquad (III\text{-}8)$$

hence θ_3 is given by:

$$\theta_3 = \pm\, a\tan 2\,(k,\, rx^2 + ry^2 + (rz - l_0 - l_1)^2 - d_2^2 - l_2^2 - l_3^2) \qquad (III\text{-}9)$$

where

$$k = \sqrt{\{(rx^2 + ry^2 + (rz - l_0 - l_1)^2 - d_2^2 + l_3^2 + l_2^2)^2 - 2\,[(rx^2 + ry^2 + (rz - l_0 - l_1)^2 - d_2^2)^2 + l_2^4 + l_3^4]\}} \qquad (III\text{-}10)$$

for θ_2 , we obtain from equations (III-*4a)* and (III-*4b)*:

$$\theta_2 = a\tan 2\,[-(rz - l_0 - l_1)\,(l_3\, C_3 + l_2) - (C_1\, rx + S_1\, ry)\, l_3\, S_3\,,\, (C_1\, rx + S_1 ry)(l_3\, C_3 + l_2) - (rz - l_0 - l_1)\, l_3\, S_3] \qquad (III\text{-}11)$$

As you can see, one value of θ_2 will be determined for each combination of two values of θ_1 and θ_3, therefore, a choice of an appropriate combination must be done depending on the region we want the end-effector to move.

d) The Jacobian Matrix

Here we will derive the jacobian matrix of our manipulator following the procedure described in section **I.2.7**:

From the previous section we have:

$$^0p_E = \begin{bmatrix} \frac{1}{2} l_3\, C_{1-2-3} + \frac{1}{2} l_3\, C_{123} + \frac{1}{2} l_2\, C_{1-2} + \frac{1}{2} l_2\, C_{12} - S_1\, d_2 \\ \frac{1}{2} l_3\, S_{123} + \frac{1}{2} l_3\, S_{1-2-3} + \frac{1}{2} l_2\, S_{12} + \frac{1}{2} l_2\, S_{1-2} + C_1\, d_2 \\ -S_{23}\, l_3 - S_2\, l_2 + l_1 + l_0 \end{bmatrix} \qquad (III\text{-}12)$$

and using equation (I-*34)* with the homogenous transforms in equations (III-*1a)*, (III-*1b)* and (III-*1c)* we get:

$$^0p_1 = [\,0,\, 0,\, l_1 + l_0\,]^T \qquad (III.13a)$$

$$^0p_2 = [-S_1\, d_2\,,\, C_1\, d_2\,,\, l_1 + l_0\,]^T \qquad (III.13b)$$

$$ {}^0p_3 = \left[\frac{1}{2} l_2 C_{1-2} + \frac{1}{2} l_2 C_{12} - S_1 d_2 \, , \, \frac{1}{2} l_2 S_{12} + \frac{1}{2} l_2 S_{1-2} + C_1 d_2 \, , \, - S_2 l_2 + l_1 + l_0 \right]^T \quad (\text{III}.13c) $$

$$ {}^0z_1 = [0, 0, 1]^T \quad (\text{III}.13d) $$

$$ {}^0z_2 = [-S_1, C_1, 0]^T \quad (\text{III}.13e) $$

$$ {}^0z_3 = [-S_1, C_1, 0]^T \quad (\text{III}.13f) $$

using equation *(*I-*35)*:

$$ {}^0p_{E,1} = {}^0p_E - {}^0p_1 = \begin{bmatrix} (\frac{1}{2} C_{1-2-3} + \frac{1}{2} C_{123}) l_3 + \frac{1}{2} l_2 C_{1-2} + \frac{1}{2} l_2 C_{12} - S_1 d_2 \\ (\frac{1}{2} S_{123} + \frac{1}{2} S_{1-2-3}) l_3 + \frac{1}{2} l_2 S_{12} + \frac{1}{2} l_2 S_{1-2} + C_1 d_2 \\ - S_{23} l_3 - S_2 l_2 \end{bmatrix} \quad (\text{III-}14a) $$

$$ {}^0p_{E,2} = {}^0p_E - {}^0p_2 = \begin{bmatrix} (\frac{1}{2} C_{1-2-3} + \frac{1}{2} C_{123}) l_3 + \frac{1}{2} l_2 C_{1-2} + \frac{1}{2} l_2 C_{12} \\ (\frac{1}{2} S_{123} + \frac{1}{2} S_{1-2-3}) l_3 + \frac{1}{2} l_2 S_{12} + \frac{1}{2} l_2 S_{1-2} \\ - S_{23} l_3 - S_2 l_2 \end{bmatrix} \quad (\text{III-}14b) $$

$$ {}^0p_{E,3} = {}^0p_E - {}^0p_3 = \begin{bmatrix} \frac{1}{2} (C_{1-2-3} + C_{123}) l_3 \\ \frac{1}{2} (S_{123} + S_{1-2-3}) l_3 \\ - S_{23} l_3 \end{bmatrix} \quad (\text{III-}14c) $$

and finally, from equation *(*I-*33)* we obtain the expression of the jacobian matrix:

$$ J = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{bmatrix} \quad (\text{III-}15) $$

such that:

$$ j_{11} = \frac{1}{2} l_3 S_{-123} - \frac{1}{2} l_3 S_{123} - \frac{1}{2} l_2 S_{1-2} - \frac{1}{2} l_2 S_{12} - C_1 d_2. $$

$$ j_{12} = -\frac{1}{2} l_3 S_{-123} - \frac{1}{2} l_3 S_{123} + \frac{1}{2} l_2 S_{1-2} - \frac{1}{2} l_2 S_{12}. $$

$$ j_{13} = -\frac{1}{2} l_3 (S_{-123} + S_{123}). $$

$$j_{21}=\frac{1}{2}l_3 C_{-123}+\frac{1}{2}l_3 C_{123}+\frac{1}{2}l_2 C_{1-2}+\frac{1}{2}l_2 C_{12}-S_1 d_2.$$
$$j_{22}=\frac{1}{2}l_3 C_{123}-\frac{1}{2}l_3 C_{-123}+\frac{1}{2}l_2 C_{12}-\frac{1}{2}l_2 C_{1-2}.$$
$$j_{23}=\frac{1}{2}l_3 (C_{123}-C_{-123}).$$
$$j_{31}=0.$$
$$j_{32}=-C_{23}l_3-C_2 l_2.$$
$$j_{33}=-C_{23}l_3.$$

e) The Determinant of the Jacobian

$$detJ=-\frac{1}{2}l_3^2 l_2 S_2+\frac{1}{2}l_3^2 l_2 S_{2\,2\times3}+\frac{1}{2}l_2^2 l_3 S_{-23}+\frac{1}{2}l_2^2 l_3 S_{23}. \quad (III\text{-}16)$$

III.1.3 Dynamics

a) The Lagrangian Formulation

The equation of motion of the manipulator is given by:

$$\tau=M(q)\ddot{q}+h(q,\dot{q})+g(q) \quad (III\text{-}17)$$

the pseudo inertia matrix $\hat{H}_i$ of link *i* is defined as:

$$\hat{H}_1=\begin{bmatrix} \frac{I_1}{2} & 0 & 0 & 0 \\ 0 & \frac{I_1}{2} & 0 & 0 \\ 0 & 0 & m_1 lg_1^2-\frac{I_1}{2} & -m_1 lg_1 \\ 0 & 0 & -m_1 lg_1 & m_1 \end{bmatrix} \quad (III\text{-}18a)$$

$$\hat{H}_2=\begin{bmatrix} m_2 lg_2^2+\frac{I_2}{2} & 0 & m_2{*}lg_2{*}h & m_2 lg_2 \\ 0 & \frac{I_2}{2} & 0 & 0 \\ m_2 lg_2 h & 0 & m_2 h^2-\frac{I_2}{2} & m_2 h \\ m_2 lg_2 & 0 & m_2 h & m_2 \end{bmatrix} \quad (III\text{-}18b)$$

$$\hat{H}_3=\begin{bmatrix} m_3 lg_3^2+\frac{I_3}{2} & 0 & 0 & m_3 lg_3 \\ 0 & \frac{I_3}{2} & 0 & 0 \\ 0 & 0 & -\frac{I_3}{2} & 0 \\ m_3 lg_3 & 0 & 0 & m_3 \end{bmatrix} \quad (III\text{-}18c)$$

where the inertia matrix *M(q)* can be derived using equation *(*I-*52)*:

$$M(q)=\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \qquad (\text{III-}19)$$

such that:

$$M_{11}=m_3 l_{g3} l_2 C_{2\times 23}+\frac{1}{2}m_3 l_{g3}^2+\frac{1}{2}m_3 l_2^2+\frac{1}{2}m_3 l_2^2 C_{2\times 2}+\frac{1}{2}m_2 l_{g2}^2 C_{2\times 2}+m_3 d_3^2-2m_3 d_3 d_2+m_3 d_2^2+I_1+\frac{1}{2}m_3 l_{g3}^2 C_{2\times 2\times 23}+\frac{1}{2}m_2 l_{g2}^2+d_2^2 m_2+m_3 l_{g3} l_2 C_3+m_2 h(h+2d_2).$$

$$M_{12}=S_2 m_2 l_{g2} d_2+m_3 l_{g3} d_2 S_{23}-m_3 l_{g3} d_3 S_{23}-m_3 l_2 d_3 S_2+m_3 l_2 d_2 S_2+S_2 m_2 l g_2 h.$$

$$M_{13}=-m_3 l_{g3} d_3 S_{23}+m_3 l_{g3} d_2 S_{23}.$$

$$M_{21}=-m_3 d_3 l_2 S_2+m_3 d_2 l_2 S_2+S_2 d_2 m_2 l_{g2}-m_3 l_{g3} d_3 S_{23}+m_3 l_{g3} d_2 S_{23}+S_2 m_2 l g_2 h.$$

$$M_{22}=I_3+m_3 l_{g3}^2+2l_2 m_3 l_{g3} C_3+m_3 l_2^2+m_2 l_{g2}^2.$$

$$M_{23}=I_3+m_3 l_{g3}^2+m_3 l_{g3} l_2 C_3.$$

$$M_{31}=-m_3 l_{g3} d_3 S_{23}+m_3 l_{g3} d_2 S_{23}.$$

$$M_{32}=I_3+m_3 l_{g3}^2+l_2 m_3 l_{g3} C_3.$$

$$M_{33}=I_3+m_3 l_{g3}^2.$$

the centrifugal and coriolis force term $h(\theta,\dot{\theta})$, using equation *(*I-*53)*, is given by:

$$h(\theta,\dot{\theta})=\begin{bmatrix} h_{11}+h_{12}+h_{13} \\ h_2 \\ h_3 \end{bmatrix} \qquad (\text{III-}20)$$

such that:

$$h_{11}=(-m_3 l_{g3} \dot{\theta}_3 l_2 S_3-m_3 l_{g3} \dot{\theta}_3 l_2 S_{2\times 2_3}-2m_3 \dot{\theta}_2 l_2 l_{g3} S_{2\times 23}-m_3 l_{g3}^2 \dot{\theta}_3 S_{2\times 2\times 23}-m_2 l_{g2}^2 S_{2\times 2} \dot{\theta}_2-m_3 \dot{\theta}_2 l_2^2 S_{2\times 2}-m_3 \dot{\theta}_2 l_{g3}^2 S_{2\times 2\times 23})\dot{\theta}_1.$$

$$h_{12}=2m_3 l_{g3} C_{23} \dot{\theta}_2 \dot{\theta}_3 d_2-m_3 l_{g3} C_{23} \dot{\theta}_3^2 d_3+m_3 l_{g3} C_{23} \dot{\theta}_3^2 d_2+\dot{\theta}_2^2 C_2 l_{g2} m_2 d_2+\dot{\theta}_2^2 m_3 d_2 C_2 l_2.$$

$$h_{13}=-\dot{\theta}_2^2 m_3 d_3 C_2 l_2-\dot{\theta}_2^2 m_3 l_{g3} d_3 C_{23}+\dot{\theta}_2^2 m_3 l_{g3} d_2 C_{23}-2m_3 l_{g3} C_{23} \dot{\theta}_2 \dot{\theta}_3 d_3+\dot{\theta}_2^2 C_2 m_2 l g_2 h.$$

$$h_2=(\frac{1}{2}m_3 l_{g3}^2 S_{2\times 2\times 23}+m_3 l_{g3} l_2 S_{2\times 23}+\frac{1}{2}m_2 l_{g2}^2 S_{2\times 2}+\frac{1}{2}m_3 l_2^2 S_{2\times 2})\dot{\theta}_1^2-2\dot{\theta}_2 \dot{\theta}_3 l_2 m_3 l_{g3} S_3-\dot{\theta}_3^2 l_2 m_3 S_3.$$

$$h_3=(\frac{1}{2}m_3 l_{g3}^2 S_{2\times 2\times 23}+\frac{1}{2}m_3 l_{g3} l_2 S_3+\frac{1}{2}m_3 l_{g3} l_2 S_{2\times 23})\dot{\theta}_1^2+\dot{\theta}_2^2 l_2 m_3 l_{g3} S_3.$$

using equation *(*I-*54)*, the gravity force term is:

$$g(\theta)=\begin{bmatrix} 0 \\ -g(m_2 C_{2 l_{g2}}+m_3 C_{23 l_{g3}}+m_3 C_{2 l_2}) \\ -m_3 g C_{23 l_{g3}} \end{bmatrix} \qquad (\text{III-}21)$$

b) The Newton-Euler Formulation

Here we will apply the Newton-Euler formulation to derive the equation of motion of the manipulator, and of course, the results are exactly the same.

From equations *(*III-*1a)*, *(*III-*1b)* and *(*III-*1c)* we get the rotation matrices:

$${}^0R_1=\begin{bmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (\text{III-}22a)$$

$$ {}^{1}R_{2}=\begin{bmatrix} C_2 & -S_2 & 0 \\ 0 & 0 & 1 \\ -S_2 & -C_2 & 0 \end{bmatrix} \quad (\text{III-}22b) $$

$$ {}^{2}R_{3}=\begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{III-}22c) $$

$$ {}^{0}\hat{p}_{1}=\left[0,0,l_0+l_1\right]^{T} \quad (\text{III-}23a) $$

$$ {}^{1}\hat{p}_{2}=\left[0,d_2,0\right]^{T} \quad (\text{III-}23b) $$

$$ {}^{2}\hat{p}_{3}=\left[l_2,0,0\right]^{T} \quad (\text{III-}23c) $$

the gravitational acceleration vector:

$$ \bar{g}=\left[0,0,-g\right]^{T} \quad (\text{III-}24) $$

the inertia tensor with respect to a frame with its origin at the center of mass of the link *i*:

$$ {}^{1}I_{1}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_1 \end{bmatrix} \quad (\text{III-}25a) $$

$$ {}^{2}I_{2}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_2 \end{bmatrix} \quad (\text{III-}25b) $$

$$ {}^{3}I_{3}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \quad (\text{III-}25c) $$

we define the vectors ${}^{i}s_{i}$ from the origin of $\sum_i$ to the center of mass of each link *i*:

$$ {}^{1}s_{1}=\left[0,0,-lg_1\right]^{T} \quad (\text{III-}26a) $$

$$ {}^{2}s_{2}=\left[lg_2,0,h\right]^{T} \quad (\text{III-}26b) $$

$$ {}^{3}s_{3}=\left[lg_3,0,0\right]^{T} \quad (\text{III-}26c) $$

taking these initial conditions:

$$ {}^{0}\ddot{p}_{0}=-\bar{g}=\left[0,0,g\right]^{T} \quad (\text{III-}27a) $$

$$ {}^{0}\omega_{0}=\left[0,0,0\right]^{T} \quad (\text{III-}27b) $$

$$ {}^{0}\dot{\omega}_{0}=\left[0,0,0\right]^{T} \quad (\text{III-}27c) $$

and following the procedure listed in equations *(I-55)-(I-63)* by taking the equation part corresponding to revolute joints, we get the following equation of motion:

$$\tau = \begin{bmatrix} \tau_{11}+\tau_{12}+\tau_{13}+\tau_{14}+\tau_{15} \\ \tau_{21}+\tau_{22}+\tau_{23}+\tau_{24} \\ \tau_{3} \end{bmatrix} \qquad (III\text{-}28)$$

such that:

$$\tau_{11} = m_3\ddot{\theta}_1 C_3 + \frac{1}{2}l_2^2 m_3\ddot{\theta}_1 C_{2\times 2} + \frac{1}{2}l_2^2 m_3\ddot{\theta}_1 - l_2^2 m_3\dot{\theta}_1\dot{\theta}_2 S_{2\times 2} - l_2 m_3 lg_3\dot{\theta}_1\dot{\theta}_3 S_{2\times 23} - l_2 m_3 lg_3\dot{\theta}_1\dot{\theta}_3 S_3$$

$$\tau_{12} = d_2 C_2 m_3\dot{\theta}_2^2 l_2 + d_2 S_2 m_3\ddot{\theta}_2 lg_2 + d_2 S_2 m_3\ddot{\theta}_2 l_2 + C_2 hm_2\dot{\theta}_2^2 lg_2 + d_2 C_2 m_2\dot{\theta}_2^2 lg_2 + S_2 hm_2\ddot{\theta}_2 l_2 + C_2 hm_2\dot{\theta}_2^2 lg_2$$

$$\tau_{13} = d_2 C_2 m_2\dot{\theta}_2^2 lg_2 + S_2 hm_2\ddot{\theta}_2 lg_2 + \frac{1}{2}m_2 lg_2^2\ddot{\theta}_1 C_{2\times 2} + \frac{1}{2}m_2 lg_2^2\ddot{\theta}_1 - m_2 lg_2^2\dot{\theta}_1\dot{\theta}_2 S_{2\times 2} + m_3\ddot{\theta}_1 d_2^2 + m_2\ddot{\theta}_1 h^2$$

$$\tau_{14} = m_2\ddot{\theta}_1 d_2^2 + 2hm_2\ddot{\theta}_1 d_2 + d_2 m_3 lg_3\ddot{\theta}_3 S_{23} + d_2 m_3 lg_3\ddot{\theta}_2 S_{23} + d_2 m_3 lg_3\dot{\theta}_2^2 C_{23} + d_2 m_3 lg_3\dot{\theta}_2^2 C_{23} + 2d_2 m_3 lg_3\dot{\theta}_2\dot{\theta}_3 C_{23}$$

$$\tau_{15} = -m_3 lg_3^2\dot{\theta}_1\dot{\theta}_3 S_{2\times 2\,2\times 3} + \frac{1}{2}m_3 lg_3^2\ddot{\theta}_1 C_{2\times 2\,2\times 3} + m_3 lg_3 l_2\ddot{\theta}_1 C_{2\times 23} - 2m_3 lg_3\dot{\theta}_1\dot{\theta}_2 l_2 S_{2\times 23} - m_3 lg_3^2\dot{\theta}_1\dot{\theta}_2 S_{2\times 2\,2\times 3} + I_1\ddot{\theta}_1 + \frac{1}{2}m_3 lg_3^2\ddot{\theta}_1$$

$$\tau_{21} = I_3(\ddot{\theta}_2 + \ddot{\theta}_3) + lg_3 m_3(\ddot{\theta}_1 d_2 S_{23} - gC_{23} + \frac{1}{2}\dot{\theta}_1^2 l_2 S_{2\times 23} + \frac{1}{2}S_3\dot{\theta}_1^2 l_2 + S_3\dot{\theta}_2^2 l_2 + C_3\ddot{\theta}_2 l_2 + lg_3\ddot{\theta}_2 + lg_3\ddot{\theta}_3 + \dot{\theta}_1^2 S_{23} C_{23} lg_3)$$

$$\tau_{22} = I_2\ddot{\theta}_2 + lg_2(m_2(hS_2 + d_2 S_2)\ddot{\theta}_1 + m_2(S_2\dot{\theta}_1^2 C_2 lg_2 - C_2 g + \ddot{\theta}_2 lg_2)) + l_2(S_3 m_3(-\ddot{\theta}_1 d_2 C_{23} - gS_{23} - \frac{1}{2}\dot{\theta}_1^2 l_2 C_{2\times 23}$$

$$\tau_{23} = -\frac{1}{2}C_3\dot{\theta}_1^2 l_2 - C_3\dot{\theta}_2^2 l_2 + S_3\ddot{\theta}_2 l_2 - \dot{\theta}_1^2 C_{23}^2 lg_3\dot{\theta}_2^2 - 2lg_3\dot{\theta}_2\dot{\theta}_3 - lg_3\dot{\theta}_3^2) + C_3 m_3(\ddot{\theta}_1 d_2 S_{23} - gC_{23} + \dot{\theta}_1^2 l_2 S_{2\times 23} + \frac{1}{2}S_3\dot{\theta}_1^2 l_2$$

$$\tau_{24} = S_3\dot{\theta}_2^2 l_2 + C_3\ddot{\theta}_2 l_2 + lg_3\ddot{\theta}_2 + lg_3\ddot{\theta}_3 + \dot{\theta}_1^2 S_{23} C_{23} lg_3)).$$

$$\tau_3 = I_3(\ddot{\theta}_2 + \ddot{\theta}_3) + lg_3 m_3(\ddot{\theta}_1 d_2 S_{23} - gC_{23} + \frac{1}{2}\dot{\theta}_1^2 l_2 S_{2\times 23} + \frac{1}{2}S_3\dot{\theta}_1^2 l_2 + S_3\dot{\theta}_2^2 l_2 + C_3\ddot{\theta}_2 l_2 + lg_3\ddot{\theta}_2 + lg_3\ddot{\theta}_3 + \dot{\theta}_1^2 S_{23} C_{23} lg_3)$$

III.2 The Direct Current Motors

As a robot is supposed to realize a mechanical function such that moving and positioning the end-effector, it needs an actuator to ensure the information transmission and the energy conversion, different types of actuators exist like pneumatic, hydraulic, and electrical, here we will be interested on the electrical actuators which are in general rotating motors.

The electrical actuator performance is closely related to the environment in which it will be used: the energy converter and its control electronics, sensing mechanism and the load dynamics.

III.2.1 Advantages of Electrical Actuators

Compared to the pneumatic and hydraulic actuators, electrical actuators present some advantages among them:

- Easy to find operating energy.
- Perfect compatibility between the electronic control and the electrical nature of the actuator components.
- And due to the technological advances in the associated electronics of the motors, and the improvements in performance of the motors by the use of new materials and components, their fields of applications have been significantly enlarged.

III.2.2 Types of DC Motors

There are several different types of dc (direct current) motors that are available. Their advantages, disadvantages, and other basic information are listed below in tabular form, **Table 3.2**:

Type	Advantages	Disadvantages
Stepper Motor	-Very precise speed and position control. -High Torque at low speed.	-Expensive and hard to find. -Require a switching control circuit.
DC Motor w/field coil	-Wide range of speeds and torques. -More powerful than permanent magnet motors.	-Require more current than permanent magnet motors, since field coil must be energized. -Generally heavier than permanent magnet motors. -More difficult to obtain.
DC permanent magnet motor	-Small, compact, and easy to find. -Very inexpensive.	-Generally small. -Cannot vary magnetic field strength.

Table 3.2: Comparative illustration of DC Motors.

Almost without exception, dc permanent magnet motors are, by far, the best choice for Robotic applications. Their small size, compactness, and common availability make them a good choice on almost any machine that doesn't require a huge amount of power output.

III.2.3 Elementary Theory of DC Permanent Magnet Motors

A current-carrying conductor located in a magnetic field experiences a force proportional to: the magnitude of the flux, the current, the length of the conductor, and the (*sin*) of the angle between the conductor and the direction of the flux. For any given motor, the only two adjustable quantities are the flux and the armature current **Figure 3.2**.

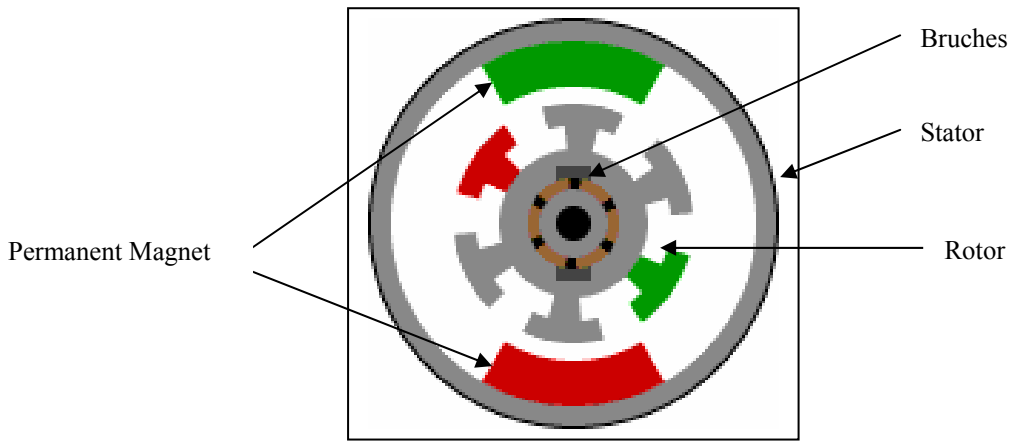


Figure 3.2: Permanent magnet DC motor.

The developed torque can be expressed as:

$$\tau_m = \frac{\rho}{2\pi\ a} N\Phi i \qquad (III\text{-}29)$$

Where the torque τ_m is in (Newton-Meter *N.M*), Φ is the magnetic flux in Weber *(Wb)*, i is the armature current in ampere *(A)*, ρ is the pole pairs number, a is the number of conductors of the motor, N is the number of active conductor. For a given motor; a , ρ , and N are constant and Since this type of motor uses a permanent magnet instead of field windings to generate the magnetic field in which the armature rotates, which simplifies the energy production and eliminates the energy loss due to the joule effect.
we can put:

$$K_t = \frac{\rho}{2\pi\ a} N\Phi \qquad (III\text{-}30)$$

K_t is the motor constant and we will get:

$$\tau_m = K_t\, i \qquad (III\text{-}31)$$

the equation of motion of a DC motor is given by:

$$\tau_m = J_m\ \ddot{\theta}_m + D_m\ \dot{\theta}_m + \tau_l \qquad (III\text{-}32)$$

where τ_l is the load torque on the motor, J_m is the moment of inertia, and D_m is the viscous-damping coefficient of the armature. Lettingθ_l be the rotational angle, J_l the moment of inertia, D_l the viscous damping coefficient of the load, and *n* be the gear reduction ratio, we obtain:

$$n\,\tau_l = J_l\ \ddot{\theta}_l + D_l\,\dot{\theta}_l \qquad (III\text{-}33)$$

electrically, the motor can be modelled by the electrical circuit in the armature, **Figure 3.3**.

we know that:

$$v_b = K_b\,\dot{\theta}_m \qquad (III\text{-}34)$$

where v_b is the back electromotive force voltage, K_b is the back electro motive force constant (generally K_b is taken to be equal to K_t), θ_m is the rotational angle.

from the armature circuit we have:

$$L\,\frac{di}{dt} + R\,i + v_b = v \qquad (III\text{-}35)$$

where v is the input voltage, L the inductance, and R the resistance of the armature.

From equations (III-*31)-(*III-*35)*, we can derive the relation between the input voltage *v* and the output angleθ_l :

$$L\,J\,\dddot{\theta}_l + (L\,D + R\,J)\ddot{\theta}_l + (RD + n^2\,K_t\,K_b)\,\dot{\theta}_l = n\,K_t\,v \qquad (III\text{-}36)$$

where

$$J = n^2 J_m + J_l \qquad (III\text{-}37)$$

$$D = n^2\,D_m + D_l \qquad (III\text{-}38)$$

hence, we get the transfer function from *v* to θ_l :

$$G(s) = \frac{n\,K_t}{s\{L\,J\,s^2 + (L\,D + R\,J)\,s + (R\,D + n^2\,K_t K_b)\}} \qquad (III\text{-}39)$$

which represents a third order system which can reduces to a second-order system if we neglect the inductance *L* which is in general very small. We can force the output θ_l to follow a given trajectory by closing the loop and designing a compensator to stabilize and tune the system response using the various methods available, one of the widely used compensator is the **PID/PD** compensator.

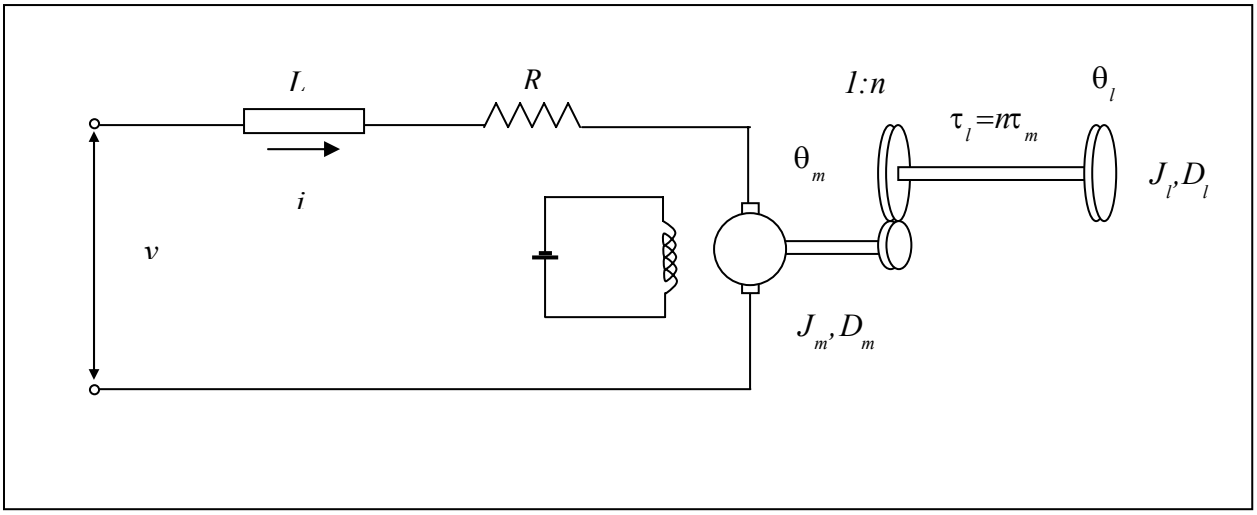


Figure 3.3: The DC motor and the load.

III.3 Conclusion

In this chapter, we have derived the kinematics and dynamics manipulator model applying the relations and equations presented in chapter I.

The link structure presented in **Figure 3.1** is derived so that to be as close as possible to the real manipulator in the laboratory and the link frames have been placed on the links in a manner to get the simplest formulas of the homogenous transforms and so that decreasing the amount of calculations in what follows.

After that, we have introduced the actuators that will be used to derive the joints of that manipulator in order to permit the end-effector to execute the affected task.

In the next chapter, we will use this manipulator model with the actuator model to simulate the laboratory manipulator and to test the performances of the controllers considered.

CHAPTER IV
SIMULATIONS AND REAL TIME APPLICATION

IV.1 Introduction

Here we will simulate the different controllers explained in chapter I "PD/PID and The Computed Torque controllers" with our model of the Puma Type three degrees of freedom manipulator under Simulink respecting these requirements:

- The simulation conditions are the same conditions required to download the simulink model to the DSP Card which means: fixed step "ode1"solver type in single tasking mode.
- Due to the fact that the dynamic effects of the motors are always present and may have a significant influence on the behaviour of the manipulator under control, the dynamic effects of the motors are always included.

The natural frequency of the manipulator is calculated according to the mechanical structure and characteristics of the manipulator and is obtained for each moving link; this natural frequency will be used to calculate the controller parameters.

The DSP Card permits the simulation of simulink models without the need to the hardware inputs or outputs. After verifying that our controller meets our requirements "time response requirement and error behaviour", the controller can be tested on the real plant for further refinement before implementing it in hardware "Microcontroller memory" using in this case the external hardware inputs and outputs in place of the plant model, this technique is called rapid control prototyping **(RCP)**. The major feature of this real-time simulation is that the simulation has to be carried out as quickly as the real system would actually run, thereby allowing you to combine the simulation and the real plant.

When your simulated controller is able to control your real plant, you typically produce the actual controller. For the final tests you usually connect the real controller to a model of the plant, which of course, has to be simulated in real-time. This way you can ensure that the controller does not contain any errors that could damage the real plant. This technique is called hardware-in-the-loop simulation **(HIL)**.

For both RCP and HIL the real-time simulation is rather important. The computing power required by real-time simulation highly depends on the characteristics of the simulated model, if it contains very demanding calculations you have to provide a lot of computing power because the timing cannot be satisfied otherwise. Our DSP Card fulfil this demand for computing power which will allow us to test our controllers without any restrictions other then the maximum sampling frequency which take into account the time constant of the model and the complexity of the control algorithm.

The simulink blocs used for the simulation are almost all realized as S-functions and this concern the manipulator model, which includes the calculation of all the matrices necessary to simulate the behaviour of the robot, as well as some controllers. To obtain the most speed of MATLAB/SIMULINK and to be able to use these blocs with the Real Time Workshop and test them in the DSP Card, all these blocs are C-MEX –files which mean S-functions written in **C**.

IV.2 The Manipulator and the DC Motors Parameters

These are the parameters used for the simulation measured experimentally for the manipulator or found in the Data sheet for the DC motor:

Manipulator parameters	
l_0	0.168m
l_1	0.498m
l_2	0.295m
l_3	0.28m
m_1	2kg
m_2	1.37kg
m_3	0.425kg
d_2	0.1m
h	0.03m

Table 4.1: Manipulator parameters.

DC motor parameters	
Moment of inertia of the rotor (J)	7.1E-6 kg.m^2
Damping ratio of the mechanical system (b)	03.5E-6 Nms
Torque constant = Electromotive force constant (K=Ke=Kt)	0.0458 Nm/Amp
Electric resistance (R)	2.49 ohm
Electric inductance (L)	2.63E-3 H
Reduction Ration (Gr)	65.5

Table 4.2: DC motor parameters.

IV.3 Useful Programs

We have developed some programs that will permit us to perform our simulations and to see the behaviour of the manipulator:

The first program is a matlab script that permits us to generate trajectories in the manipulator workspace and to see if the desired angles trajectories can be realized by plotting the manipulability measure, **Figure 4.1**.

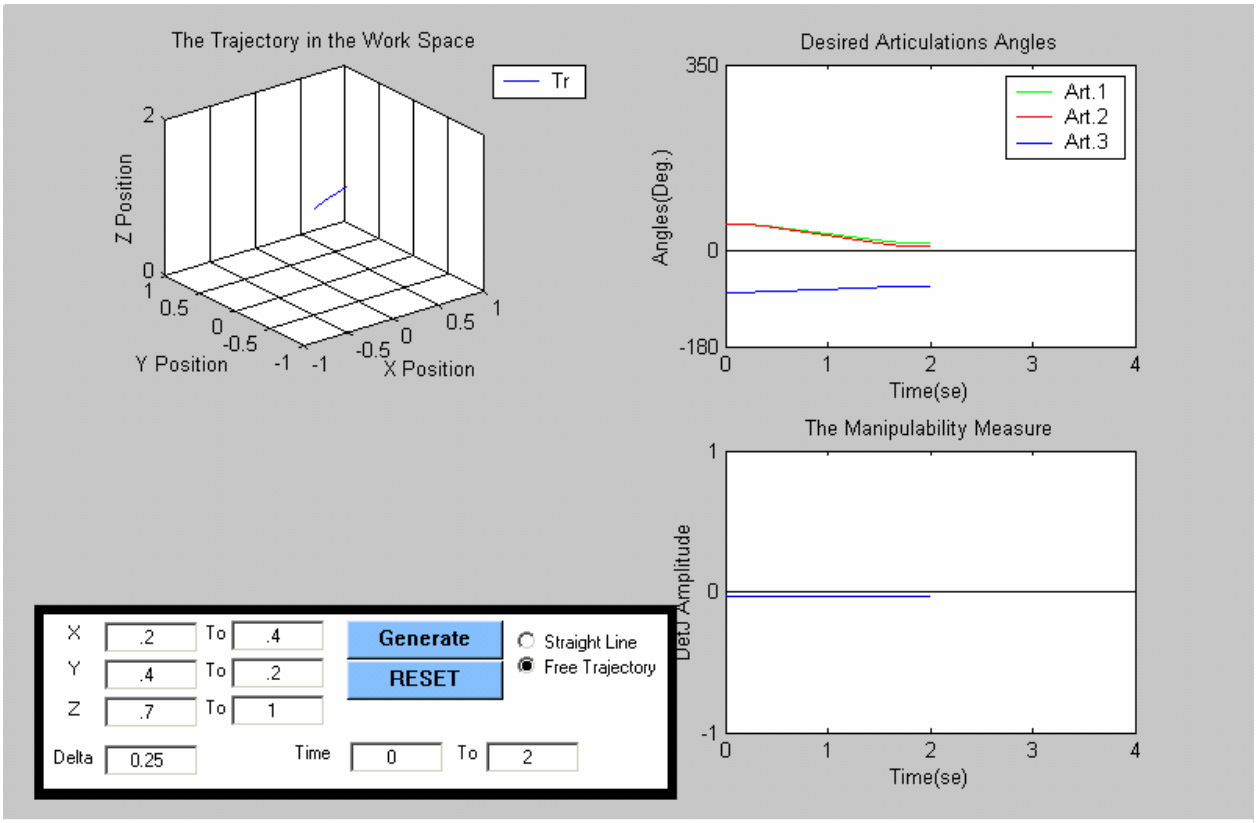


Figure 4.1: Trajectory generation script GUI.

The second program is an S-function that implements under simulink an animation program that will show us the manipulator executing the trajectory, **Figure 4.2**.

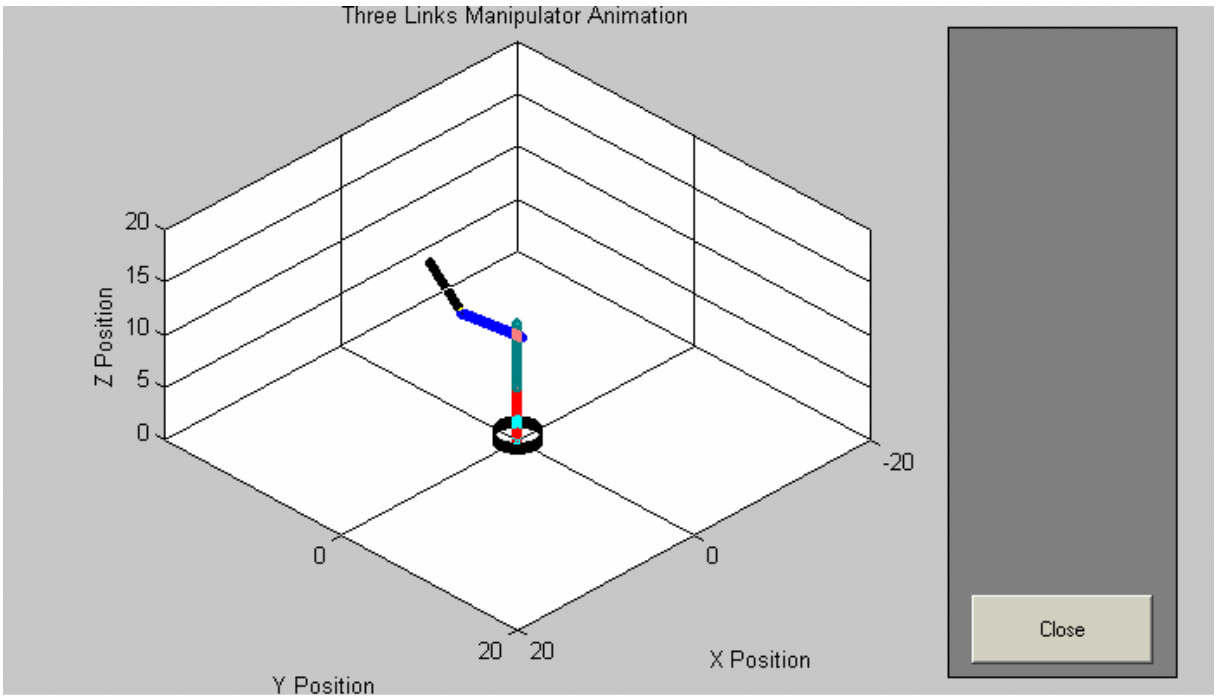


Figure 4.2: Animation script GUI.

IV.4 The Simulation Trajectories

In these simulations, we will use two types of trajectories for the end-effector:

a) A free trajectory between two points (x_0, y_0, z_0) and (x_f, y_f, z_f) in the work space:
   a. Short trajectory: between (0.2, 0.4, 0.7) and (0.4, 0.2, 1).
   b. Long trajectory: between (0.2, 0.4, 0.7) and (-0.4, -0.1, 1), **Figure 4.3**.

b) An imposed trajectory for the end-effector (a straight line) between two points (x_0, y_0, z_0) and (x_f, y_f, z_f) in the workspace:
   a. Short trajectory: between (0.2, 0.4, 0.7) and (0.4, 0.2, 1).
   b. Long trajectory: between (0.2, 0.4, 0.7) and (-0.5, 0.1, 0.7), **Figure 4.4**.

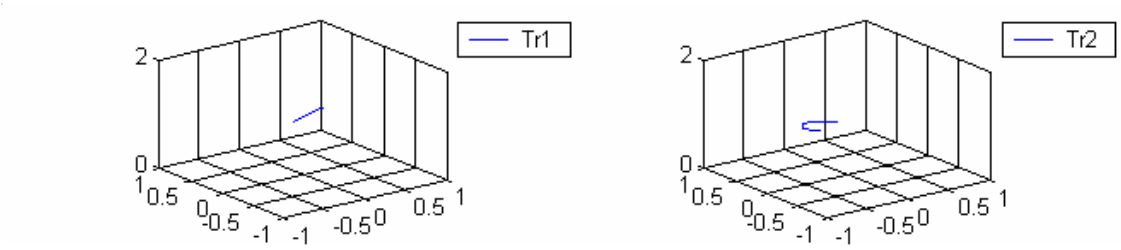


Figure 4.3: The free trajectories.

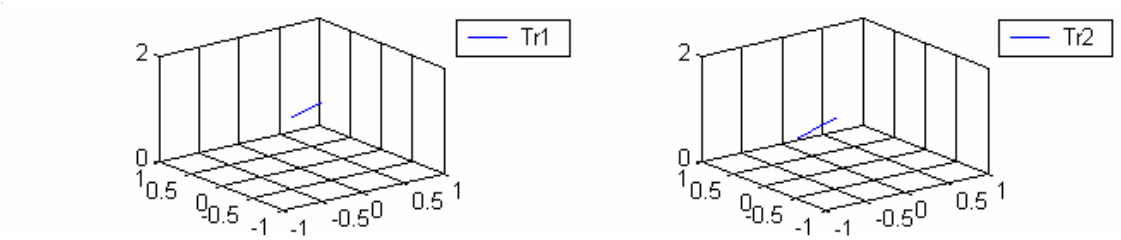


Figure 4.4: The imposed trajectories.

IV.5 The PD and PID Controllers

IV.5.1 The PD Controller

From equations *(I.75)* and *(I.76)* "including the predictive term", we can derive the following equation:

$$q / q_d = (K_d\, s + K_p) / ((K_d + G_r{}^2 V_a + V) s + K_p + (G_r{}^2 M_a + M(q))\, s^2) \qquad (IV.1)$$

considering $h(q, \dot{q}), V$ and $g(q)$ as perturbations to be compensated.
The characteristics equation is:

$$\Delta(s) = (G_r{}^2 M_a + M(q))\, s^2 + (K_d + G_r{}^2 V_a + V) s + K_p \qquad (IV.2)$$

A common choice in robotics concerning the parameters K_p and K_d is to place double real negative poles of the Transfer function which will give the fastest response without oscillations [2], so the characteristics equation can be rewritten as:

$$\Delta(s) = (G_r^2 M_{aj} + a_j)(s + \omega_j)^2 \qquad (IV.3)$$

such that $a_j = \max(M_{jj})$ of the elements of the manipulator inertia matrix $M(q)$ and neglecting the joint friction coefficient V.
We will get the following PD parameters:

$$\begin{cases} K_{pj} = (G_r^{\,2} M_{aj} + a_j)\omega_j^{\,2} \quad j = 1,..,3; \\ K_{dj} = 2(G_r^{\,2} M_{aj} + a_j)\omega_j - G_r^{\,2} V_{aj} \quad j = 1,..,3. \end{cases} \qquad (IV.4)$$

ω_j is chosen to be as large as possible without exceeding the resonant frequency of the structure in order to not destabilize the system, experimentally, a value of ω_j=50 *(rad/s)* gives the best result with our system and this for all the links for the PD as well as for the PID controllers..

It can be shown that the PD/PID controllers are asymptotically stable using the Lyapunov function definition.

IV.5.1.1 Simulations with a PD Controller

The PD controller parameters for each link considering both the motor dynamics and the manipulator dynamics, **Figure 4.5**:

K_p	K_d	$Link_i$
498.2636	19.9155	1
375.2760	14.9960	2
103.9186	4.1417	3

Table 4.3: PD Parameters.

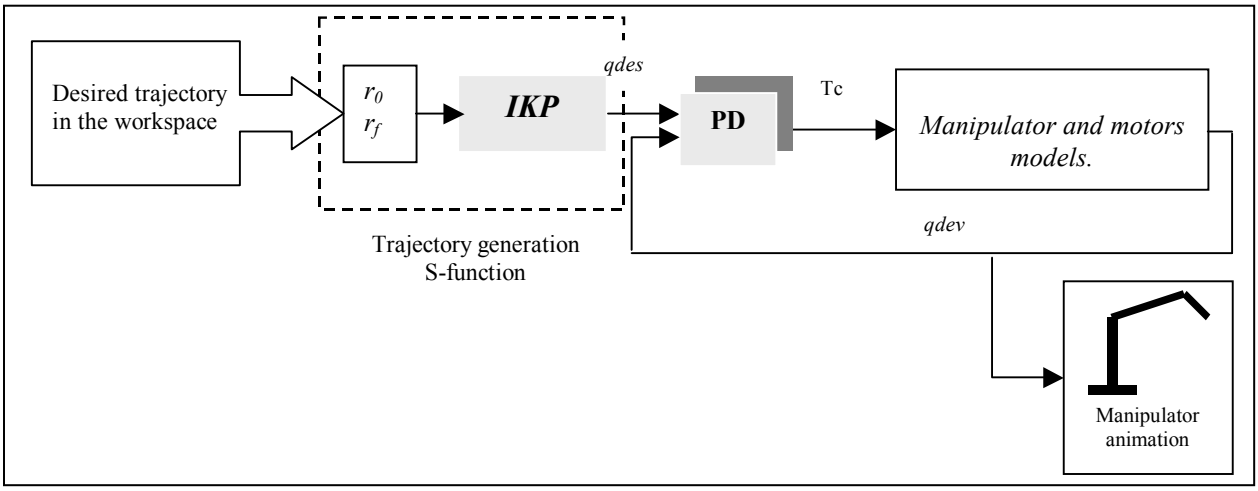


Figure 4.5: The PD controller.

Short time trajectories: Time length=0.7s and Δ=0.05s,
Long time trajectories: Time length=3s and Δ=0.25s.

Long free trajectory

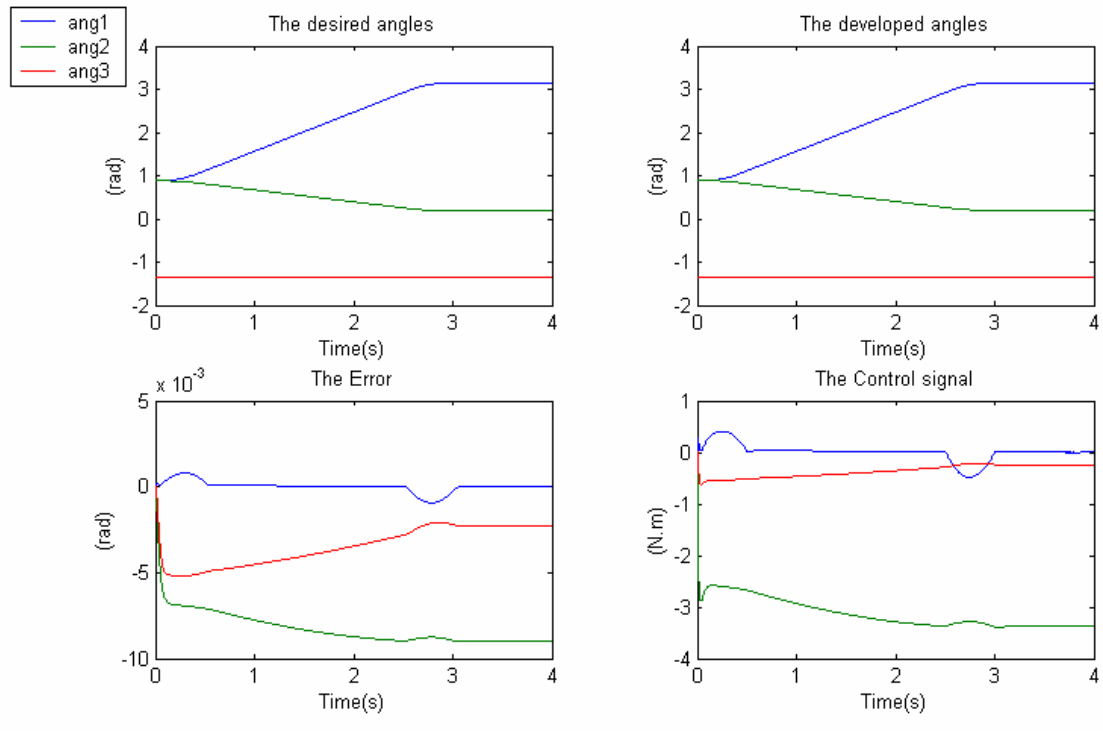


Figure 4.6

Here, we will see the effect of long and short imposed trajectories on the controller Performance since for the free trajectories, this effect is not important.

Imposed short trajectory

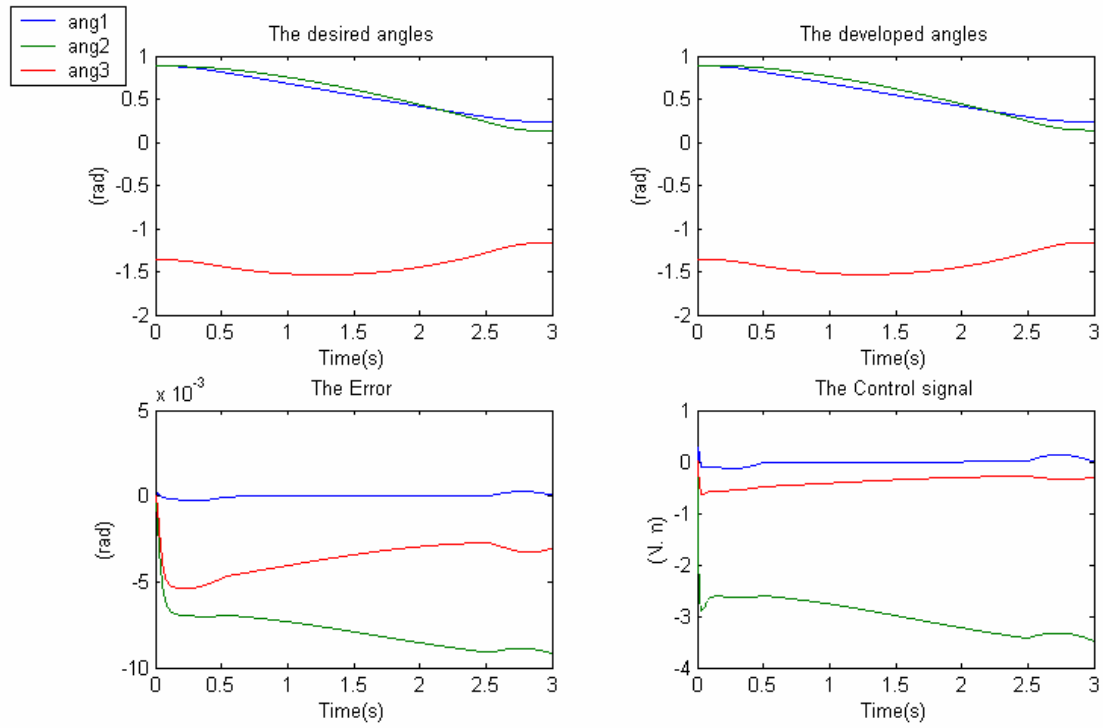


Figure 4.7

Imposed long trajectory

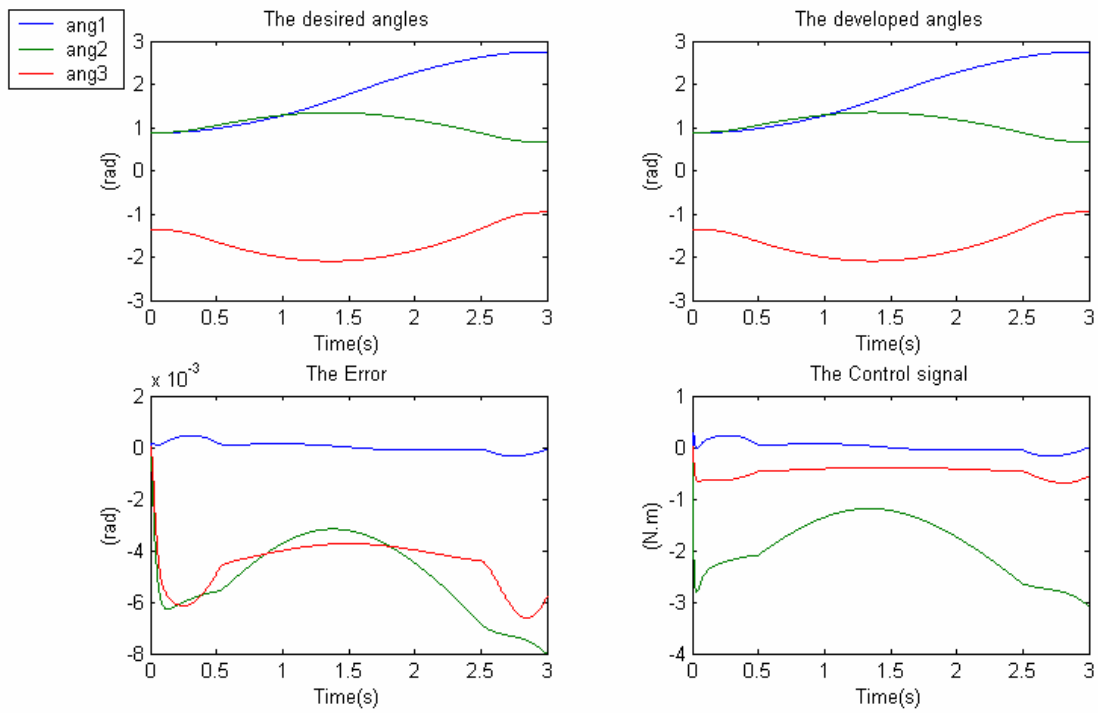


Figure 4.8

Now, we will try to illustrate the effect of short time length trajectories in the case of long trajectories -free and imposed:

Free long trajectory "short time"

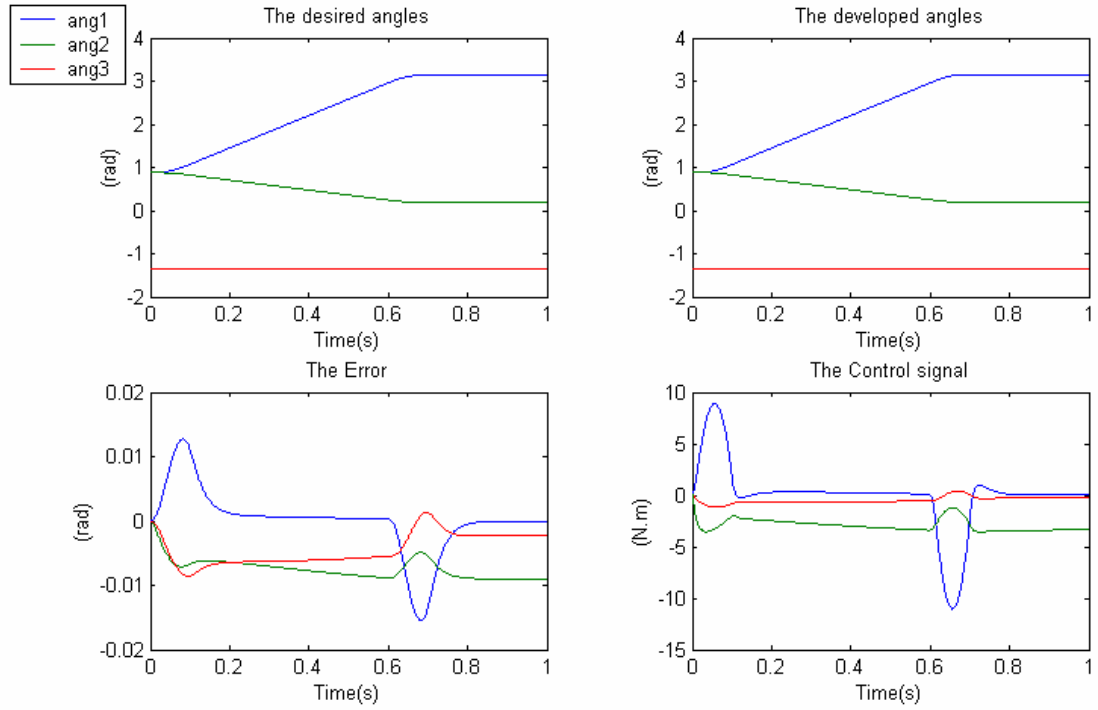


Figure 4.9

Imposed long trajectory "short time"

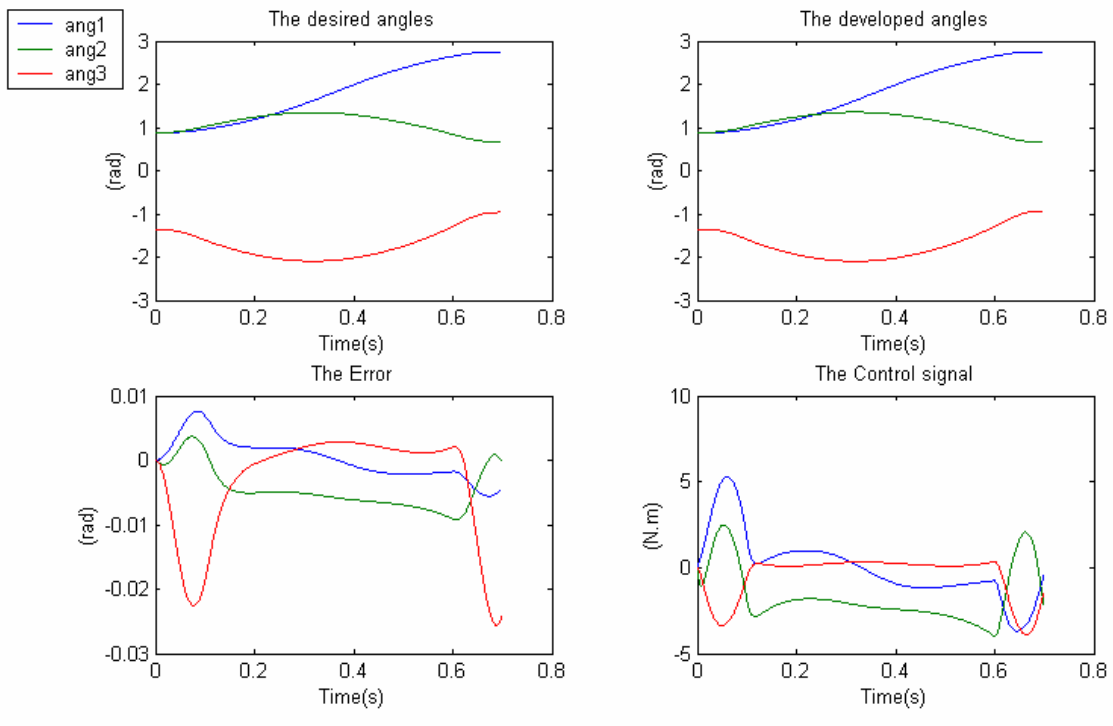


Figure 4.10

But due to practical considerations, we have to limit the control signal in a safe region determined by the hardware, here the current is limited to [-1.5A 1.5A] which will limit the control signal to [-4.72Nm 4.72Nm]:

Free long trajectory "short time"

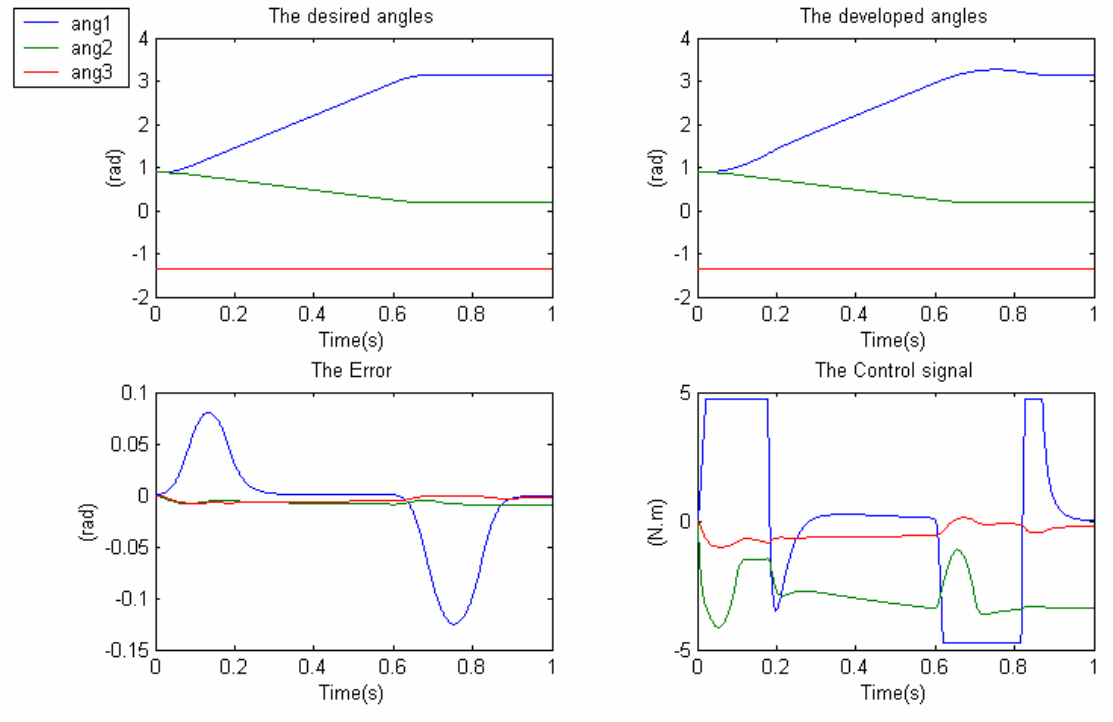


Figure 4.11

Imposed long trajectory "short time"

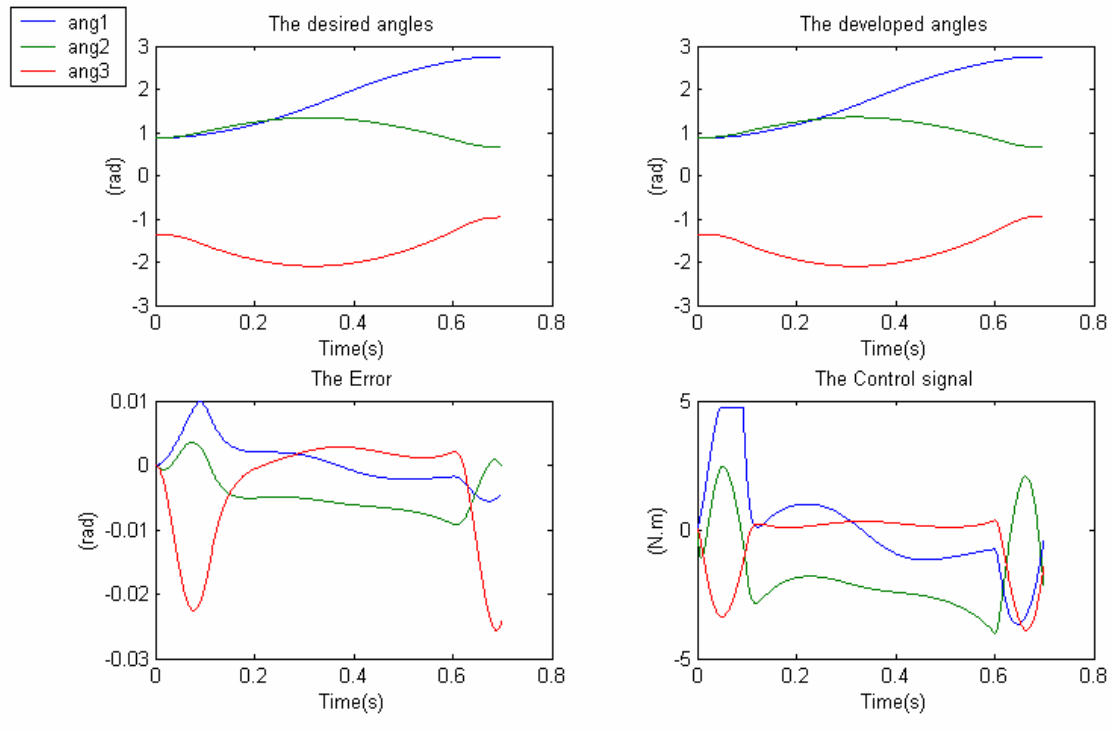


Figure 4.12

IV.5.1.2 Discussion

It can clearly be seen that the **PD** controller performances with large position and speed gains and with a relatively large reduction ratio are quite good especially for long free trajectories executed in a sufficiently long time, **Figure 4.6**, which means requiring small joints speeds.

In **Figures (4.7-4.8)**, we can note that long imposed trajectories are quite well followed and the controller performances are not affected even though the desired angles are not generated normally but according to the imposed trajectory in the workspace.

In **Figures (4.9-4.10)**, we can demonstrate that high speeds trajectories will deteriorate significantly the performance of the controller and this is clearer if we introduce some practical saturation terms as in **Figures (4.11-4.12)**.

In the next subsection, we will test the addition of an integral term on the controller performance.

IV.5.2 The PID Controller

From equations *(I-75)* and *(I-76)* "including the predictive term", with an additional integral term K_i we can derive the following equation:

$$q / q_d = (K_d\, s^2 + K_p\, s + K_i) / ((M(q) + G_r^2 M_a) s^3 + (K_d + G_r{}^2 V_a + V) s^2 + K_p\, s + K_i) \qquad (IV.5)$$

The characteristics equation is:

$$\Delta(s) = (M(q) + G_r^2 M_a) s^3 + (K_d + G_r{}^2 V_a + V) s^2 + K_p\, s + K_i \qquad (IV.6)$$

making the same choice as for the PD controller with, this time, a triple negative real poles and neglecting the joint friction coefficient V we get the PID parameters:

$$\begin{cases} K_{pj} = 3(G_r^2 M_{aj} + a_j)\omega_j{}^2 \quad j = 1,..,3, \\ K_{ij} = 3(G_r^2 M_{aj} + a_j)\omega_j - G_r^2 V_a \quad j = 1,..,3, \\ K_{dj} = (G_r^2 M_{aj} + a_j)\omega_j^3 \quad j = 1,..,3. \end{cases} \qquad (IV.7)$$

IV.5.2.1 Simulations with a PID Controller

The PID controller parameters for each link considering both the motor dynamics and the manipulator dynamics, **Figure 4.13**:

K_p	K_i	K_d	$Link_i$
1.4948e+003	2.4913e+004	29.88	1
1.1258e+003	1.8764e+004	22.5	2
311.7558	5.1959e+003	6.22	3

Table 4.4: PID Parameters.

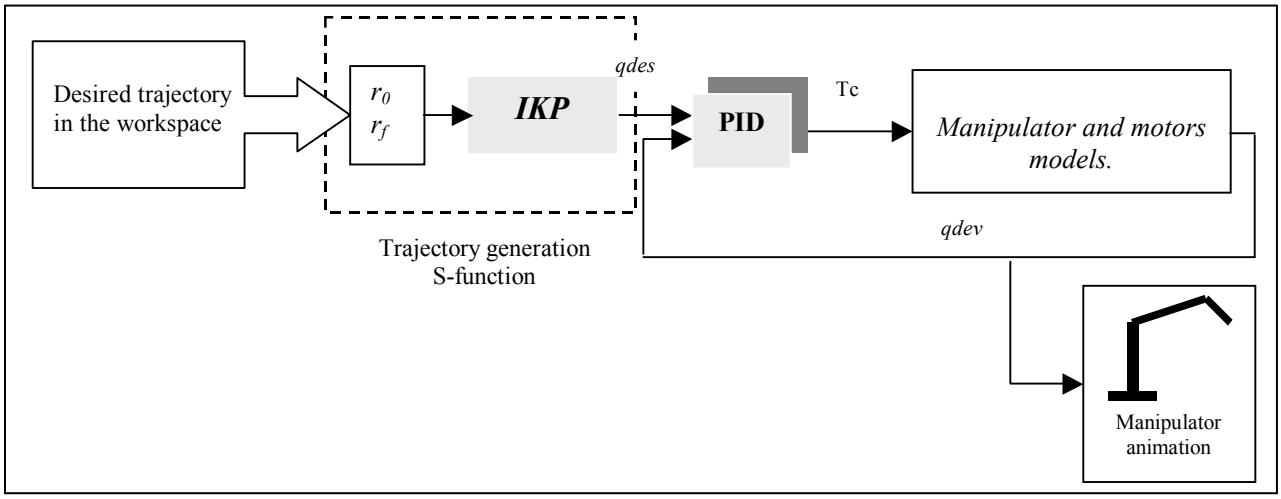


Figure 4.13: The PID controller.

Short time trajectories: Time length=0.7s and Δ=0.125s,
Long time trajectories: Time length=3s and Δ=0.25s.

Long free trajectory

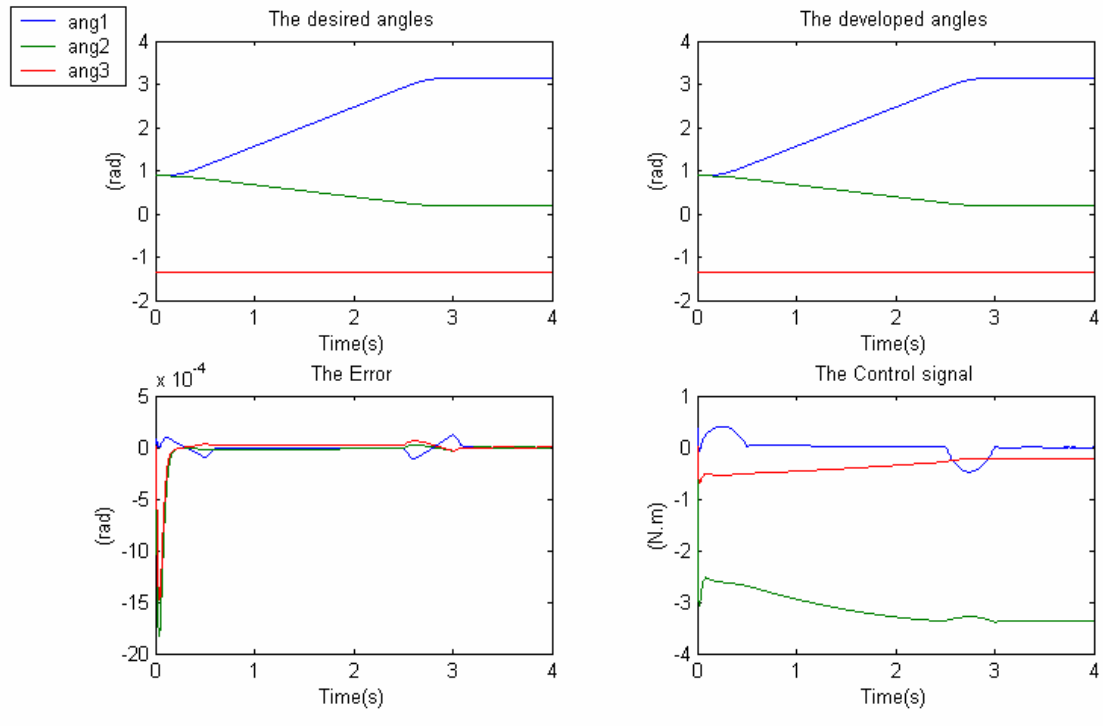


Figure 4.14

Imposed long trajectory

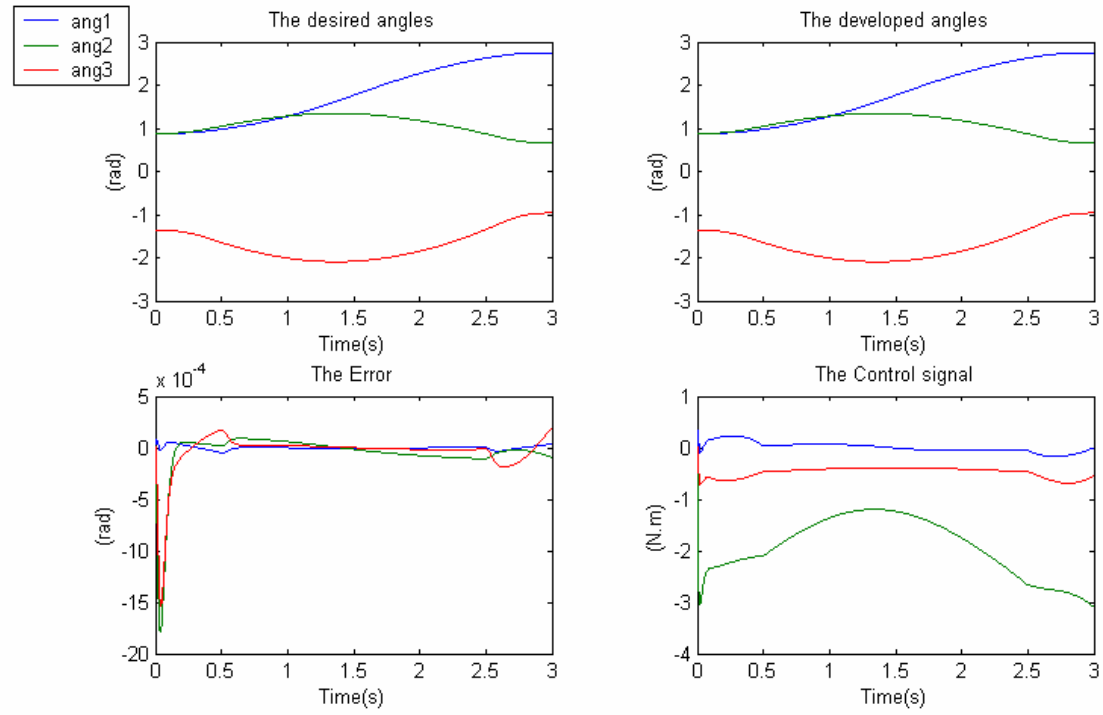


Figure 4.15

Now, let's see the effect of short time trajectories "imposed and free" without saturation.

Free long trajectory "short time"

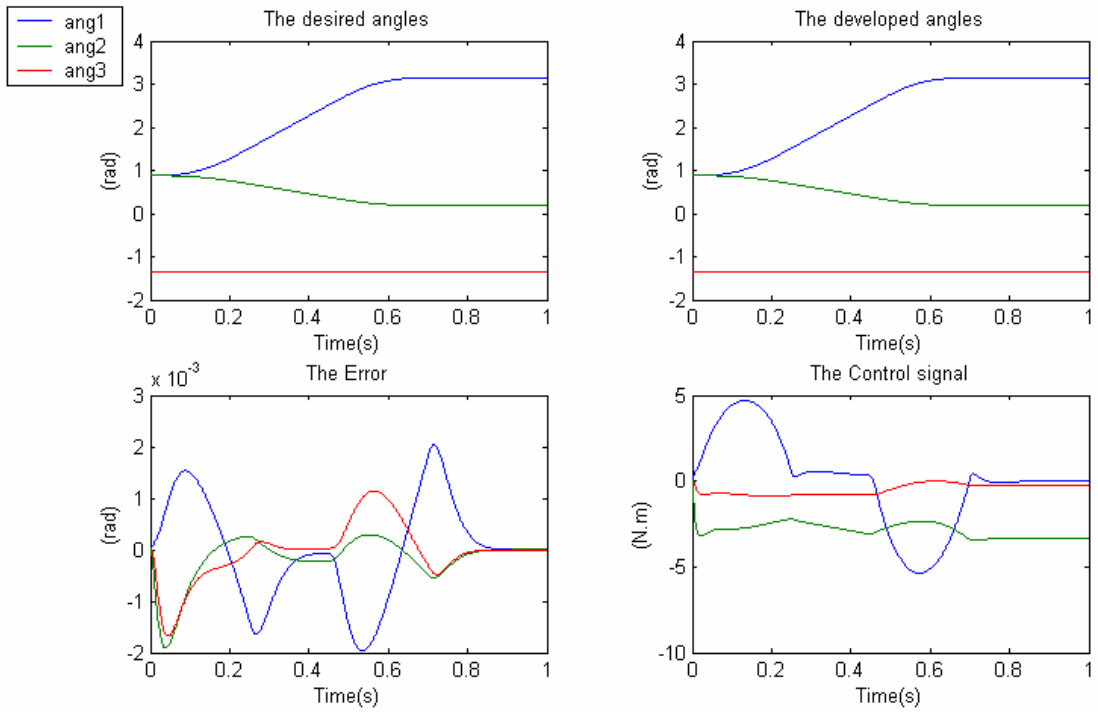


Figure 4.16

Imposed long trajectory "short time"

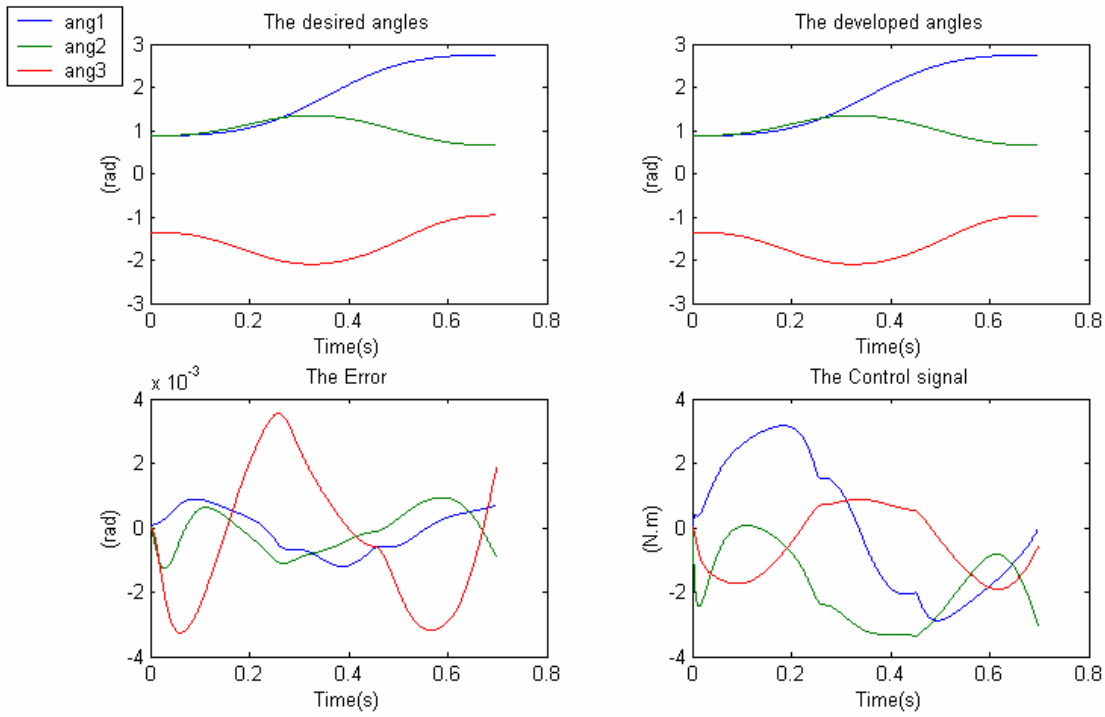


Figure 4.17

Now, let's introduce the hardware saturation terms and see the behaviour of the controller.

Free long trajectory "short time"

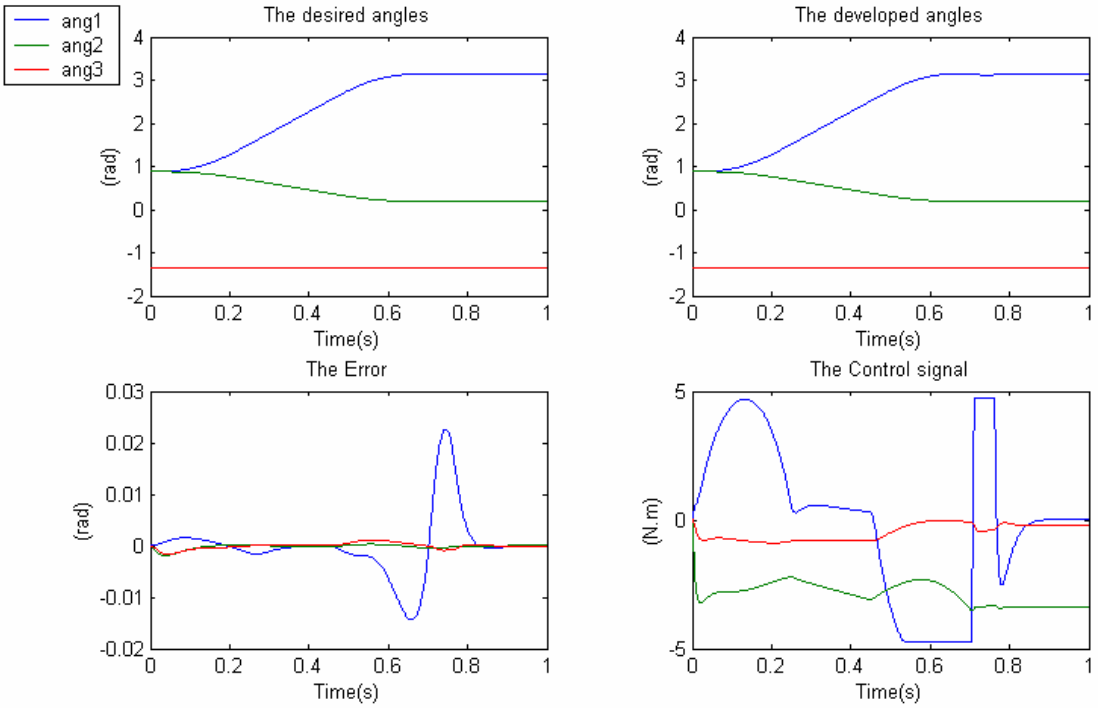


Figure 4.18

Imposed long trajectory "short time"

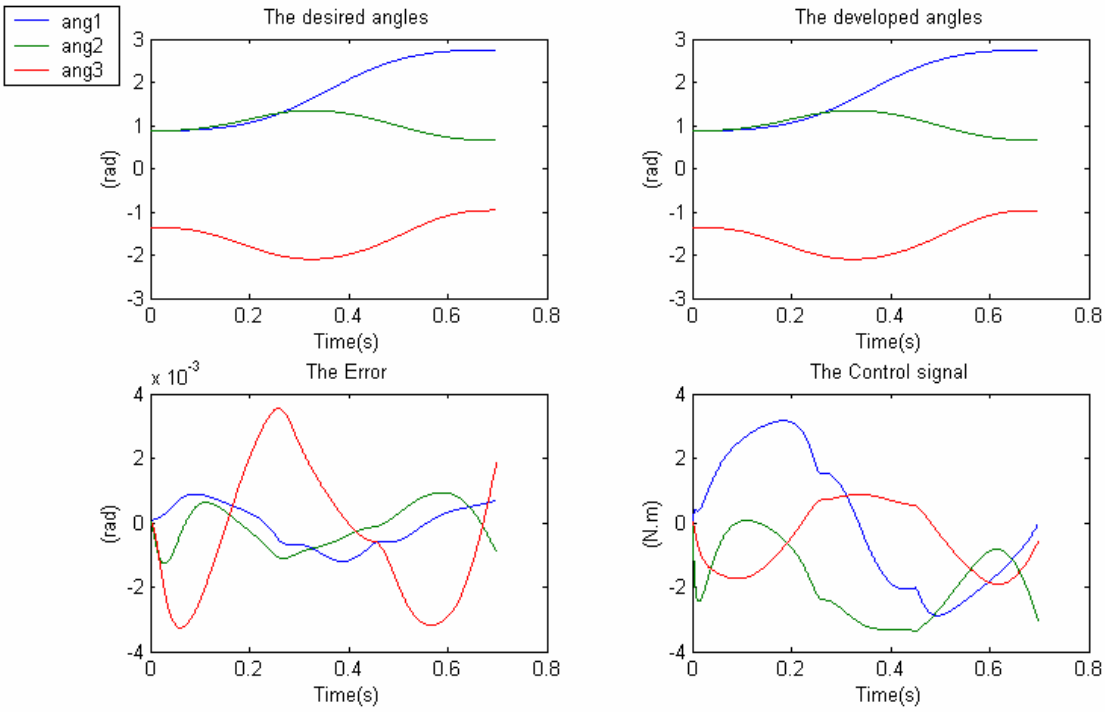


Figure 4.19

IV.5.2.2 Discussion

Some of the motor parameters may not be exactly known, in addition to that, the manipulator parameters are configuration dependent and depend also on the measurements precision; all these parameters introduce a steady state error. A way to cope with this error is to add an integral action term.

Figures (4.14-4.15) show a better controller performance for long trajectories with sufficient time length than the **PD** controller especially concerning the steady state error, and this is true for both free and imposed trajectories. The deteriorations of the **PID** controller performance for high speed trajectories with or without hardware saturation terms are smaller than those of the **PD** controller **Figures (4.16-4.19)**.

As a conclusion, a **PD** controller performance with large position and speed feedback gains is largely sufficient for applications requiring the achievement of long trajectories in a sufficiently long time and with a relatively large steady state error, but if we need better results for these applications and eliminate the steady state error that may be a source of problems for some specific applications "ex: medical applications" a **PID** controller is a solution.

IV.5.3 The Predictive Term

The derivative action of the **PD/PID** controllers may poses problems due to the sharp transitions on the controller input. These high frequency components may be due either to sharp corners on the reference input or to measurement noise from sensors such as tachometers and encoders resulting in very large derivative term yielding a very large control input from the **PD/PID** controller.

In order solve this problem, we will omit the predictive term $K_d \dot{q}_d$ from the controllers equations to prevent any sharp corners on the reference input, and we will pass the output signal for the derivative term through a low-pass filter of the form:

$$LPF = \frac{a}{s+a} \qquad (IV.8)$$

where a is chosen according the hardware we have.

In this simulation, since there is no noise due to measurements and the desired input trajectories don't present any sharp transition that may deteriorate the controller's performance, the predictive term has not been omitted and the output signal of the system is not filtered but this will be done and tested in the real time application, where this problem may occur.

IV.6 The Computed Torque Controller

For the compensator parameters K_p and K_d in equations *(I-82a-b)*, we will take $\zeta = 1$ and ω_c=50 (rad/s) and this choice will reduce the effect of modelling errors and disturbances giving at the same time realizable controller parameters with the fastest time response without overshoots and keeping the system stable.

The computed torque controller will be applied to the manipulator in the two schemes, in terms of the joint variables and the end-effector workspace variables "linearization with respect to the developed joint variables".

The previous assumptions yield the controller parameters K_p=2500, K_d=100 and this for the two schemes - here the state vectors of the motors are not included.

Short time trajectories: T-length=0.7s and Δ=0.125s,
Long time trajectories: T-length=3s and Δ=0.25s.

IV.6.1 Simulations with a Computed Torque Controller

Scheme 1: computed torque controller in terms of joint variables, **Figure 4.20**.

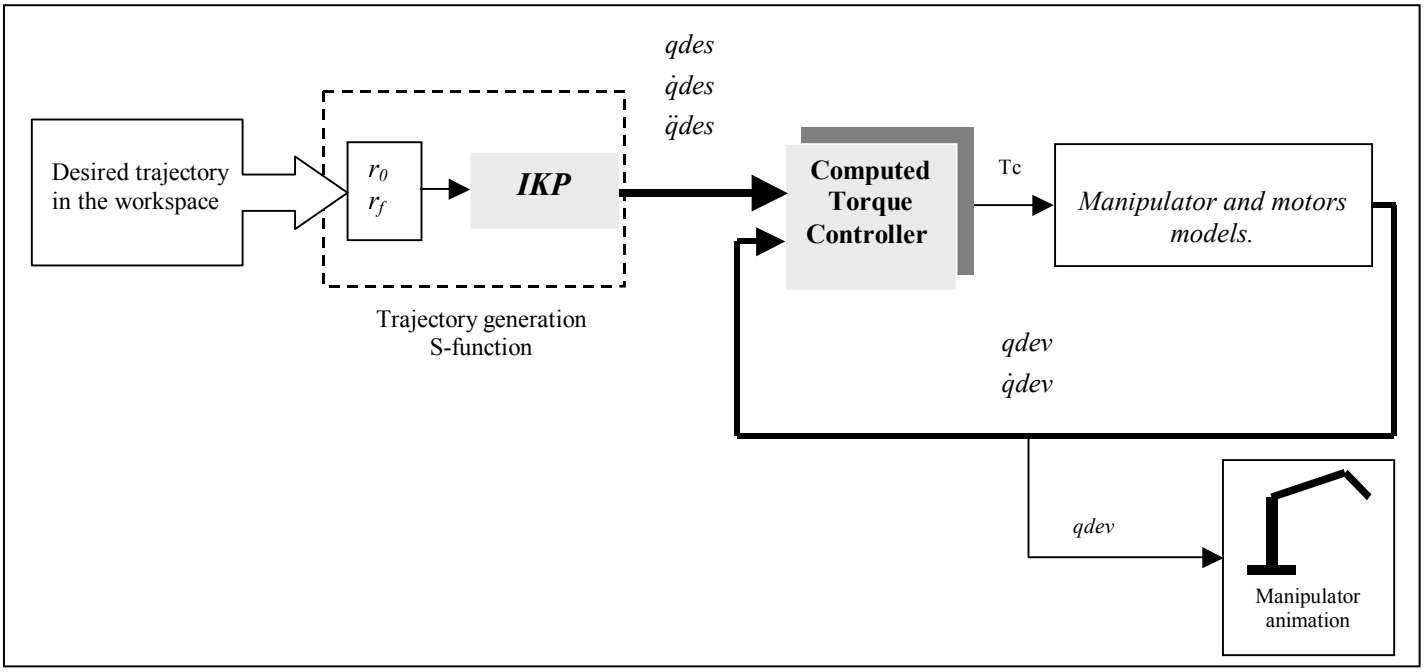


Figure 4.20: The computed torque controller - scheme 1.

Long free trajectory

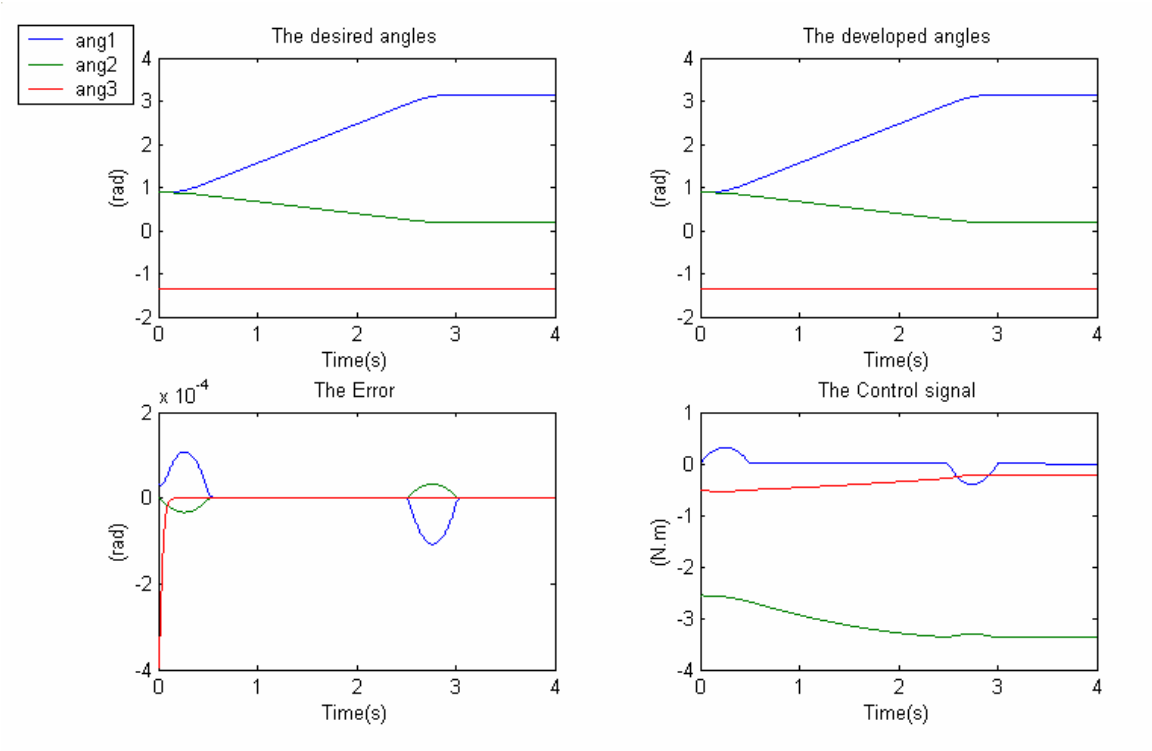


Figure 4.21

Imposed long trajectory

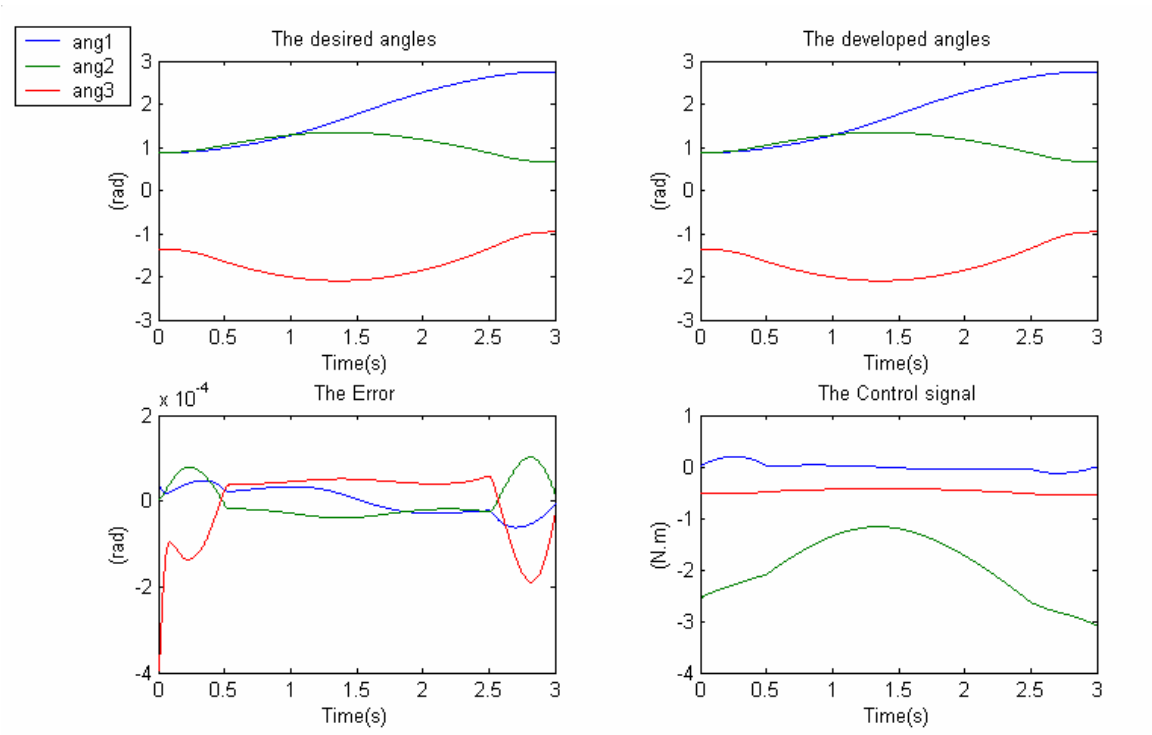


Figure 4.22

Now, we will see the effect of short time trajectories "imposed and free" without saturation.

Free long trajectory "short time"

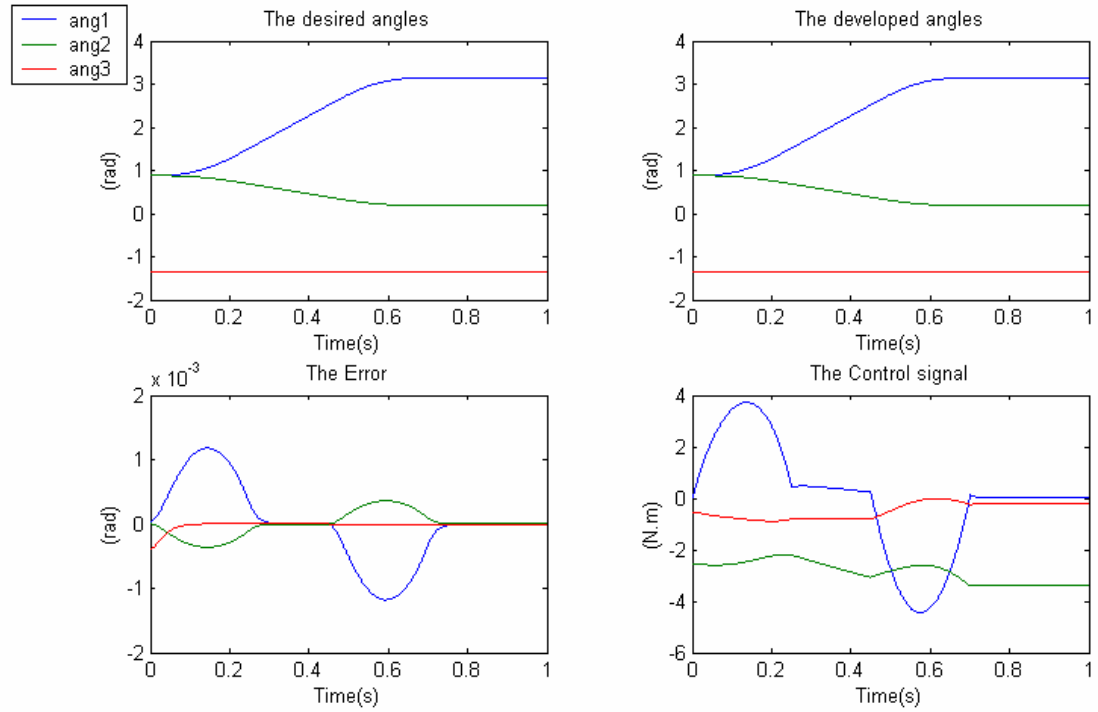


Figure 4.23

Imposed long trajectory "short time"

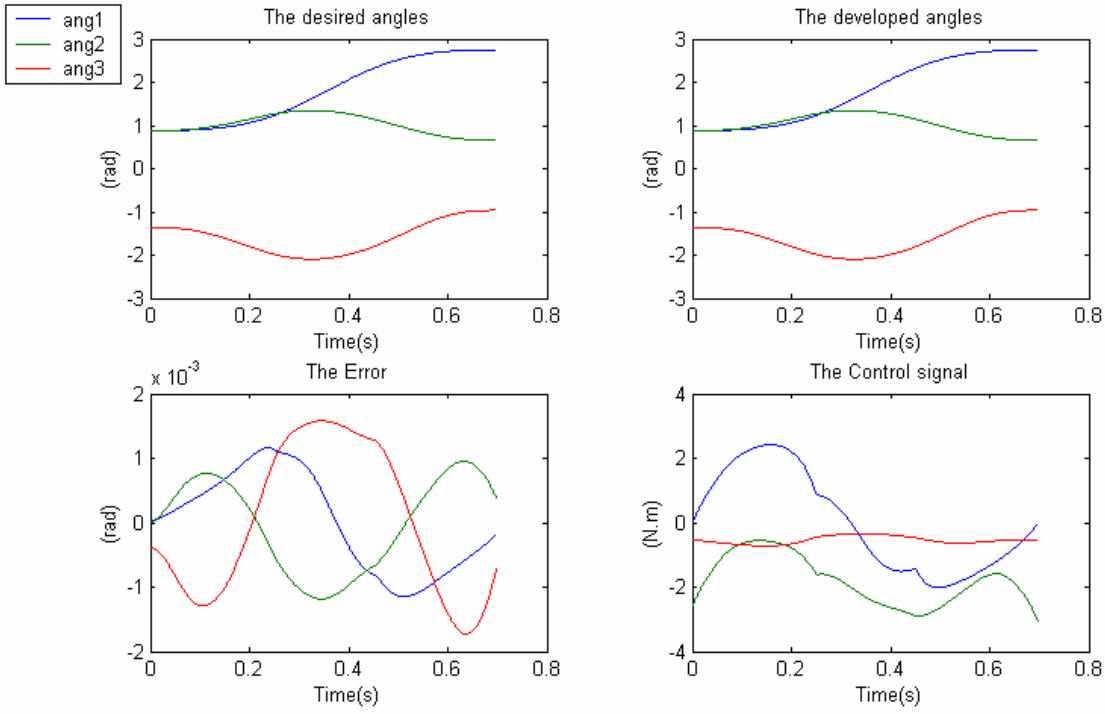


Figure 4.24

Scheme 2: computed torque controller in terms of workspace variables, **Figure 4.25**.

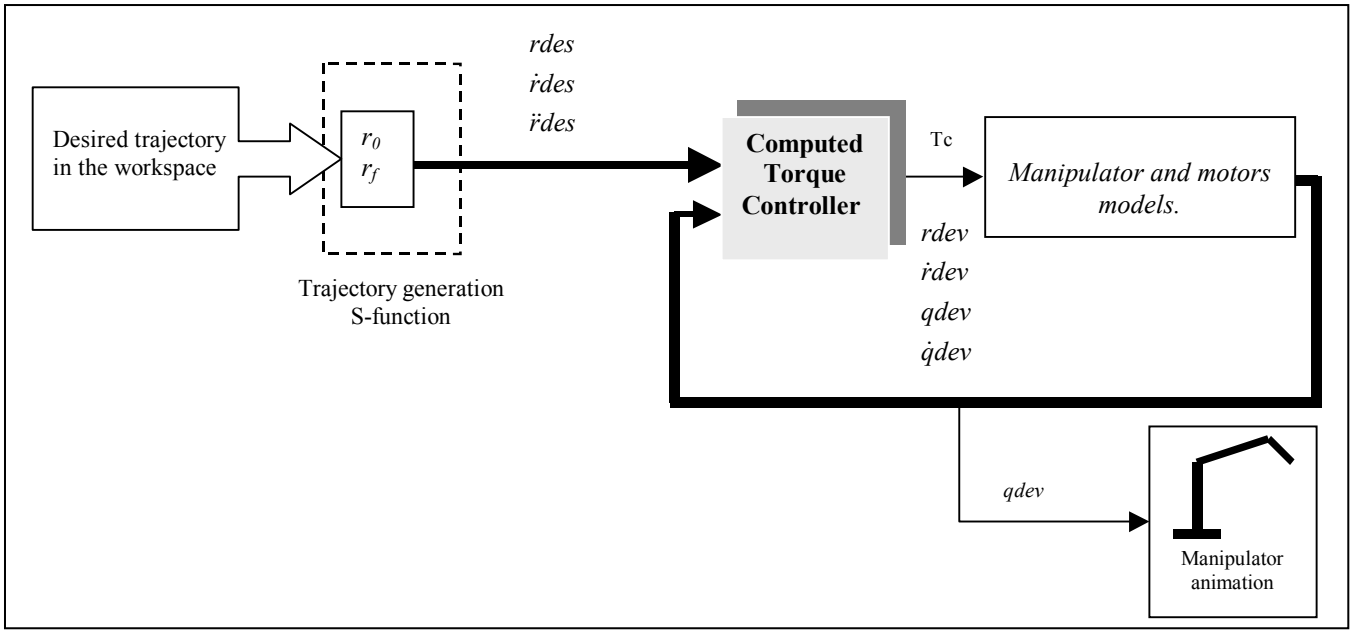


Figure 4.25: The computed torque controller - scheme 2.

Long free trajectory

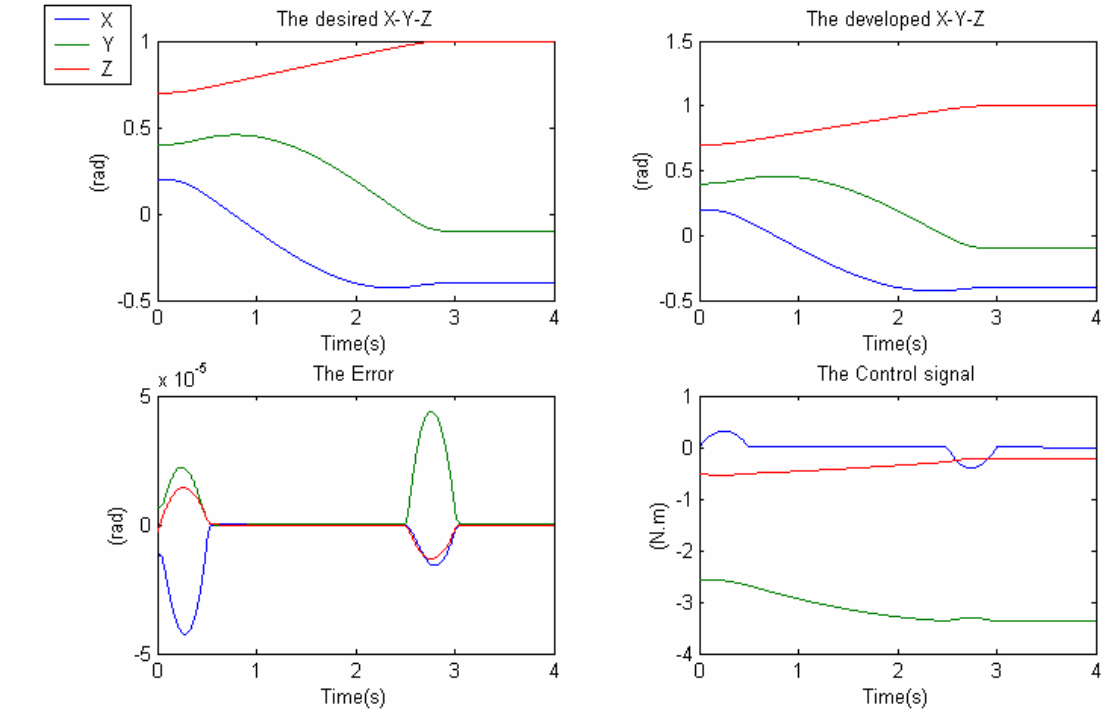


Figure 4.26

Imposed long trajectory

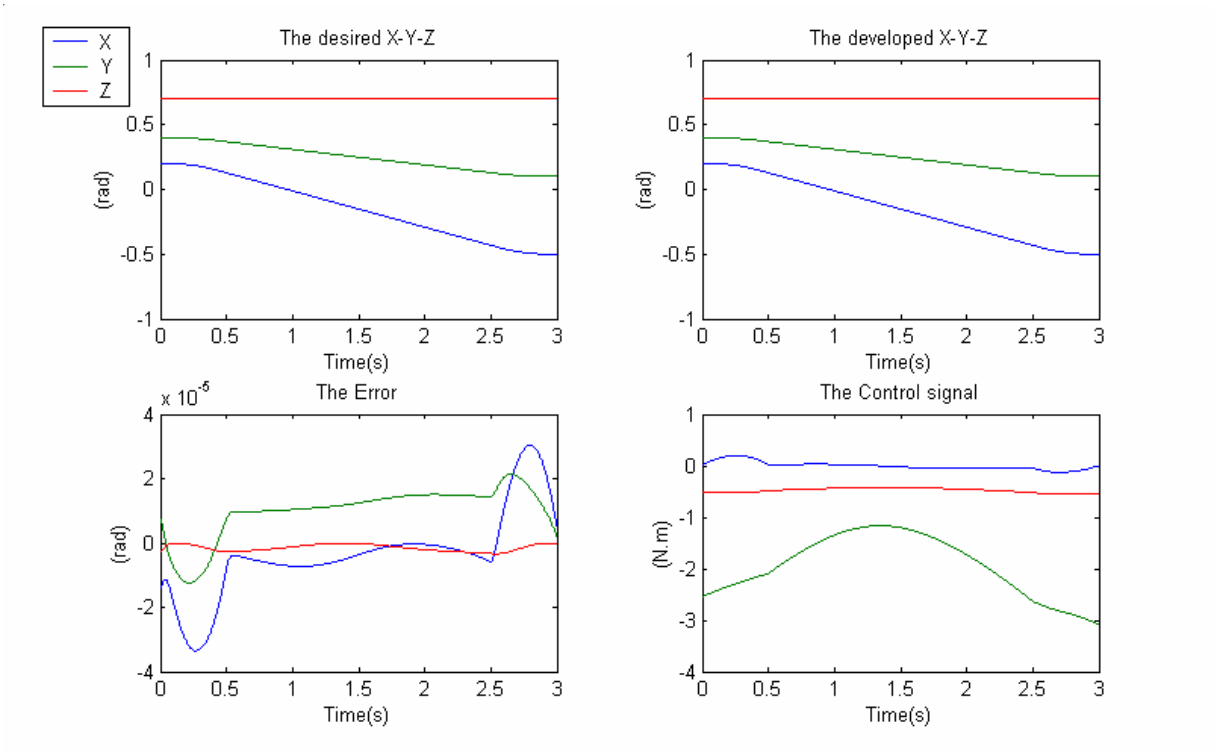


Figure 4.27

Free long trajectory "short time"

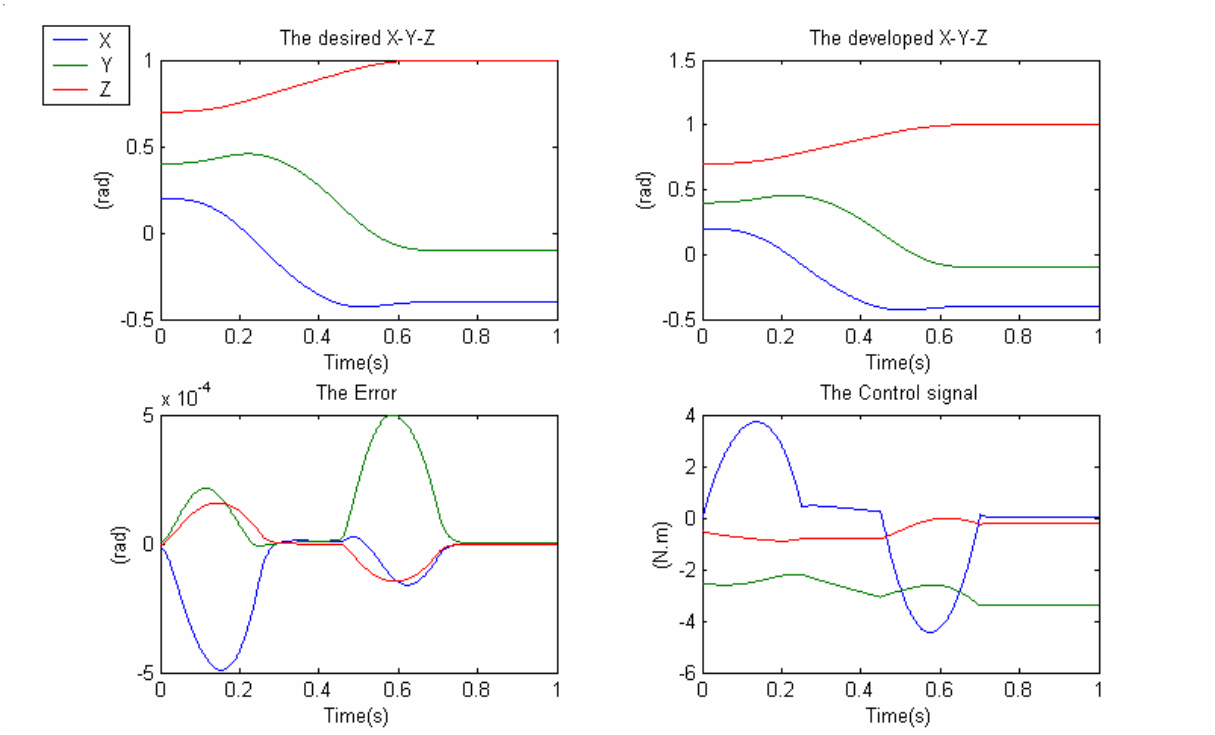


Figure 4.28

Imposed long trajectory "short time"

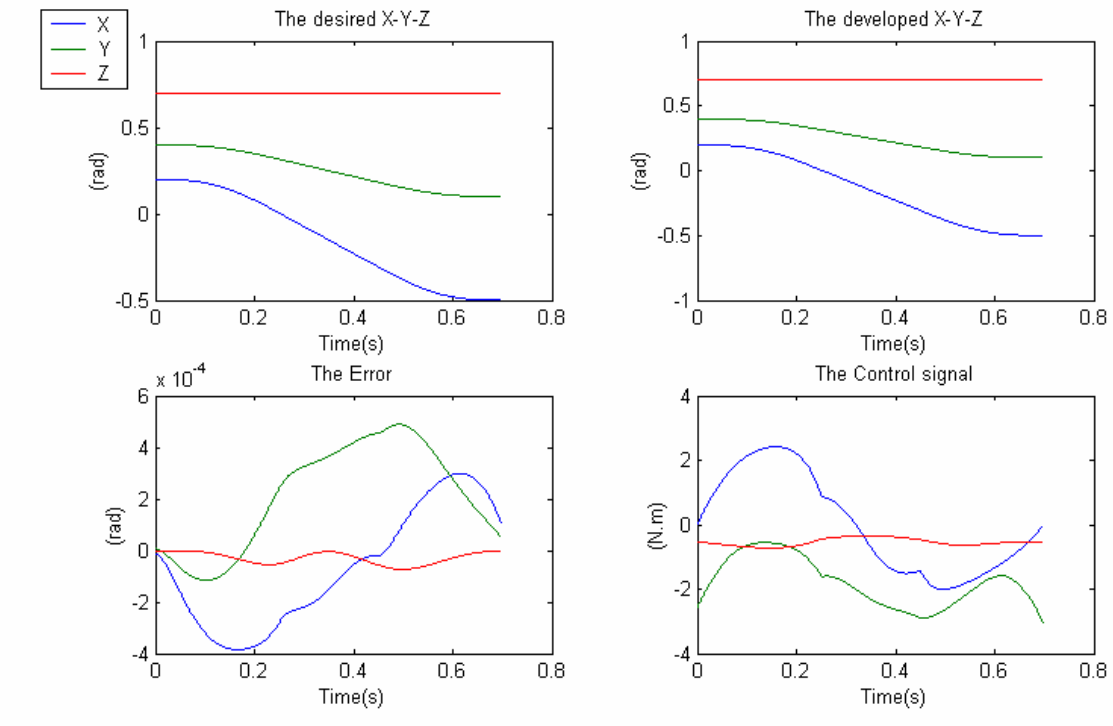


Figure 4.29

IV.6.2 Discussion

The severe demands for fast and accurate positioning led to the apparition of more complex control algorithms which take into account the dynamics interaction and the online calculation of the dynamics model of the manipulator ensuring a uniform response what ever is the configuration of the manipulator.

It can clearly be seen in **Figures (4.21-4.22)** that the computed torque controller gives better results than the previous control schemes we have tested in both free and imposed trajectories but the flagrant amelioration is in the short time trajectories in **Figures (4.23-4.24)** where we can see the performance of the controller with high speed trajectories, in addition to that, the control signals will not go beyond the imposed hardware saturation terms. In **Figures (4.26-4.29)** we have applied the controller in terms of the workspace variables performing the linearization in terms of the developed angle trajectories and the results are very acceptable.

IV.7 The PWM Signal as Control Signal

Here, we will illustrate the effect of using the voltage control signal as a PWM signal. The voltage control signal has to be compared with a triangular waveform signal; this triangular waveform with a **Relay** simulink block will be used as a block generation of PWM signals according to the input voltage control signal.

Now, we will see the behavior of the **PID** controller when using a PWM voltage control signal. Including the hardware saturation terms and setting the sampling frequency f_s=10KHz, **Figure 4.30**.

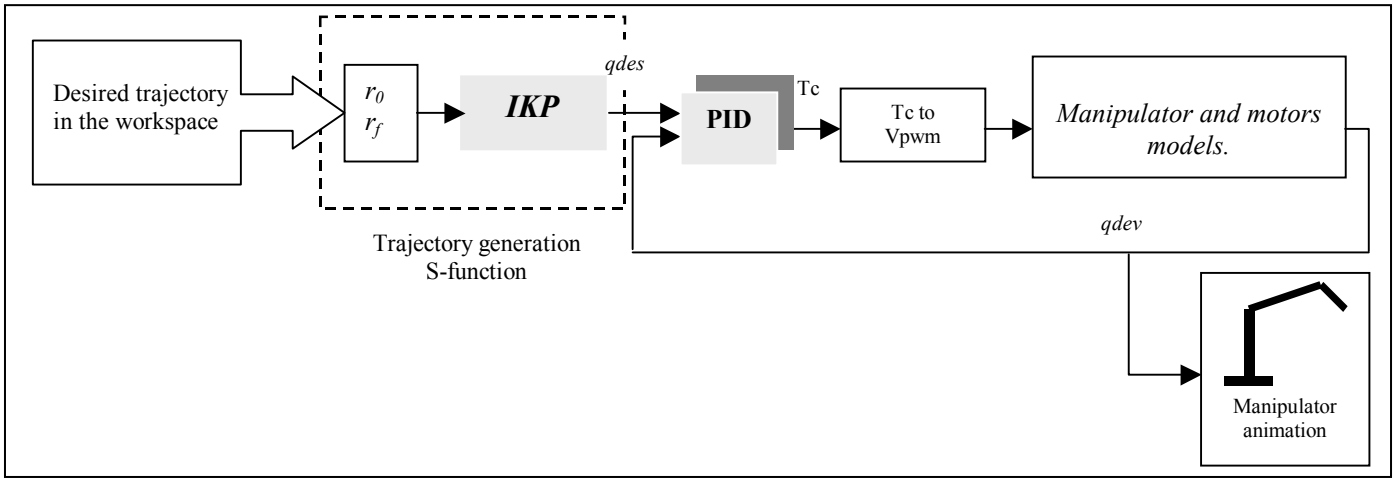


Figure 4.30: The PWM effect.

Long free trajectory

1) Switching frequency = 10000Hz and the rating voltage=18v.

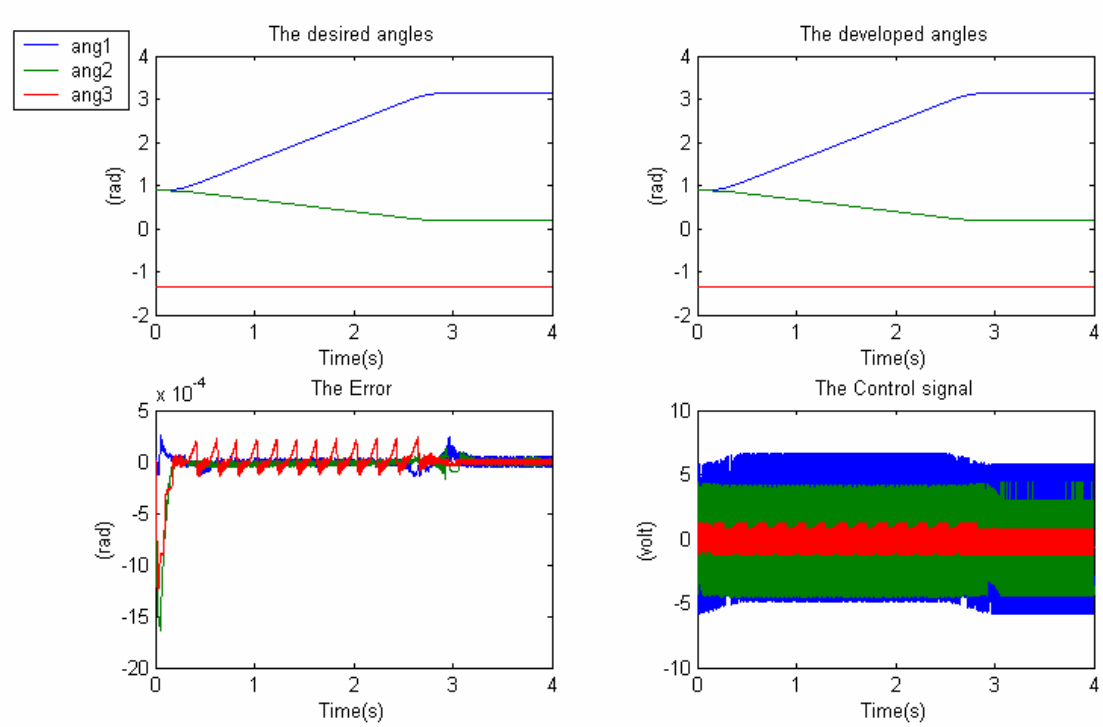


Figure 4.31

2) Switching frequency = 5000Hz and the rating voltage=18v.

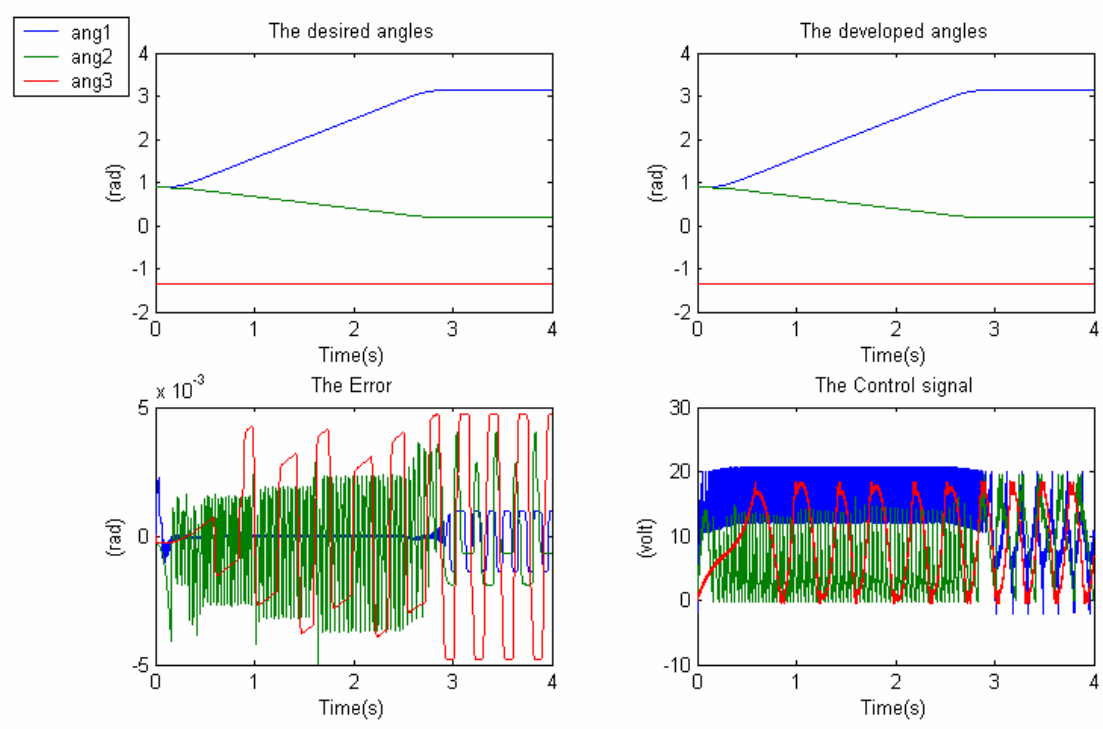


Figure 4.32

3) Switching frequency = 1000Hz and the rating voltage=18v.

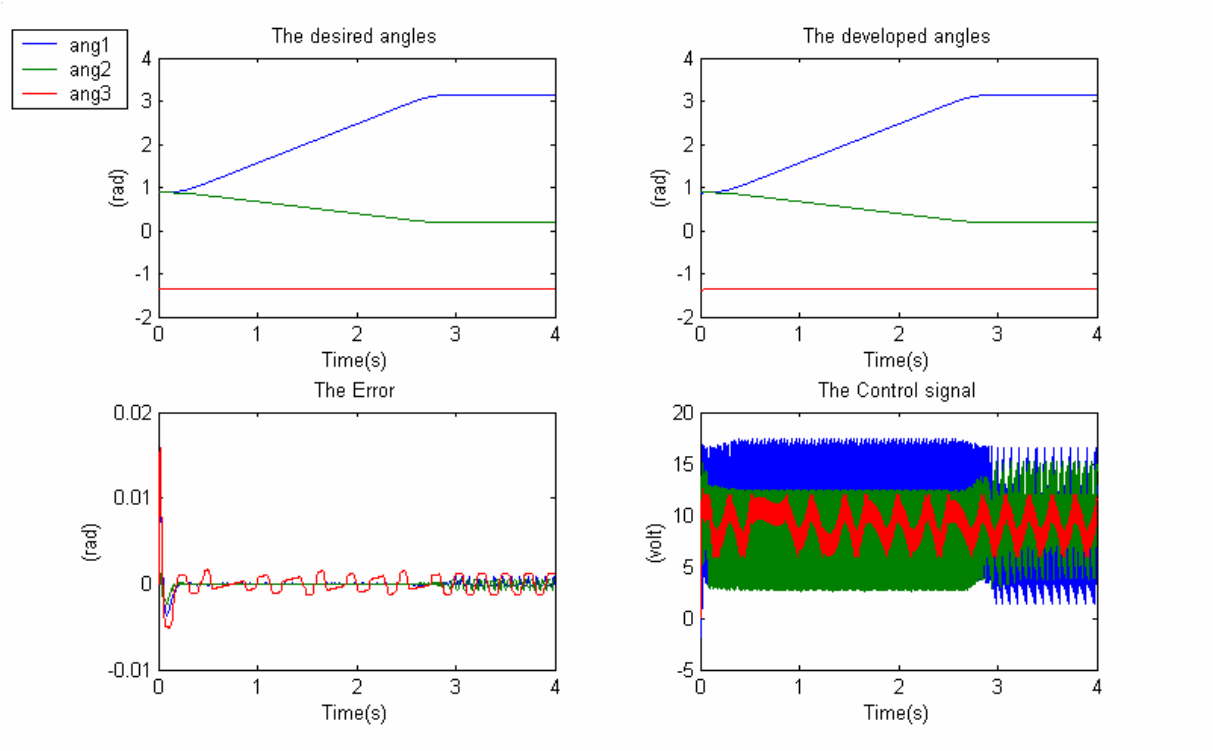


Figure 4.33

4) Switching frequency = 5000Hz and the rating voltage=18v with a short time trajectory:

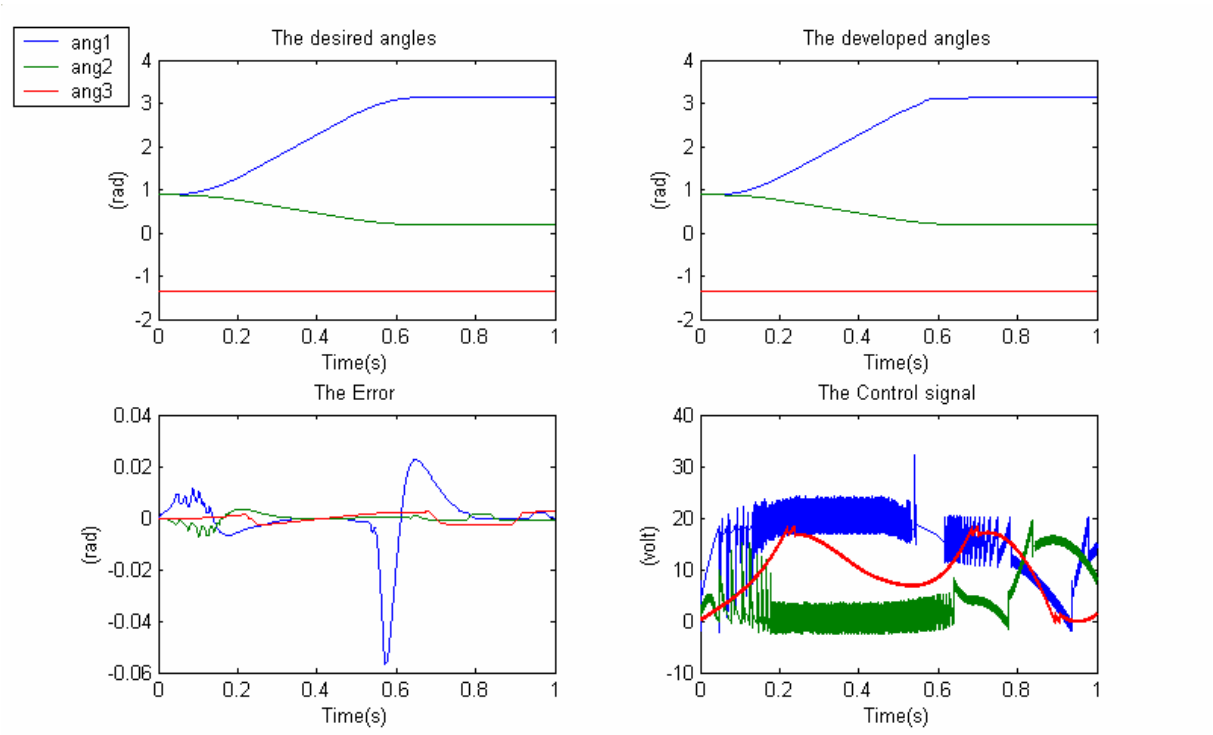


Figure 4.34

5) Switching frequency = 100000Hz and the rating voltage=18v with sampling frequency f_s=100KHz, short time trajectory:

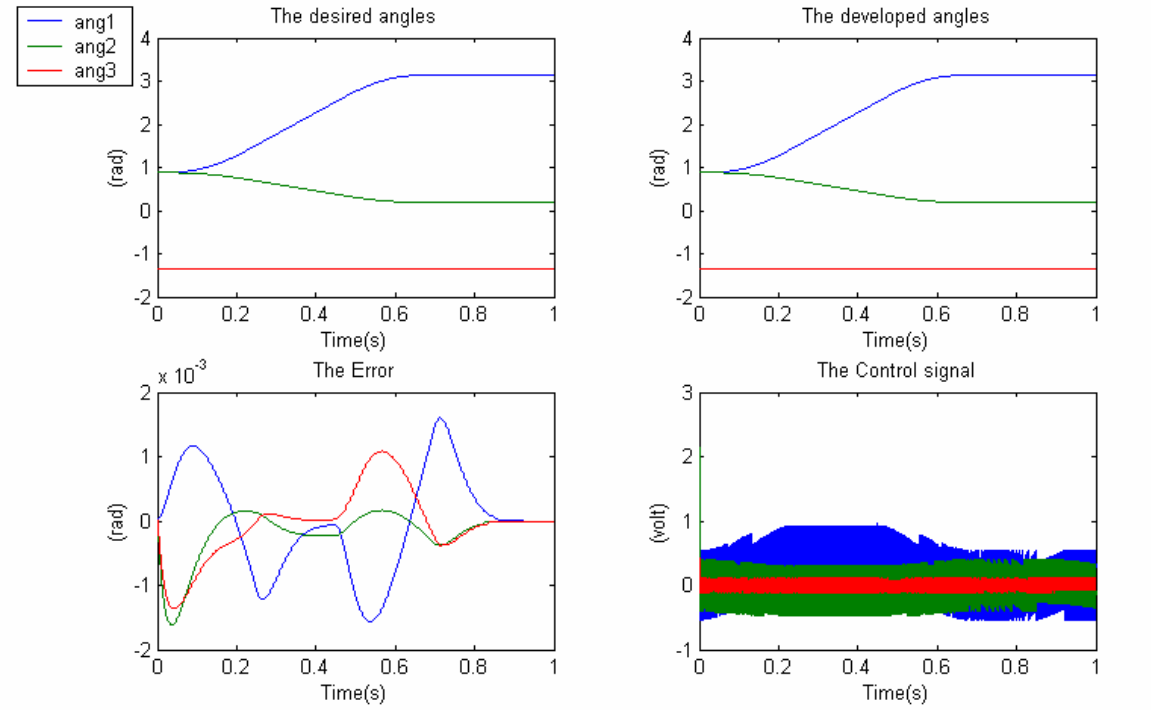


Figure 4.35

We can conclude that, **Figures (4.31-4.33)**, using a PWM signal as a voltage control signal will not affect significantly the performance of the controller as soon as we apply high switching frequencies with sufficient rating voltage which depends on the hardware characteristics and using a sufficiently small sampling periods, **Figures (4.34-4.35)**.

.

IV.8 Digital Control

Two approaches for the design of digital controllers exist:

- Design in the s-plane as if the controller were analog, and then convert the design to the z-plane.
- Design directly in the z-plane.

If we sample fast enough the first approach (digitization or emulation) will work well, advantageous if an existing analog design is to be replicated in digital controller.

If the sampling interval is constrained by the hardware or by the length of computations to be done between samples, the second approach (direct design in the z-plane) will be better, advantageous if the system to be designed is from scratch since analog emulation might not give the best digital design.

IV.8.1 The Difference Equations

Whereas continuous systems are described by differential equations, discrete systems are described by difference equations. We can see that the difference equations show the relationship between the input signal *e(k)* and the output signal *u(k)*. Suppose we are interested in the *kth* output signal *u(k)*, then to get this output signal, we have the computer that computes some function that considers past input signal *e(0)* to *e(k)* and output signals *u(0)* to *u(k-1)*, which can be expressed as a function of the form:

$$u(k) = f(e(0),...,e(k);\ u(0),...,\ u(k-1)) \qquad (\text{IV}.9)$$

We assume that the function *f* is linear and depends only on a finite number of signals *e's* and *u's*, then the basic structure of the difference equation can be written as:

$$u(k) = -a_{k-1}u(k-1) - ... - a_0\,u(0) + b_k\,e(k-1) + ... + b_0\,e(0) \qquad (\text{IV}.10)$$

IV.8.2 The Digital PID Controller

In the PID controller, the integral term is replaced by a sum operator and the differentiation term by a difference term. The difference equation of a PID controller is:

$$u(k) = k_p e(k) + k_i \sum_{i=0}^{k} e(i) + k_d\ (e(k) - e(k-1)) \qquad (\text{IV}.11)$$

All the other controllers are derived forms of the PID controller.

IV.8.3 The Transfer Function using Z-transform

To derive the transfer function in discrete form, a mathematical tool very similar to the Laplace transform called **the z-transform** will be used.
The z-transform is defined by:

$$Z\{f(k)\} = F(z) = \sum_{k=0}^{m} f(k)\ z^{-k} \qquad (\text{IV}.12)$$

where *f(k)* is the amplitude of a sample, and the values k = 0, 1, 2, 3... refer to the discrete sample times.
analogously, this can lead to the relation

$$Z\{f(k-m)\} = z^{-m}\ F(z) \qquad (IV.13)$$

By using these relations, we can easily find the discrete transfer function of a given difference equation.

IV.8.4 The Bilinear Transformation

There are several ways for mapping from the s-plane to z-plane, such as, the forward and backward difference approximations. The bilinear transformation is another method which maps the entire left half of the s-plane into the unit circle in the z-plane. The bilinear transformation maps from the s-plane to the z-plane using the following relation:

$$s = \frac{2}{T_s}\frac{z-1}{z+1} \qquad (IV.14)$$

where T_s is the sampling time of the discrete-time system.

In the continuous-time systems, a transfer function for a PID controller is described as follows:

$$k_p + \frac{k_i}{s} + k_d\ s \qquad (IV.15)$$

where k_p, k_i and k_d are the proportional, integral and derivative gains respectively. To map to the z-plane, we substitute the bilinear relation into the above PID transfer function, so, the discrete-time PID controller becomes:

$$k_p + k_i\frac{T_s}{2}\frac{z+1}{z-1} + k_d\frac{2}{T_s}\frac{z-1}{z+1} \qquad (IV.16)$$

$$\frac{(k_p + \frac{k_i\ T_s}{2} + \frac{2\ k_d}{T_s})\ z^2 + (k_i\ T_s - \frac{4k_d}{T_s})\ z + (-k_p + \frac{k_i\ T_s}{2} + \frac{2\ k_d}{T_s})}{z^2 - 1} \qquad (IV.17)$$

IV.8.5 Digital Control Features

Digital control applications are continuously growing and varying especially with the improvements in the processors technology, in the following, we will illustrate the advantages and disadvantages of digital control:

IV.8.5.1 Advantages of Digital Control

- The initial cost of a digital computer (and interfaces) is large but the incremental cost of adding extra loops is relatively small.
- The digital controller is cheaper for 'large' systems.
- Operator interaction, monitoring methods and display features are improved.
- Integration of individual controllers into the overall computer system is much easier.
- Analog systems are changed by re-wiring; changing a digital system by re-programming is much easier.
- Interaction among several control loops is easier to implement.
- Some physical processes are best represented as discrete time systems:

  - Stepping motors.
  - Radars.
  - Economic systems.
  - Power electronic control.
- The class of implementable control laws is greatly increased:
  - Non-linear calculations, e.g. linearising functions, coordinate transformations.
  - On-line parameter changes.
  - Optimisation.
  - Stable implementation of high order dynamic control laws.
  - System identification.
  - Adaptive control laws.
  - 'Intelligent' control, e.g. neural networks, genetic algorithms, fuzzy logic.
- Digital signals are relatively immune from noise and can have higher accuracy.
- Digital computation is becoming cheaper.
- It can also implement supervisory control and data acquisition systems.

IV.8.5.2 Disadvantages of Digital Control

- Converting an analog design to digital implementation usually degrades stability (due to delay in ZOH of the DAC converter).
- On balance, digital is preferred except in very simple controllers with low performance requirements.
- Analysis of hybrid systems is challenging, so the design is sometimes harder (not always).

IV.8.6 Cascade Regulation

Cascade regulation minimises the effect of one or many perturbations that may act on the control signal or on other signals after it, **Figure 4.36**. This type of regulation is very interesting especially when dealing with slow processes where the perturbation is detected only when it appears at the input of the compensator.

We will add an internal control loop in order to detect more rapidly the perturbation and compensate its effects, which means also that the internal loop is faster then the outer principal loop; the internal compensator must contain an integrator to eliminate the steady state error.

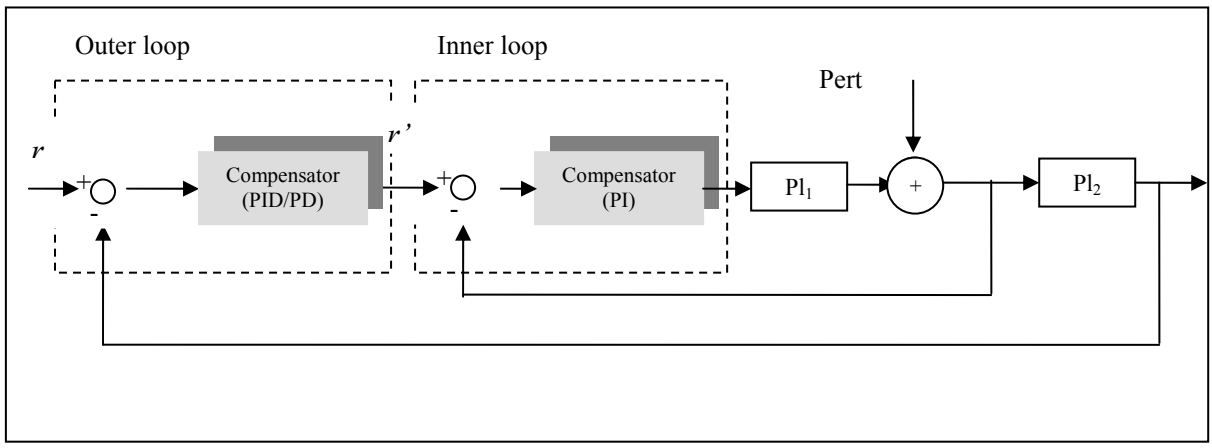


Figure 4.36: Cascade regulation.

Another feature of the cascade regulation is that actually we are ensuring that the control torque signal is the signal given to the plant, which is very useful when saturating the control signal since the signal given to the plant is automatically saturated.

Other industrial techniques of regulation exist and they will not be considered here.

IV.8.7 Cascade Regulation and Manipulators

Cascade regulation may be a good solution to the perturbations that may occur to manipulators while performing their assigned tasks allowing a fast response to the perturbations by compensating them without affecting the trajectory tracking or force tasks.

Grasping an object while tracking a desired trajectory can be considered as a perturbation, actually, when a manipulator grasps firmly an object, it may be considered part of the last link this is why, in the following simulations, we will not consider the dynamics parameters of the load and even though the load torque will not affect with the same manner the three joints, here we will consider it uniform.

Now, we will test the cascade regulation with our manipulator by applying torques perturbations with different durations and at different instants and observe their effects on our cascade regulation compared with a simple PID controller scheme, **Figure 4.37**.

We will use a PID compensator for the outer loop and PI compensator for the inner loop.

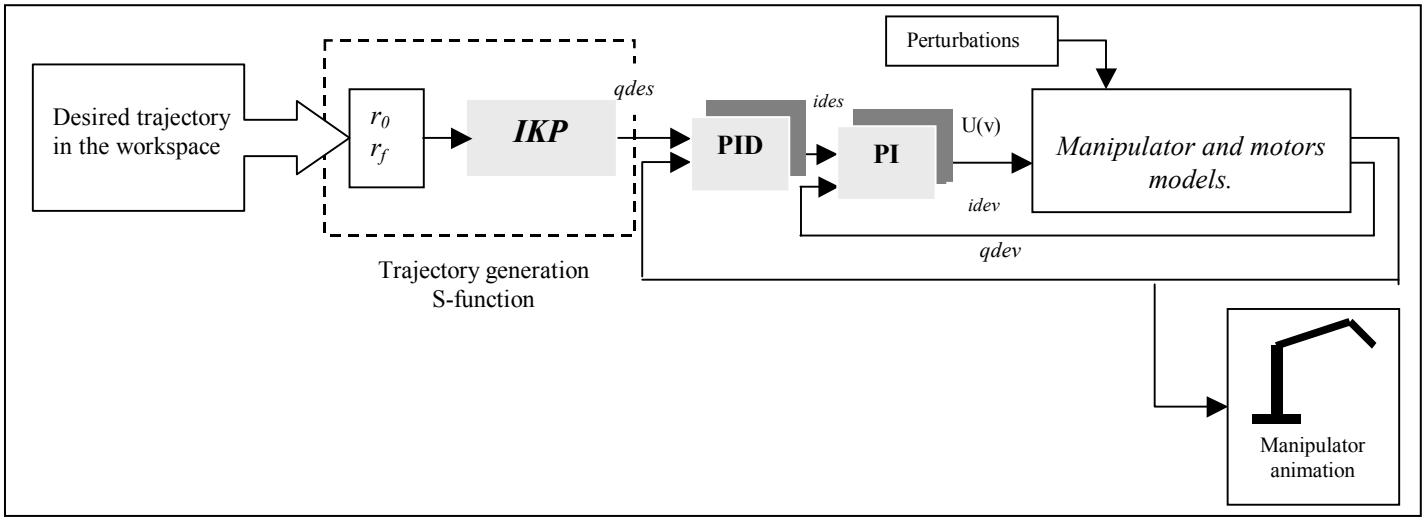


Figure 4.37: Cascade regulation with the manipulator.

IV.8.7.1 Compensators Parameters

The first step in designing a discrete control system is to convert the continuous transfer function to a discrete transfer function; to do that, we will use the Matlab command **c2dm,** the c2dm command requires the numerator, denominator, the sampling time of the transfer function and the type of conversion.

For the PID and the PI controllers we will use the c2dm command with the "Tustin" conversion type which uses the bilinear approximation.

After calculations and using the root locus method in the z plane to change the compensator design and make the system stable for at least some gains values, we will choose an appropriate gain value from the locus to satisfy our requirements.

The chosen gain k=70 for the three links with the previous PID parameters and for the PI parameters, a choice of K_p=1 and K_i=1000 will be appropriate with the electrical characteristics of the used dc motors, for the sampling times, we will apply Ts=0.001 for the outer loop and Ts=0.0001 for the inner loop giving 1 period for the outer loop for 10 periods for the inner one.

IV.8.7.2 Simulation with Cascade Regulation

The trajectory used is the long free trajectory; the control voltage is a PWM signal rated at ±18V with a switching frequency of 10000Hz and the current saturated at ±1.5A.

1) A torque perturbation of 7 (N.m) at instant t=2s.

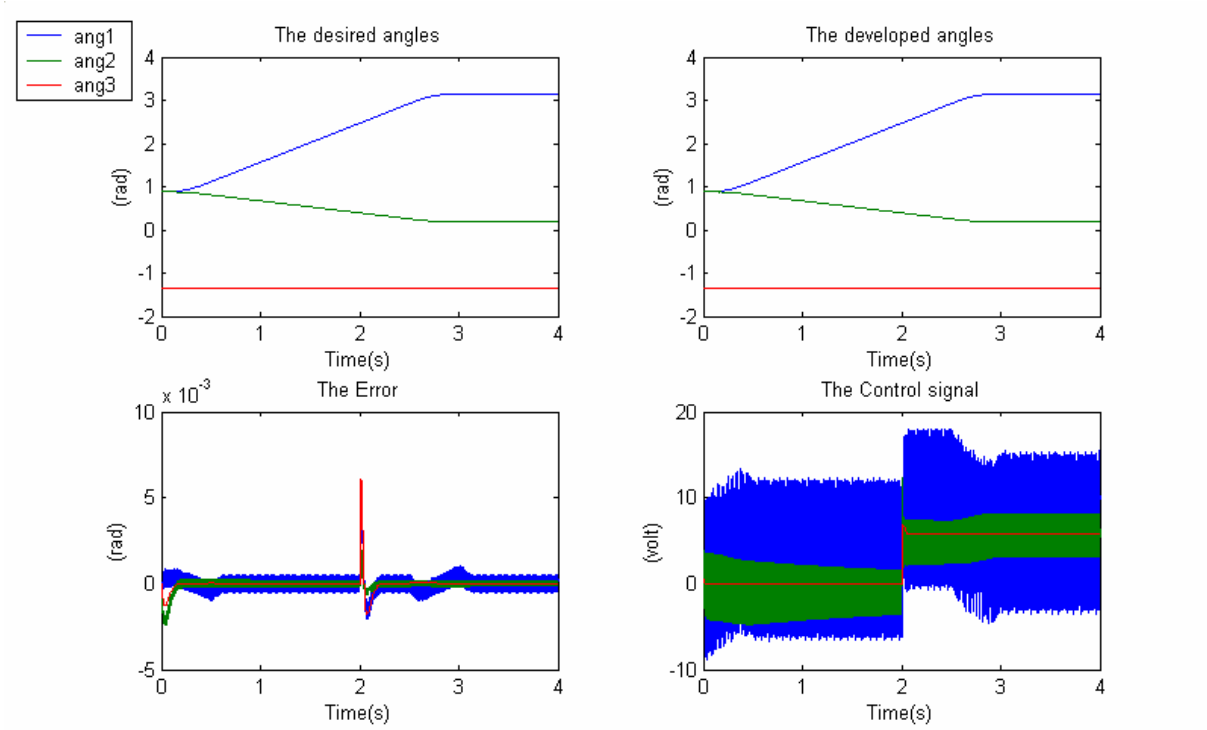


Figure 4.38

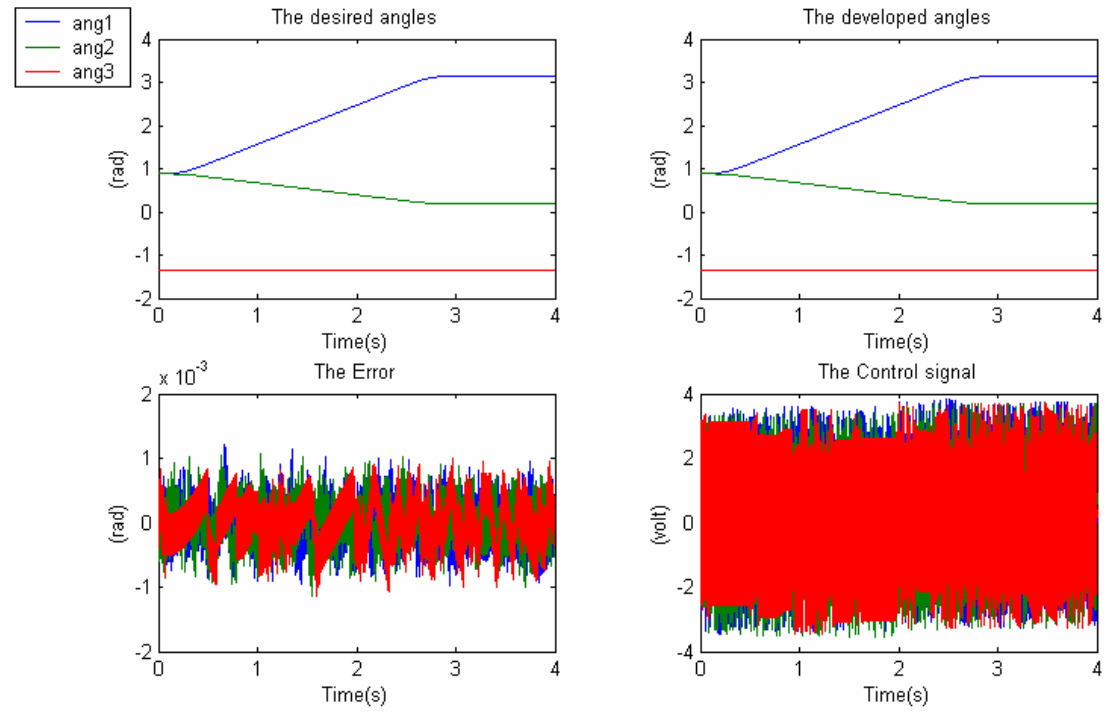


Figure 4.39

2) A torque perturbation of 7 (N.m) with a duration from t=1.5s to t=2.5s.

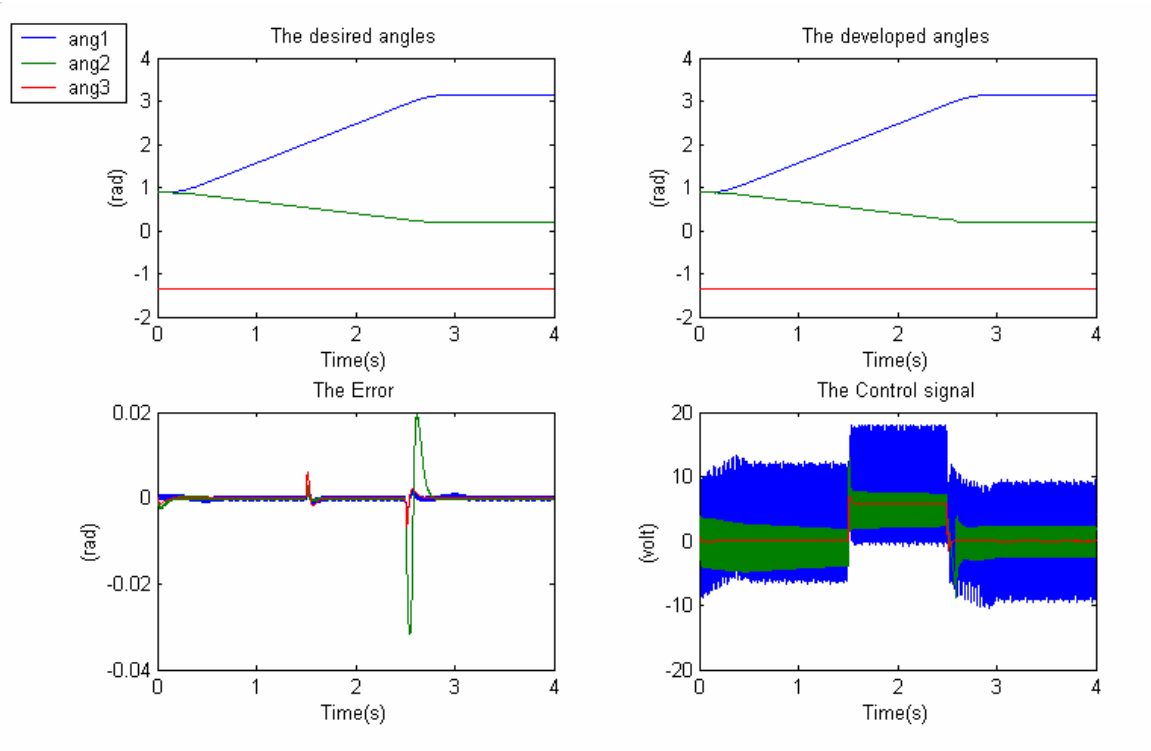


Figure 4.40

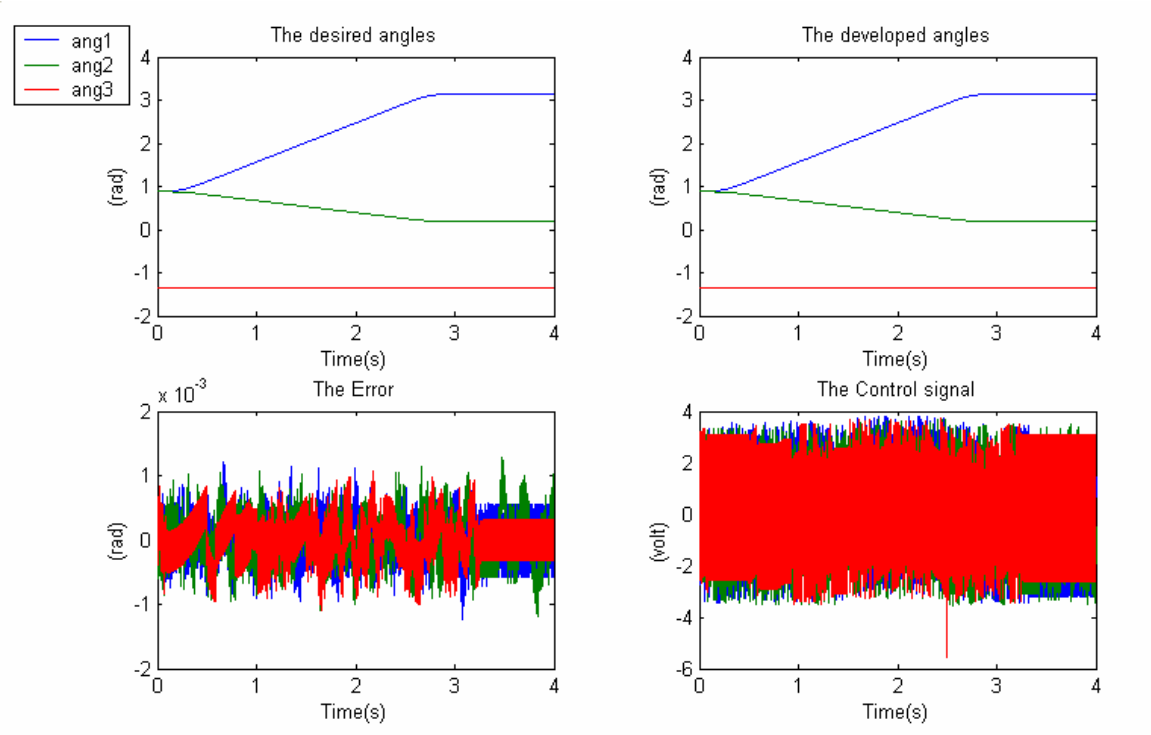


Figure 4.41

3) A torque perturbation of 7 (Nm) from t=0s to t=2s.

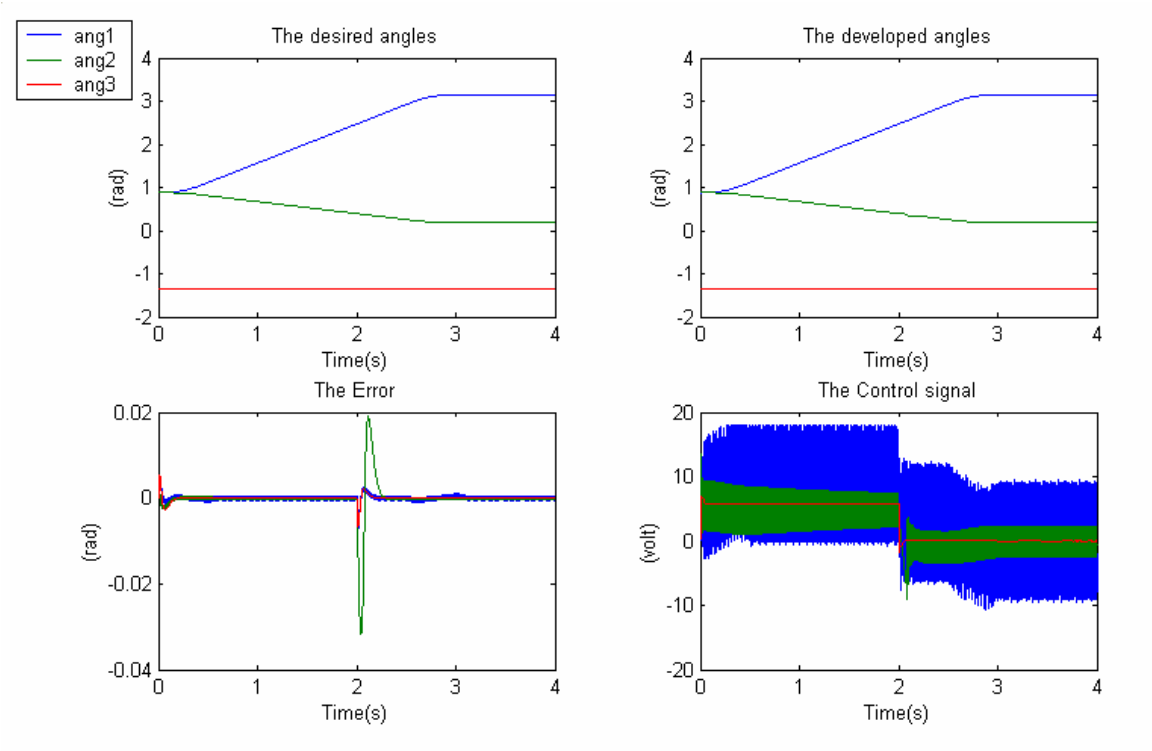


Figure 4.42

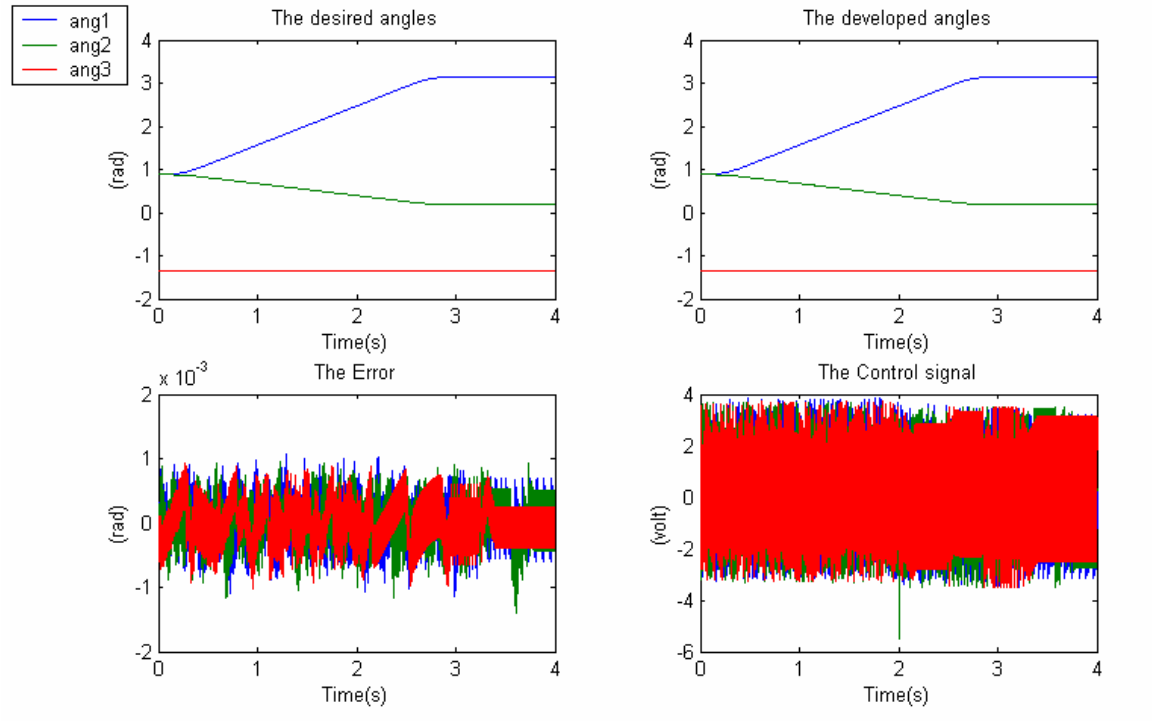


Figure 4.43

IV.8.7.3 Discussion

The **Figures (38-40-42)** show the effect of uniform perturbation torques on the performance of the simple PID controller, tested in subsection **IV.5.2**, in tracking a specified trajectory and it is clear that the effect on the tracking error is present even though it is relatively small, but also we can see that the effect of removing a perturbation while tracking a trajectory is larger than its effect when applying it especially for joint 2 which is the joint supporting the largest weight.

The **Figures (39-41-43)** show clearly the performance improvements due to the fast perturbation compensation in the inner current loop and it can clearly be seen that the tracking error is approximately the same whatever is the perturbation and the instant at which it is applied and requiring at the same time a smaller control voltage input than the simple PID controller.

IV.9 The Real Time Application

Real time means completing the processing within the allowable or available time between samples [4]. This available time, of course, depends on the application.

In the real time application, we will use the developed mechanical structure as our 3 degrees of freedom manipulator **Figure 4.44**, with the parameters shown in **Tables 4.1-4.2** for both the manipulator and the driving dc motors "links lengths, masses …" and the regulators parameters used with the manipulator model with the possibility of further tuning **(RCP)**.

Figure 4.44: The manipulator structure.

IV.9.1 The Simulink Model

The simulink model used for the real time application **Figure 4.45** must contain:

- The s-functions used to generate the trajectories to be used as reference input to the controllers: These s-functions must be written in **C** with the possibility to control the trajectory generation by specifying the final positions, the trajectory time length with a complete starting control and that at any time during the application.
- The controllers used for the application and the blocks necessary to convert the control signals to duty cycles.
- The I/O interfaces necessary to communicate with the external devices "manipulator and dc motors" "in yellow":
  - A PWM generation block to which we will inject the generated duty cycles.
  - The encoders' blocks in order to get the developed angles (positions and velocities) to be fed back to the controllers.
  - The bit I/O blocks used for the initialisation, as hardware limitations and for the motors sense control.
- Other simulink blocks such that "Enable and Trigger" subsystems are used in order to control the overall real time application process.

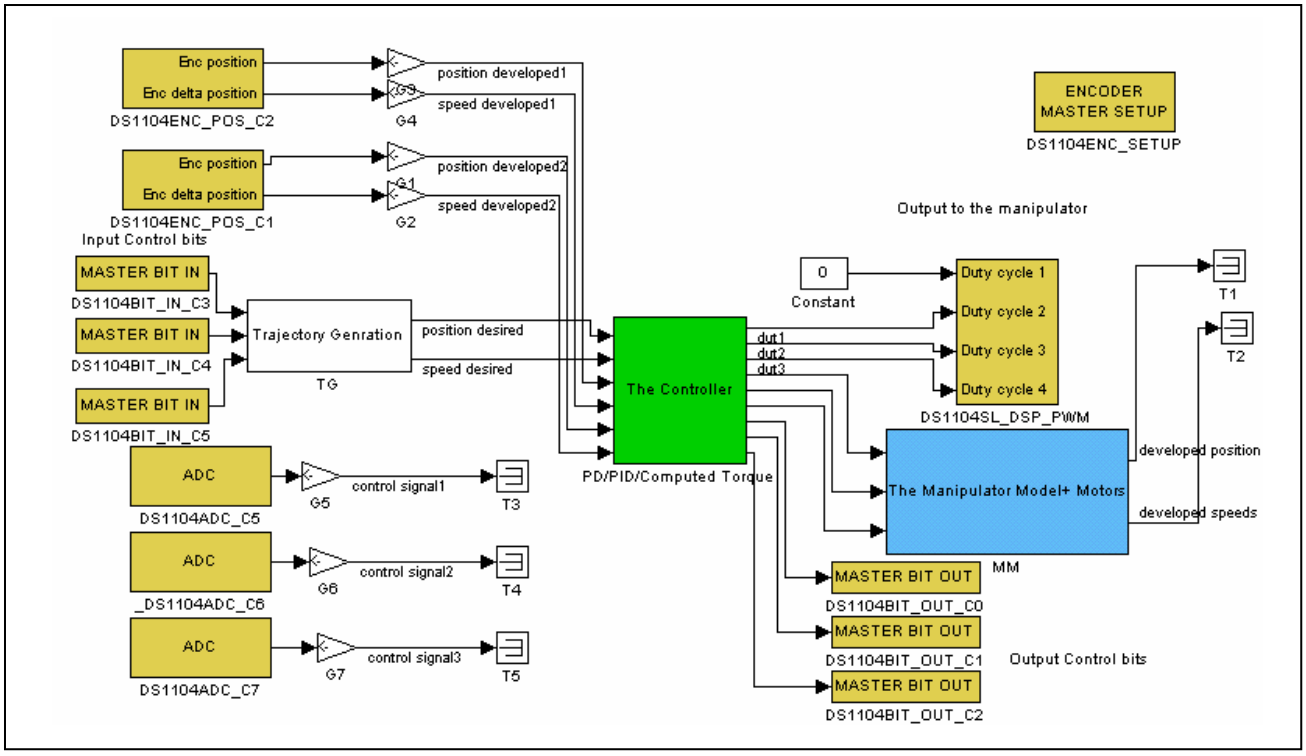


Figure 4.45: The general Simulink model used for the real time application.

IV.9.2 The ControlDesk GUI

The ControlDesk graphical user interface (GUI) will permit us to interact with our manipulator in real time by moving it with the generation of the desired angles trajectories and monitoring it by observing the developed trajectories and the other signals of concern like the control signals **Figure 4.46**.

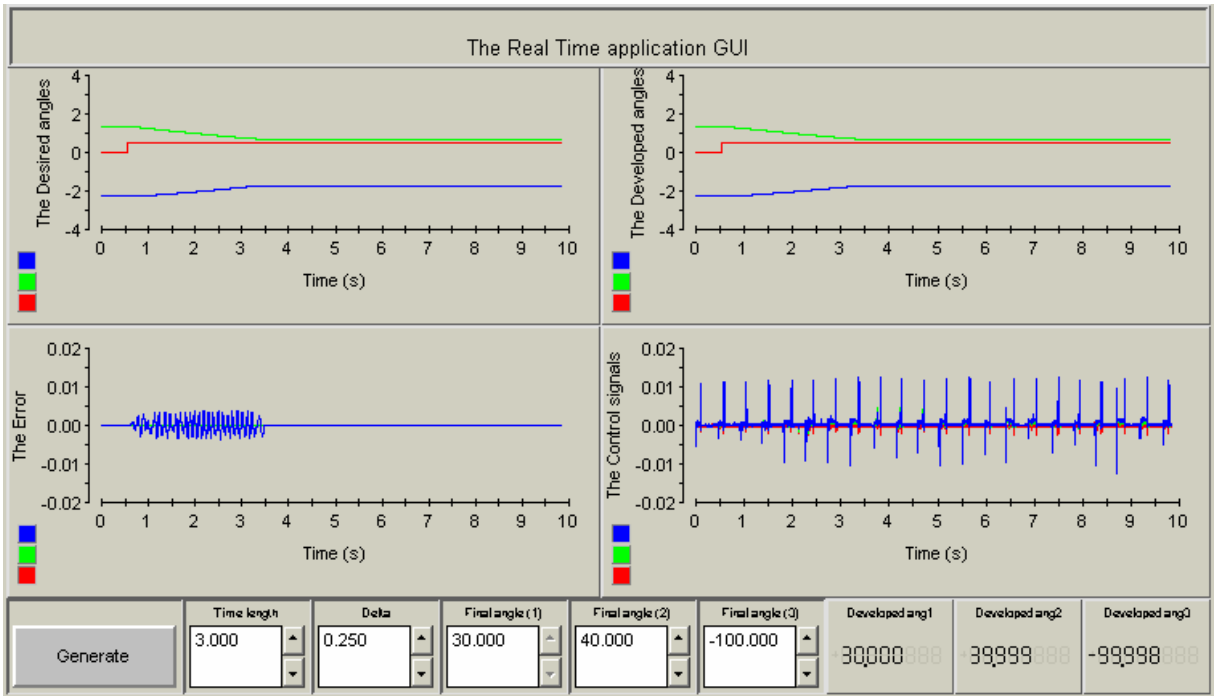


Figure 4.46: The application GUI.

IV.9.3 The Used Trajectories

For this real time application we will use different types of trajectories:

- Free trajectories: here we will test free trajectories in the (y, z) plane and in the 3D workspace, even though in the case of 3D trajectories we will delay one trajectory with respect to the others and this is due to the fact that the DSP controller board integrates only two Encoders which will enable us to move only two joints at a time. These trajectories will be applied only to PD/PID controllers, which inherently assume the manipulator as a linear decoupled system where the mutual effects of the joints movements are very small and can be neglected. For the computed torque controller, we will use only plane trajectories since the previous mutual effects cannot be neglected.
- Imposed trajectories: all the imposed trajectories tested here will be generated in the plane since the 3D imposed trajectories require the movement of the three joints at the same time which is impossible due to the same reasons.

All the results are recuperated in mat files and treated under Matlab.

IV.9.4 The Real Time Application Results

First and in order to have a comparative point of view of the real time data, all the trajectories tested with the real manipulator will be applied to the manipulator model with the same controllers and all the other conditions (sampling time – rating voltage…).

After obtaining the real and simulation data, the effects of the inevitable external perturbations and modelling errors will be illustrated.

In this application, we will show the effects of sampling time, PWM switching frequency, the rating voltage and we will illustrate the effect of perturbating masses during the execution of imposed trajectories.

Short time trajectories: Time length=1s and Δ=0.1s,
Long time trajectories: Time length=3s and Δ=0.25s.

The free trajectory used in the workspace is in **Figure 4.47**:

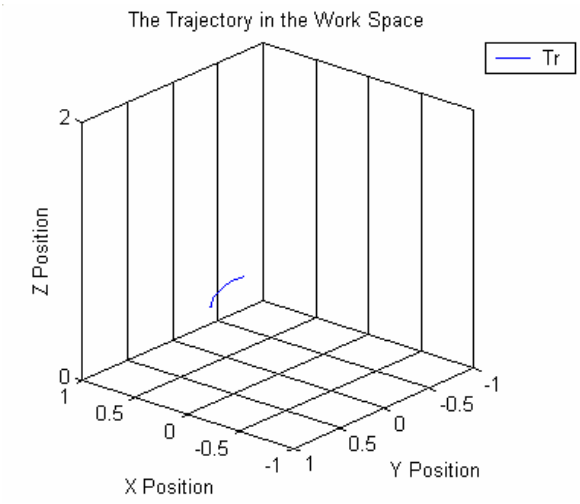


Figure 4.47: The workspace test trajectory.

IV.9.5 The PD Controller

1) Rating voltage=10v, sampling time=0.0002s, PWM switching frequency=3 MHz.

Long time trajectory

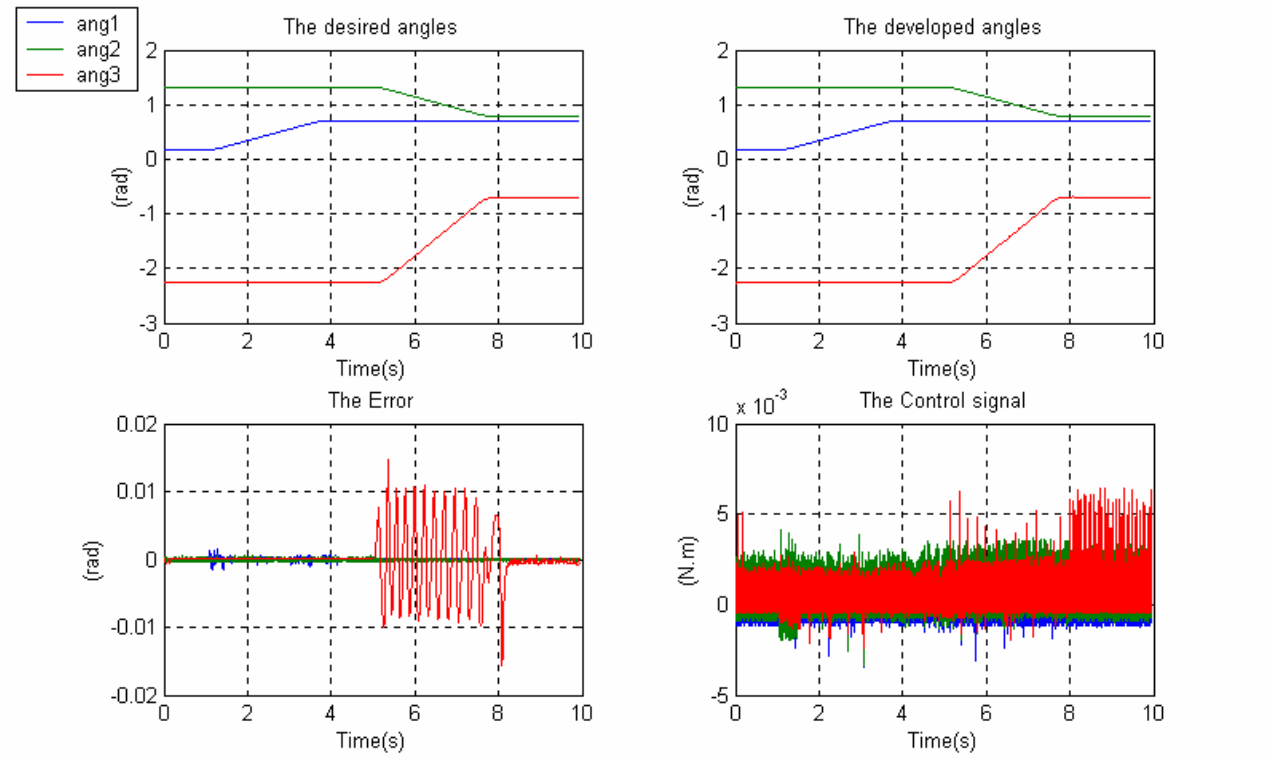


Figure 4.48

Short time trajectory

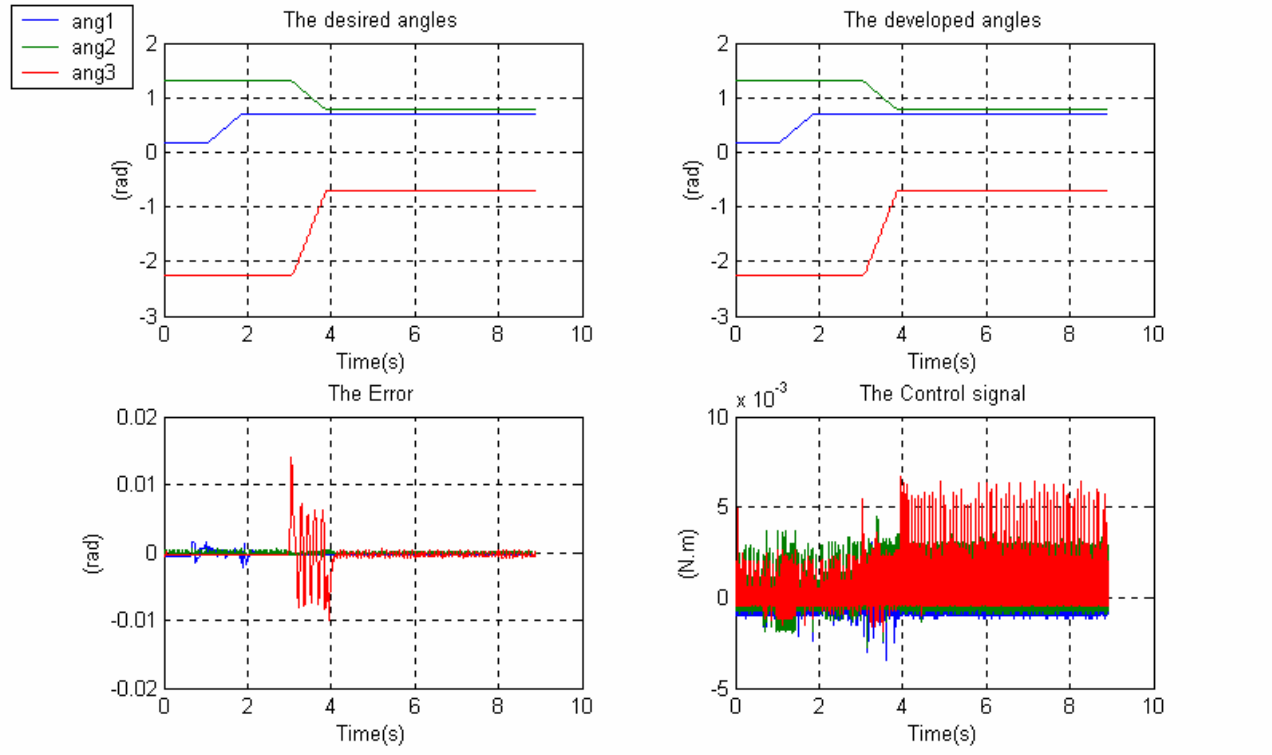


Figure 4.49

2) Rating voltage=10v, sampling time=0.0008s, PWM switching frequency=1 MHz.

Long time trajectory

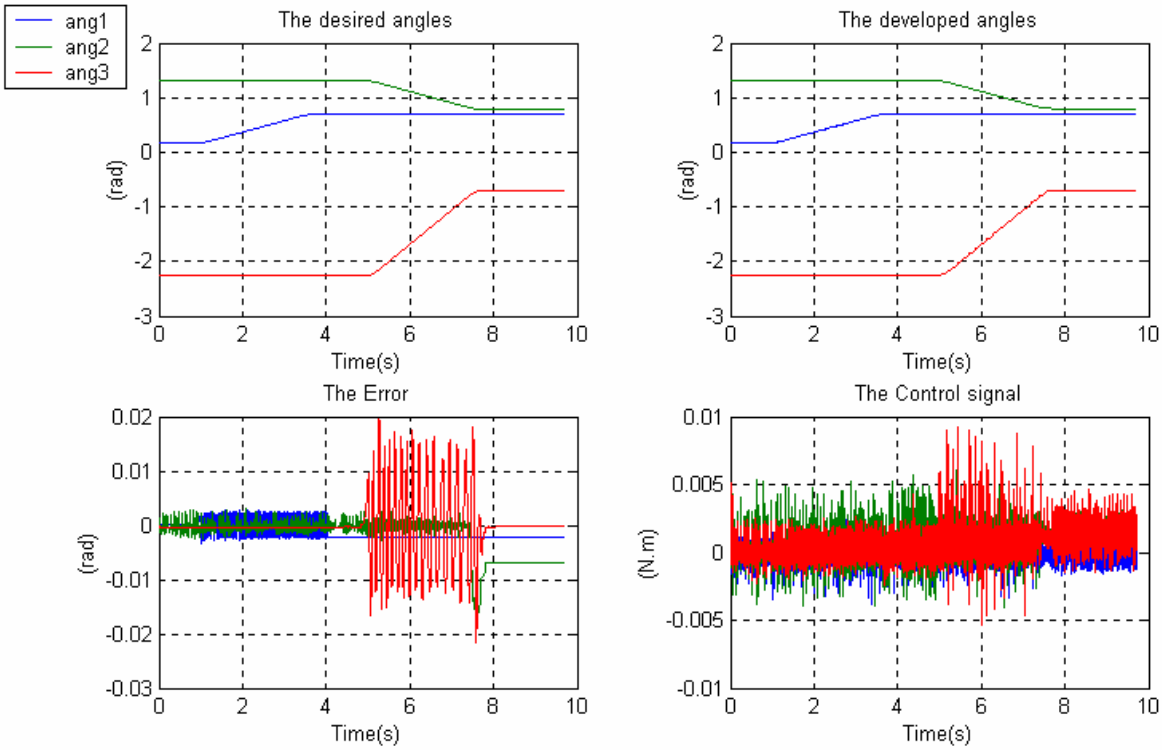


Figure 4.50

3) Rating voltage=18v, sampling time=0.0008s, PWM switching frequency=1 MHz.

Long time trajectory

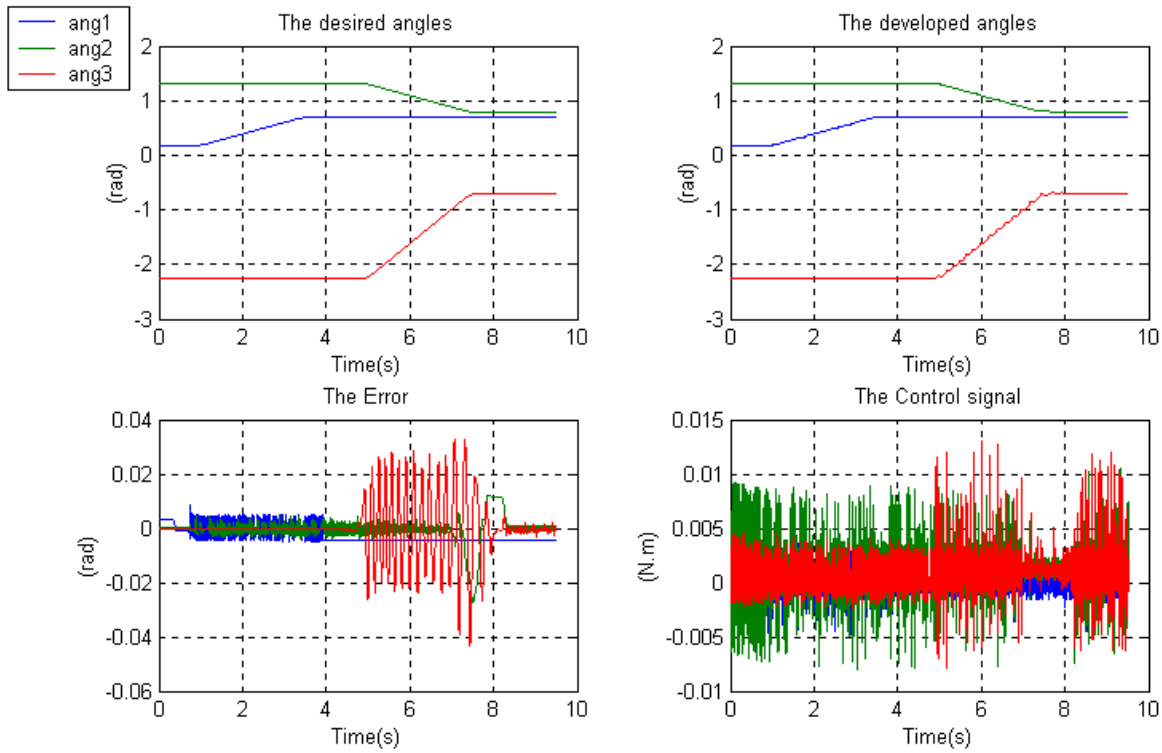


Figure 4.51

The following error signal is the difference between the simulation trajectory error and the real application trajectory error signals, this shows us the difference between the manipulator and motors models and the real manipulator with its motors.

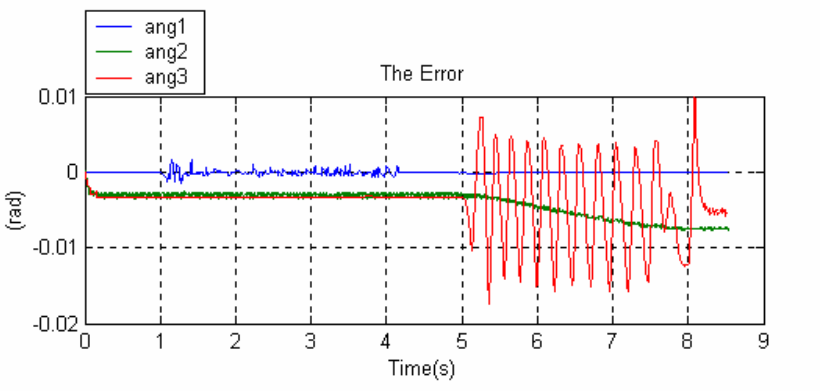


Figure 4.52

IV.9.6 The PID Controller

For the *ki* parameter for *link 3*, some refinements has to be done since the simulation parameter *ki*, even though it works very well with the model, it will introduce too much overshoots in the response of the application making the link and the overall system unstable and this is mainly due to the fact that the natural frequency which is assumed fixed and the same for the three links is actually different from one link to another.

1) Rating voltage=10v, sampling time=0.0002s, PWM switching frequency=3 MHz.

Long time trajectory

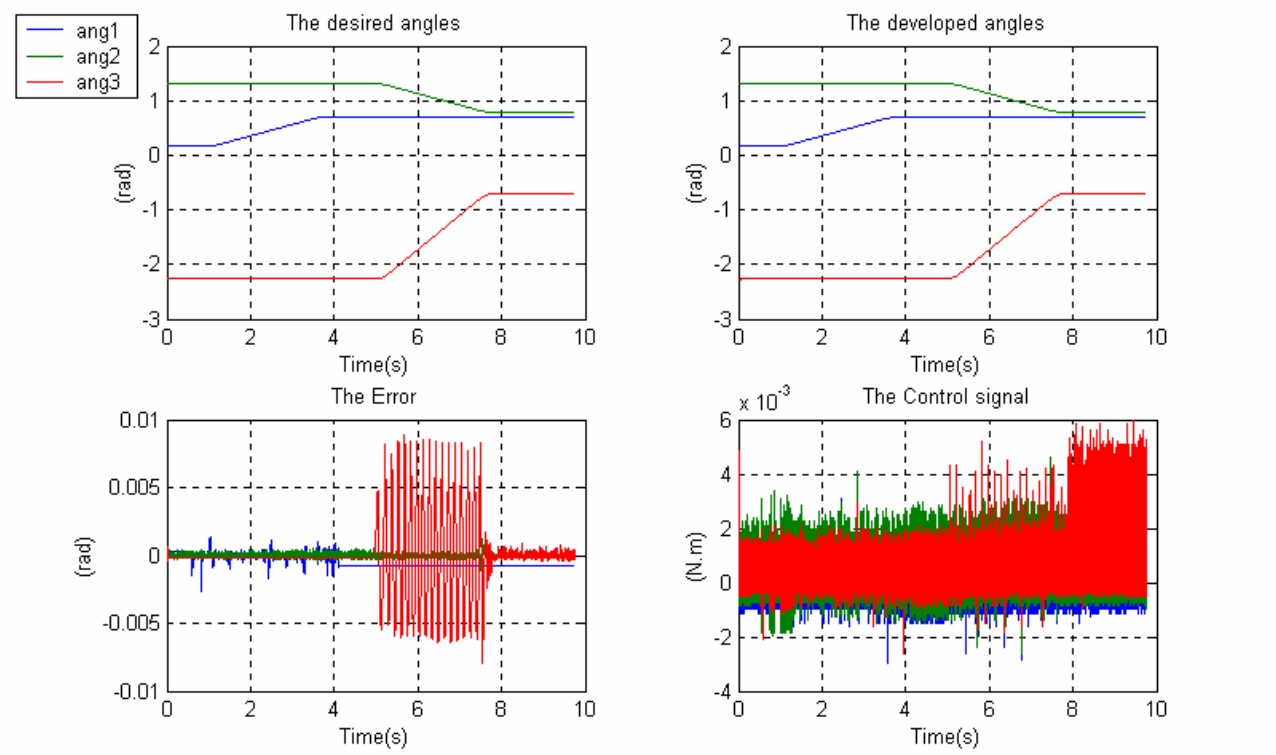


Figure 4.53

Short time trajectory

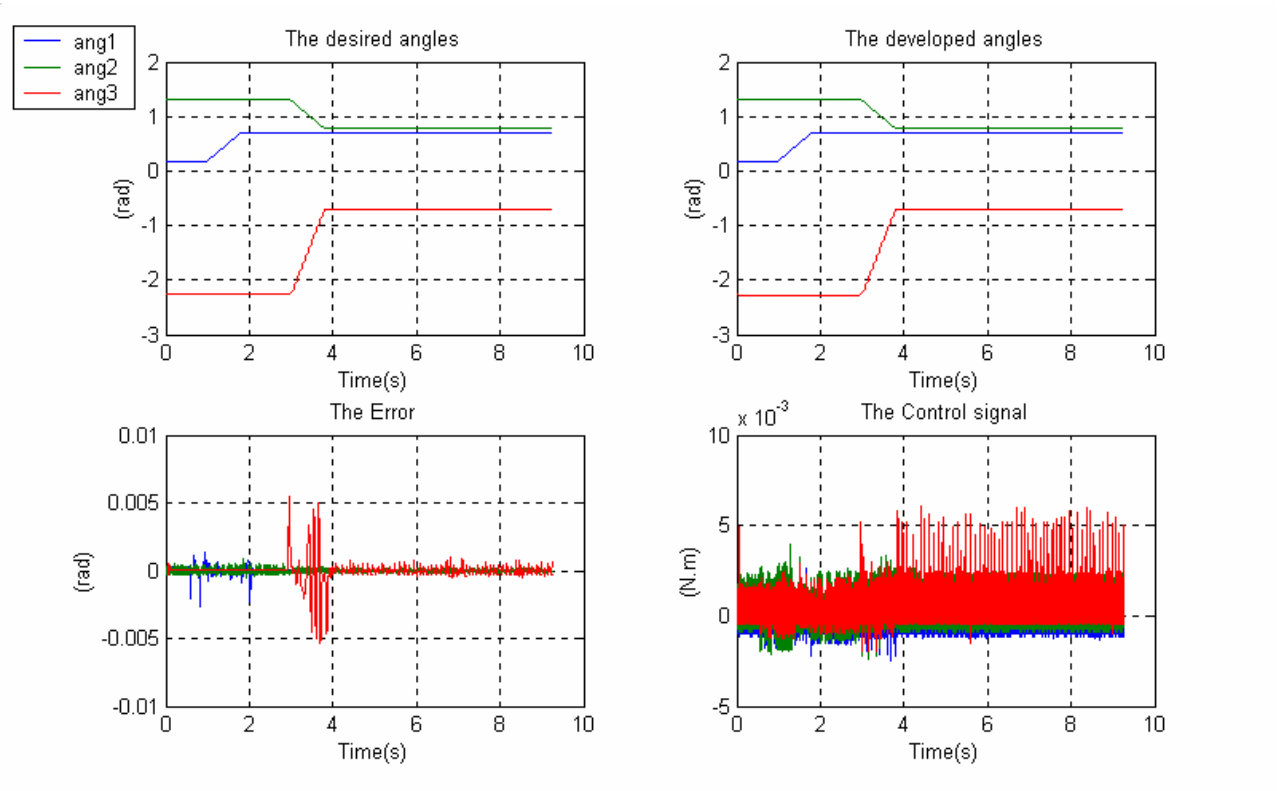


Figure 4.54

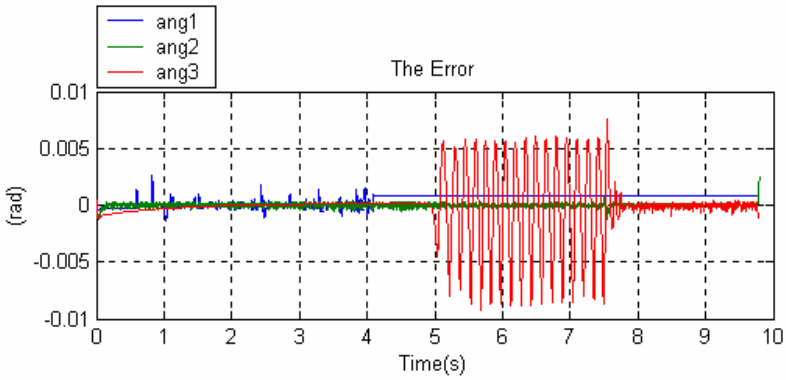


Figure 4.55

2) Rating voltage=10v, sampling time=0.0002s, PWM switching frequency=3 MHz without the predictive term in the controller.

Long time trajectory

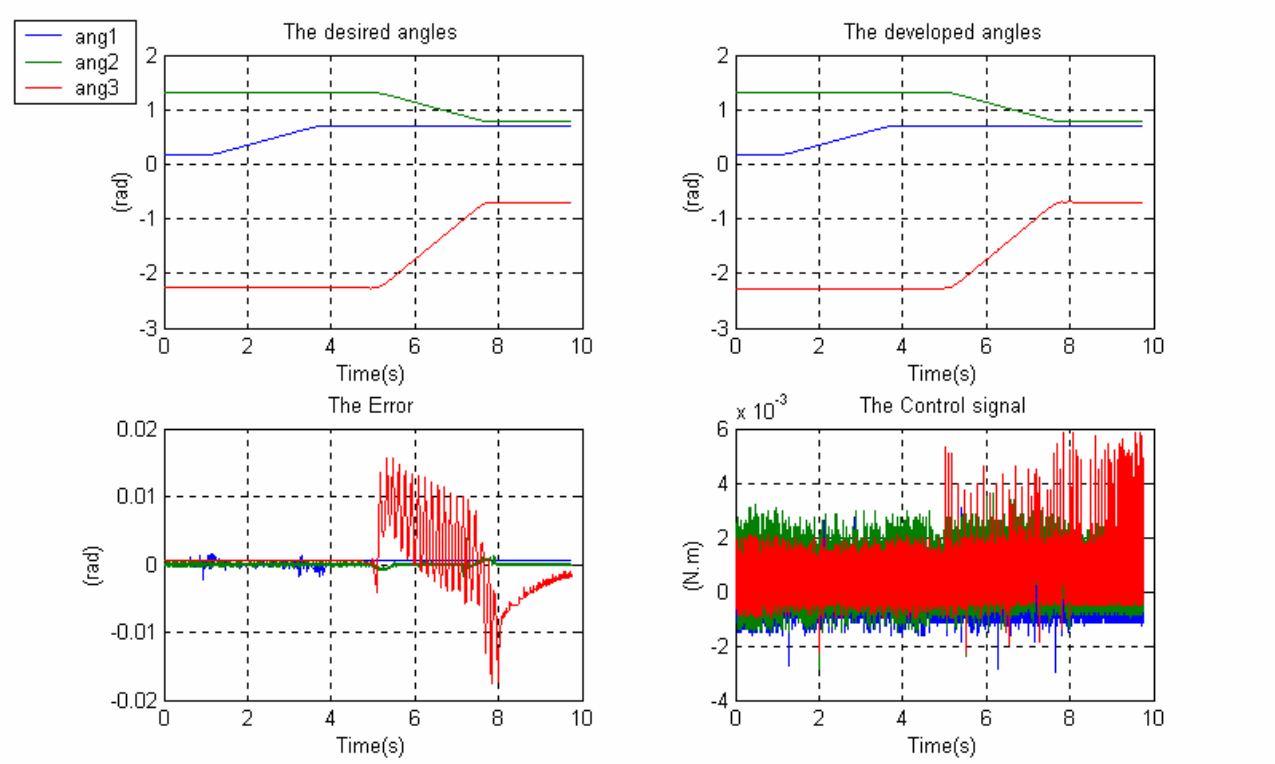


Figure 4.56

3) Rating voltage=10v, sampling time=0.0002s, PWM switching frequency=3 MHz without the predictive term in the controller with a LPF, a=3000.

Long time trajectory

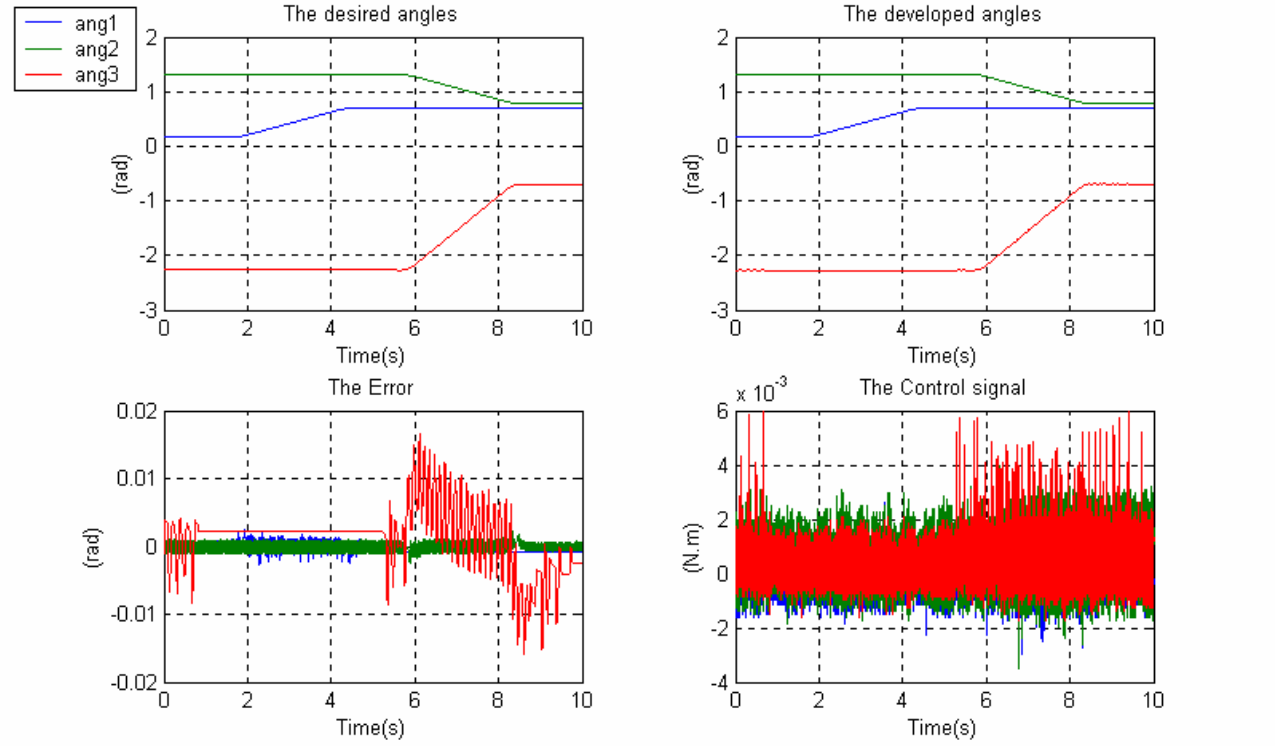


Figure 4.57

IV.9.7 Discussion

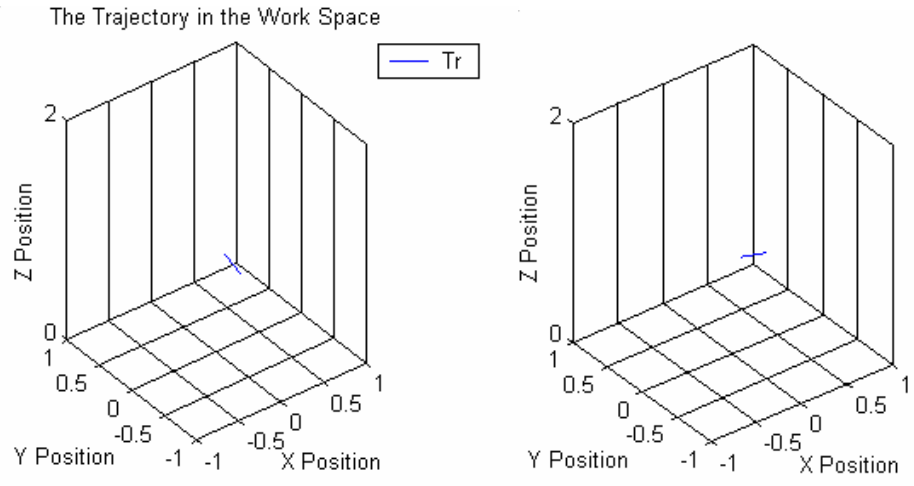
In **Figures (4.48-4.49)**, the PD controller with the simulation parameters shows good results both for long and short time trajectories and this is mainly due to the high PWM switching frequency and the low sampling period, this setting increases the performance of the PD controller and reduces the tracking error and at the same time keeping the control signal in a largely safe region. **Figures (4.50-4.51)** illustrate the deterioration in the tracking error caused by increasing the sampling period and reducing the switching frequency and note here that better tracking is obtained by smaller rating voltage, this can be explained by the fact that a regulation with smaller voltage pulses gives inevitably better results than the one with higher voltage pulses. **Figure 4.52** shows the difference between the tracking error in the simulation with the model and the real tracking error which is the difference between the two sets of developed angles with the same desired trajectory and this result shows the modelling errors, the external perturbations and the effect of using a PWM signal as a control signal.

In **Figures (4.53-4.54)**, the PID controller gives better tracking performance than the PD controller and eliminates at the same time the steady state error. **Figure 4.55** shows the "simulation - real" error for the PID controller. **Figures (4.56-4.57)** illustrate that taking off the predictive term will decrease the PID performance and this deterioration will not be overcome by the use of low pass filters "many values have been tested" which demonstrates the importance of the predictive term in reducing the tracking error.

IV.9.8 The PID and the Computed Torque Controller

IV.9.8.1 Imposed Trajectory Test

The straight-line trajectories used in the workspace are in **Figure 4.58**:



Tr_1 Tr_2

Figure 4.58: The imposed workspace test trajectories.

IV.9.8.2 The PID Controller

Rating voltage=15v, sampling time=0.0003s, PWM switching frequency=10 KHz, Tr_1.

Long time Tr_1

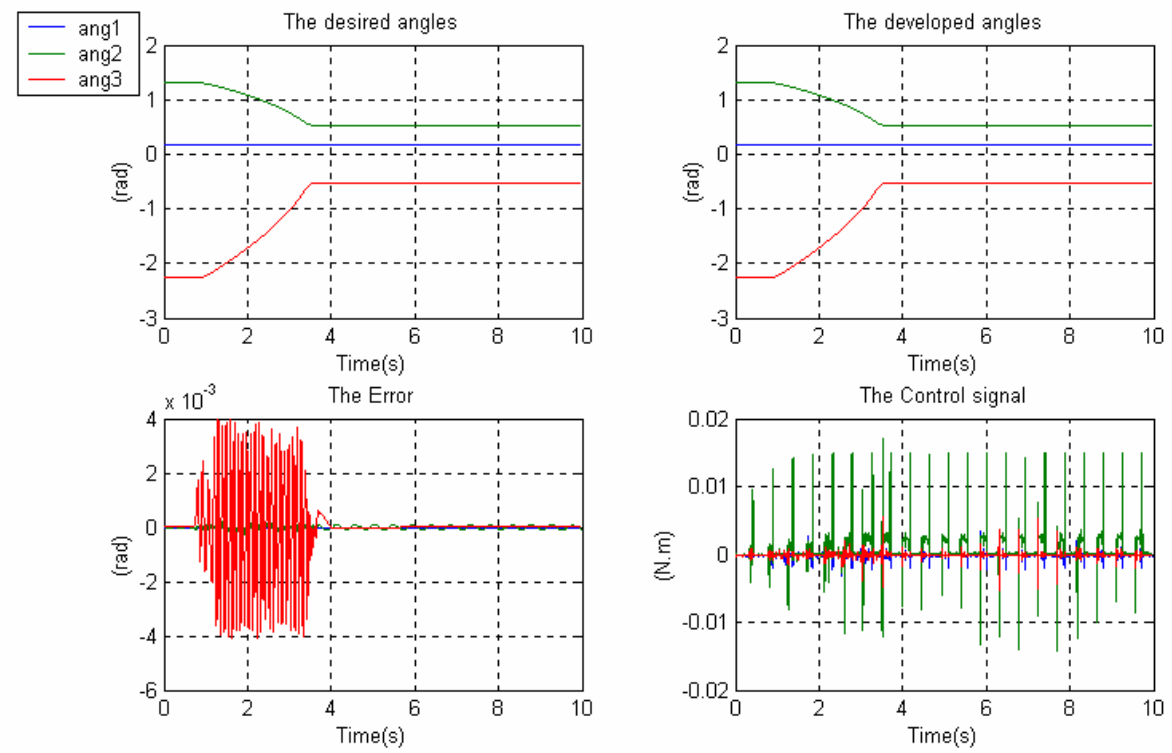


Figure 4.59

Short time Tr_1

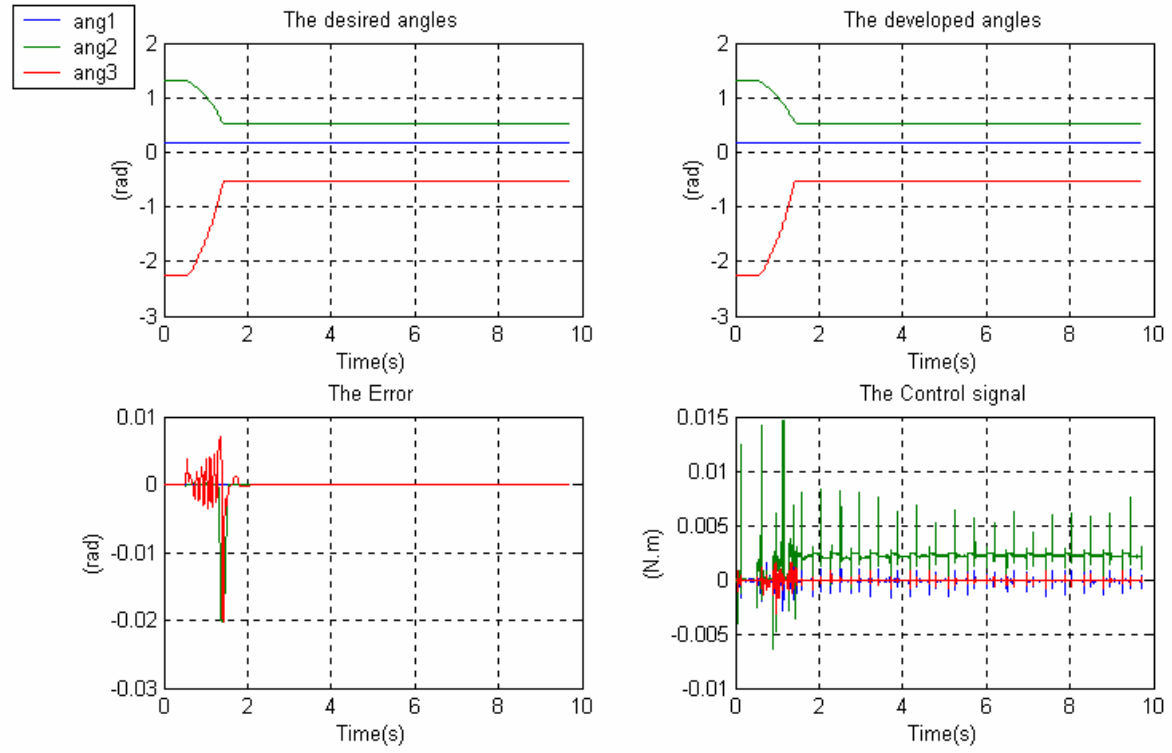


Figure 4.60

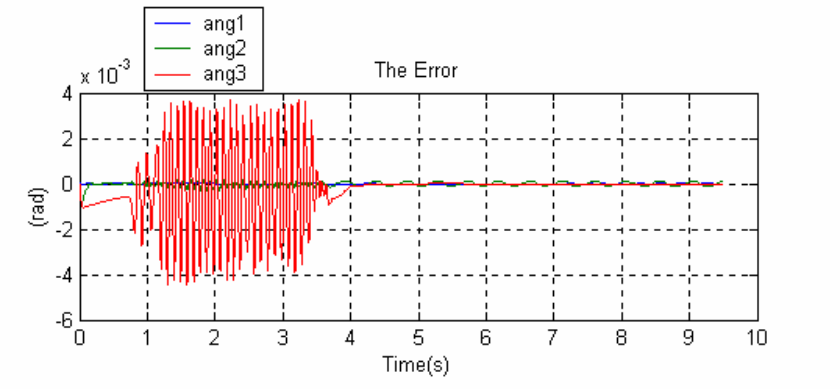


Figure 4.61

IV.9.8.3 The Computed Torque Controller

Rating voltage=15v, sampling time=0.0003s, PWM switching frequency=10 KHz, Tr_1.

Long time Tr_1

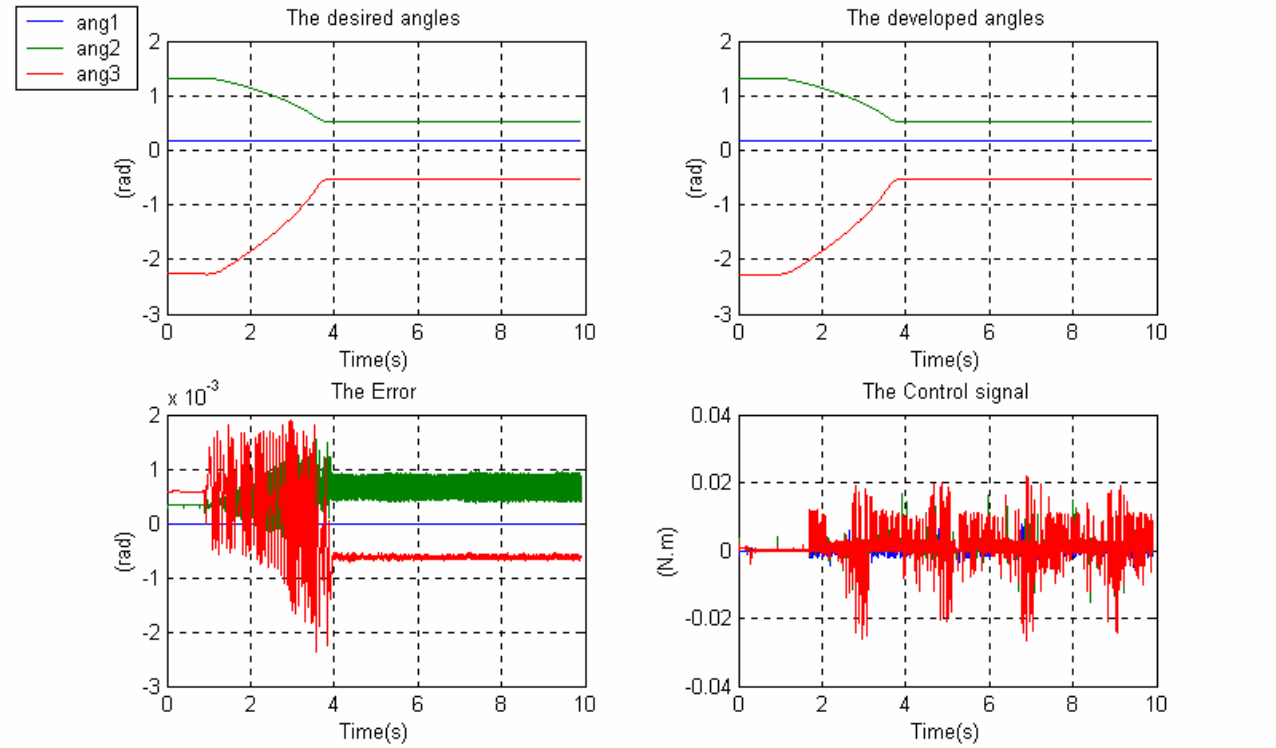


Figure 4.62

Short time Tr_1

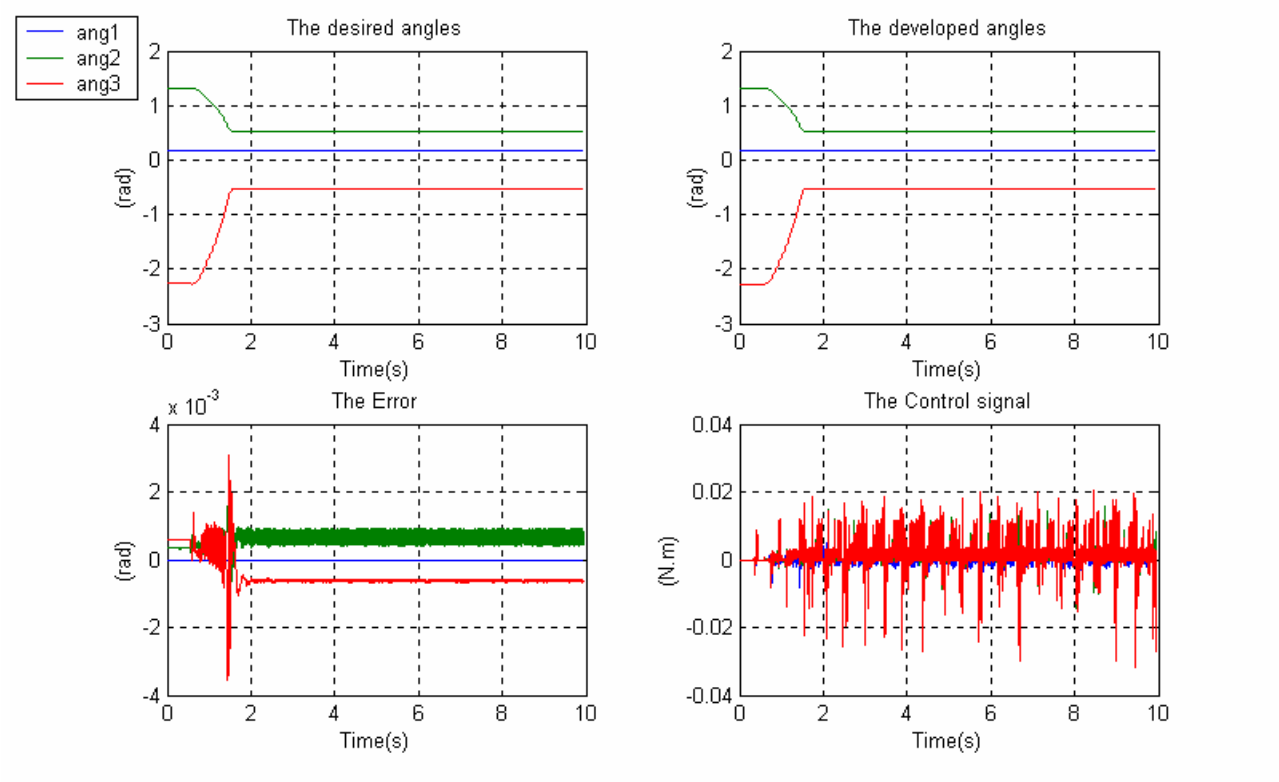


Figure 4.63

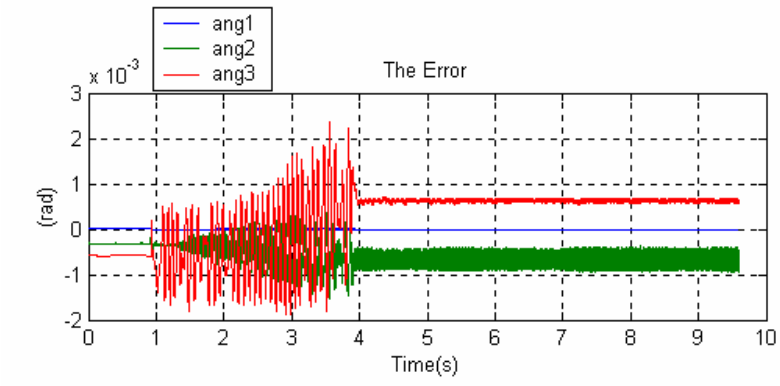


Figure 4.64

IV.9.9 Discussion

Figure 4.59 shows a very acceptable performance for the PID controller with imposed plane trajectories executed in a sufficiently long time but **Figure 4.60** shows a clear deterioration of this performance with imposed trajectories executed in a relatively short time and this can be considered as a major drawback of this kind of controllers limiting at the same time its fields of application. This drawback and the need of considering the manipulator dynamics in the control action has led to the apparition of the computed torque control and other robust control algorithms.

Figures (4.62-4.63) show the improvements in performance due to the use of the computed torque control especially for the high speed trajectories producing a fast response and an accurate trajectory tracking. **Figures (4.61-4.64)** show the effect of modelling errors and perturbations for long time trajectory for the PID and the computed torque controllers.

IV.9.10 PID Regulation with Perturbations

For these simulations, long time Tr_2 is used with a PID controller.

1) Rating voltage=15v, sampling time=0.0003s, PWM switching frequency=10 KHz, Tr_2 with a perturbating mass m=0.4Kg from the beginning, Time length=3s.

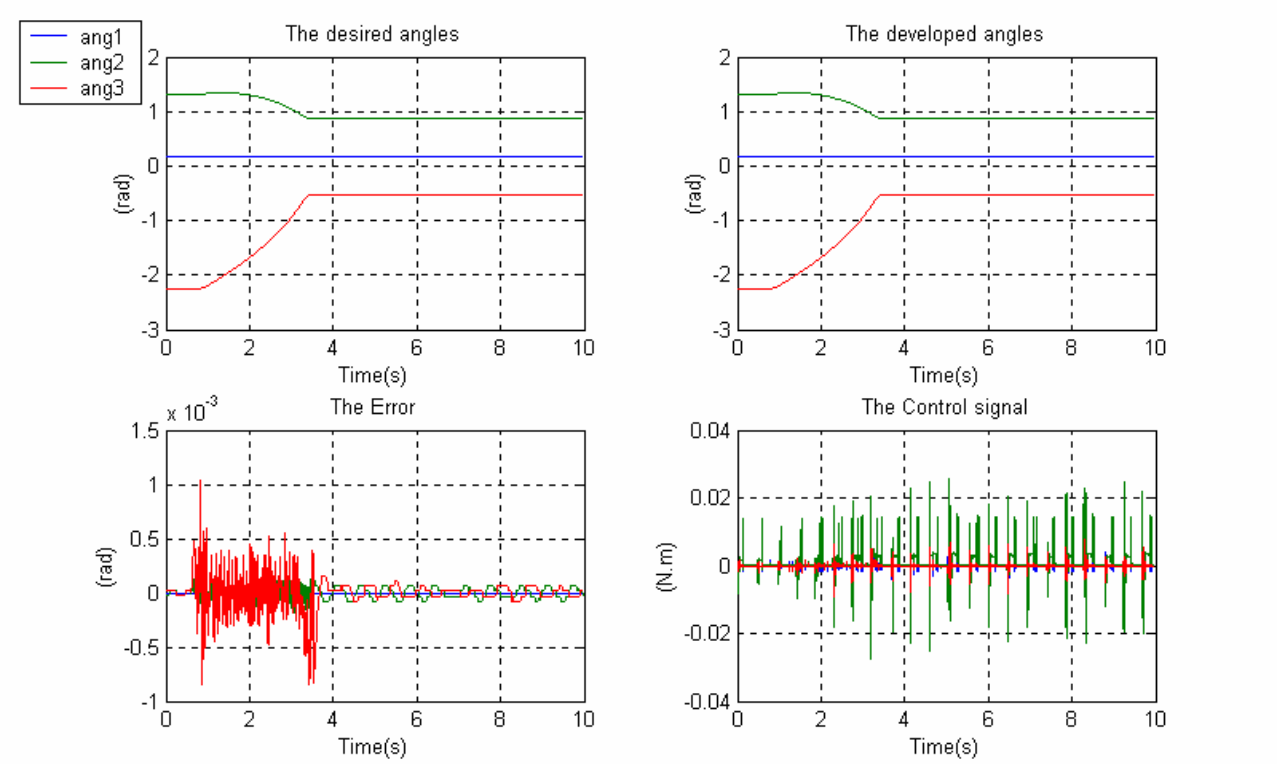


Figure 4.65

2) Rating voltage=15v, sampling time=0.0003s, PWM switching frequency=10 KHz, Tr_2 with a perturbating mass m=0.4Kg from the middle until the end, Time length=10s.

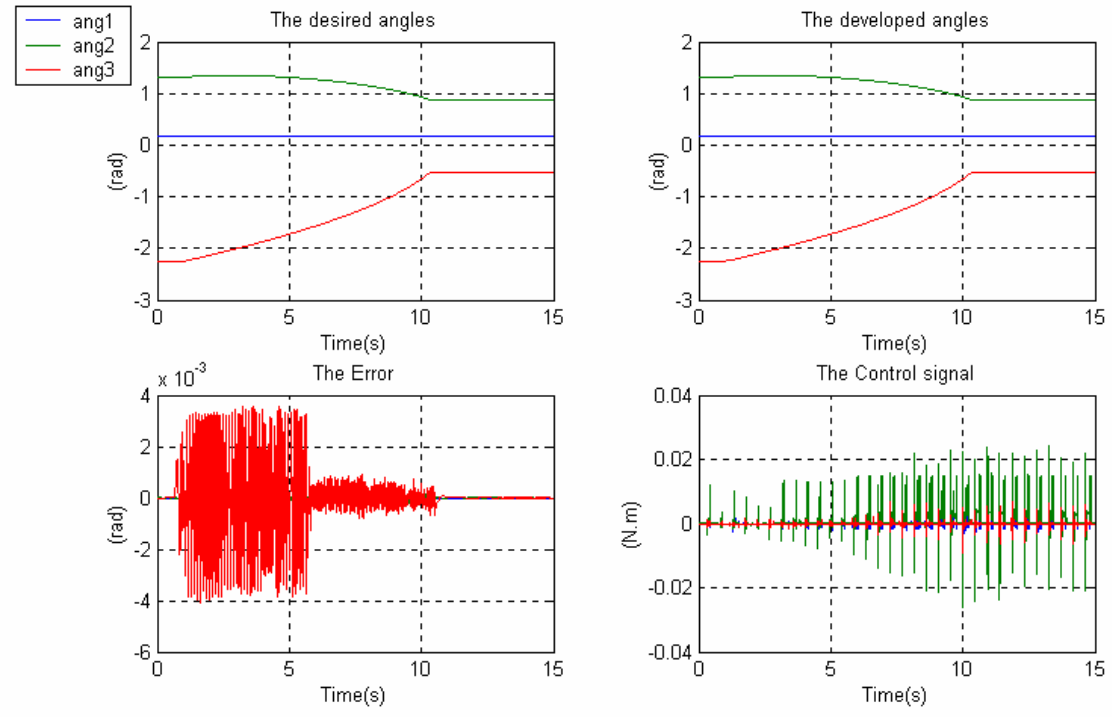


Figure 4.66

3) Rating voltage=15v, sampling time=0.0003s, PWM switching frequency=10 KHz, Tr_2 with a perturbating mass m=0.4Kg from the beginning and taken off at the middle, Time length = 10s.

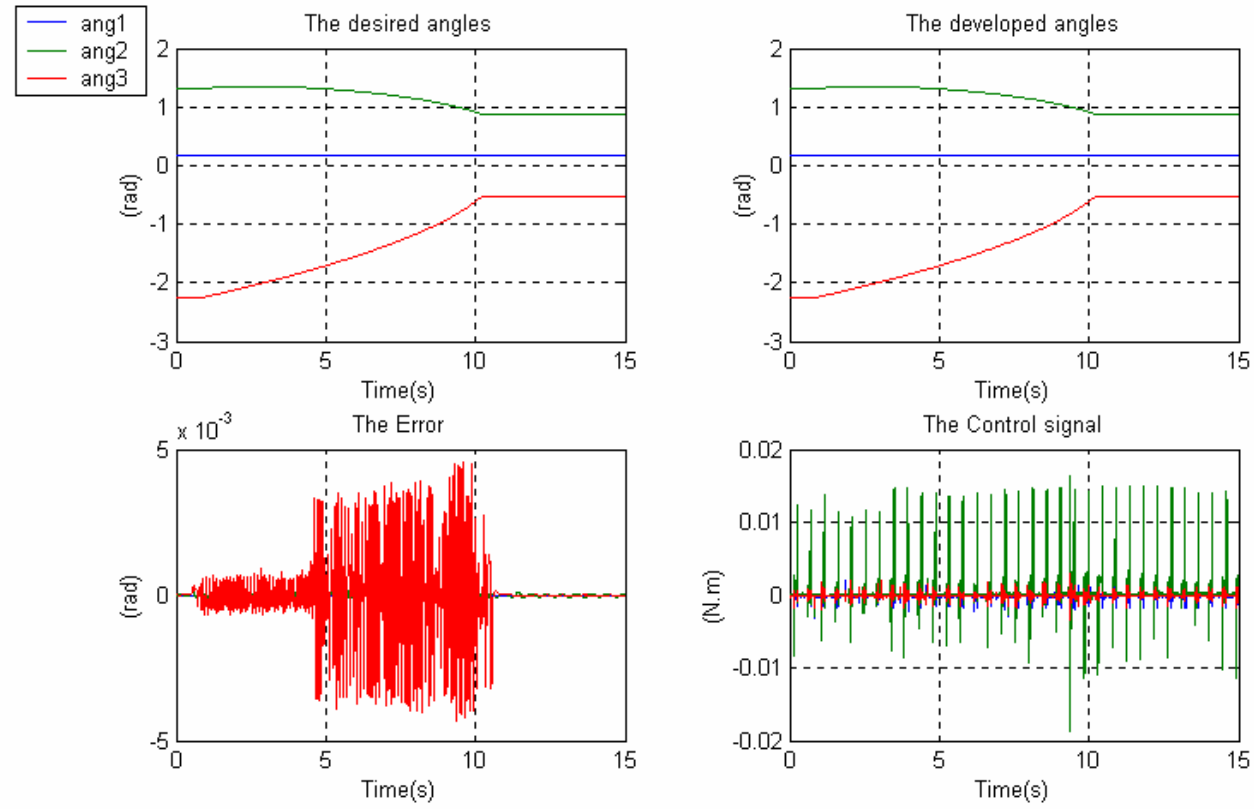


Figure 4.67

IV.9.11 Discussion

In these simulations, we have shown the behaviour of the PID controller in the presence of a controlled perturbation caused by a mass of 400g that will be applied at the link*3* end while the manipulator is executing its trajectory in three cases:

In **Figure 4.65**, the perturbation is applied from the beginning and we can see an increase in the control signal to compensate the perturbating torque caused by the mass while the tracking error is relatively not affected.

In **Figure 4.66**, the mass is applied from the middle until the end of the trajectory from which the control signal will increase to compensate the mass effect, and at the same time, the applied mass will decrease the oscillations caused by the PWM signal improving the tracking error.

In **Figure 4.67**, the mass is applied from the beginning and taken off at the middle and at this time, the PID performance is more affected especially for the tracking error.

IV.10 Conclusion

In this chapter, we have tested the different control algorithms introduced in chapter I using in the first part the manipulator and the dc motors models derived in chapter III under simulink with the same simulation parameters used by the DSP Card. In the second part of this chapter, we have tried to test the same control algorithms on the real manipulator structure.

The simulation and the real time application results are in the same way concerning the performances of the different controllers with an inevitable error between the real manipulator and its model responses due mainly to the modeling errors and the practical external and internal perturbations inexistent during the simulation; these results are also due to the computing power provided by the DSP platform which permits very low sampling periods increasing the performances of the used controllers.

GENERAL CONCLUSION

In this thesis, we have presented the work that has been done in studying, designing, implementing a 3dof manipulator and controlling it in a DSP based environment provided by the used DSP Card and trying, at the same time, to illustrate the theory to be known to do this and the steps to be followed in order to model the mechanical structure of a manipulator from determining the homogenous transforms relating its links frames to the motion equations taken as the base of the model and may take part in the design of some controller algorithms.

New complex control algorithms like the adaptive control schemes or the variable structure control systems may require an online computation of the manipulator dynamic parameters and even though the control systems used for almost all the industrial robots are based on the linear control theory, the continuous progress in the processors technologies and the increase in the memory as well as the computation performances will ensure certainly a very promising future for them.

During the simulations of the control algorithms, we always assumed a continuous behaviour of the control systems and the manipulator model without delays which is not the case for the real practical application in which the processor speed, the sampling time, the switching frequency of the PWM signal and the size of the real time application have to be taken into account leading to a digital control theory vision of the application considering it as a discrete one and the only solution to be as close as possible to the simulation results is to make the sampling time as small as possible compared to the mechanical pass band of the controlled mechanical structure. The computing power provided by the DSP Card allows us to test our manipulator with a sampling time varying between (0.0002-0.0003 (s)) which explains the quality of the real application data results as compared with the simulation one without forgetting the errors introduced by the modelling precisions and the external perturbations.

The development environment provided by the DSP Card permits the testing of approximately all the available control algorithms regardless of their complexity on real plants and a way to take maximum advantage of the DSP Card computing power is to implement these control algorithms as s-functions written in **C** and avoiding as much as possible the use of the simulink blocks.

Perspectives

For future perspectives, the presence of such a very time efficient development environment and implementation procedure as well as the computation power and the tools provided by the DSP Card will enable us to enlarge the fields of application on which we can use this Card like the new emerging control algorithms, digital signal processing new techniques, computer networks applications and why not using this tool as an aided teaching tool by which the students can test on real plants the theoretical knowledge acquired on the course.

Appendix 1: Pseudo-Inverses and Singular-Value Decomposition

1. Pseudo-Inverses

The pseudo-inverse is a generalization of the inverse of a matrix to the case of singular or even nonsquare matrices. In the field of robotics, pseudo-inverses are often used in the identification of manipulator dynamics and in the control of redundant manipulator.

In general, for every finite $m \times n$ matrix A, there is a unique $n \times m$ matrix A^+ satisfying the following four conditions:

$$A A^+ A = A \qquad \text{(A1-1)}$$

$$A^+ A A^+ = A^+ \qquad \text{(A1-2)}$$

$$(A A^+)^T = A A^+ \qquad \text{(A1-3)}$$

$$(A^+ A)^T = A^+ A \qquad \text{(A1-4)}$$

this A^+ is called the pseudo inverse of A.
the pseudo inverse has many properties among them:

1 - $(A^+)^+ = A.$ (A1-5)

2 - If a $m \times n$ matrix A satisfies *rank (A) = m*, then:

$$A^+ = A^T (A A^T)^{-1} \qquad \text{(A1-6)}$$

3 - Consider the system of simultaneous linear equations given by:

$$Ax = b \qquad \text{(A1-7)}$$

where A is a known $m \times n$ matrix, b is a known m-dimensional vector, and x is an unknown n-dimensional vector. When A is square $(m = n)$ and nonsingular, the solution x is given by :

$$x = A^{-1} b \qquad \text{(A1-8)}$$

but when A is singular, the best approximate solution of the equation is not unique and has the general form:

$$x = A^+ b + (I - A^+ A) k \qquad \text{(A1-9)}$$

where k is an arbitrary n-dimensional vector. Further-more, the solution that minimizes its own norm $\|x\|$ is given by:

$$x = A^+ b \qquad \text{(A1-10)}$$

2. Singular Value Decomposition

We will describe a scheme to obtain the singular value decomposition of a $m \times n$ matrix A:

1 - Obtain the nonnegative eigen values $(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_r > 0)$ and the eigen vectors $(v_1, v_2, ..., v_r)$ of the nonnegative matrix $A^T A$.

2 - We define the term $\sigma_i = \sqrt{\lambda_i}$, such that $i = 1, 2, ..., min(m,n)$.

3 - We obtain the orthogonal matrices U $((m \times m)$ and $V (n \times n)$ as follows:

a. we define a diagonal matrix $\sum_r$ using the r nonzero term σ_i :

$$\sum_r = \begin{bmatrix} \sigma_1 & & 0 \\ & \ldots & \\ 0 & & \sigma_r \end{bmatrix} \qquad \text{(A1-11)}$$

and let :

$$V_r = [v_1, v_2, \ldots, v_r] \qquad \text{(A1-12)}$$

we can obtain U_r using the equation :

$$U_r = A V_r \sum_r^{-1} \qquad \text{(A1-13)}$$

b. the remaining part of $\sum_r$ is filled by zeros to obtain $\sum$, the remaining parts of U_r and V_r are completed to obtain U and V such that the following equations are satisfied :

$$U^T U = U U^T = I_m \qquad \text{(A1-14)}$$

$$V^T V = V V^T = I_n \qquad \text{(A1-15)}$$

we can now express the matrix A as the product of three matrices :

$$A = U \sum V^T \qquad \text{(A1-16)}$$

the right hand side of equation (A1-*16)* is called the *singular-value decomposition*, and σ_i*(i=1,2,...,min(m,n))* are called *singular values*. The number of nonzero singular values is *r=rank(A)* .

Appendix 2: Practical Considerations

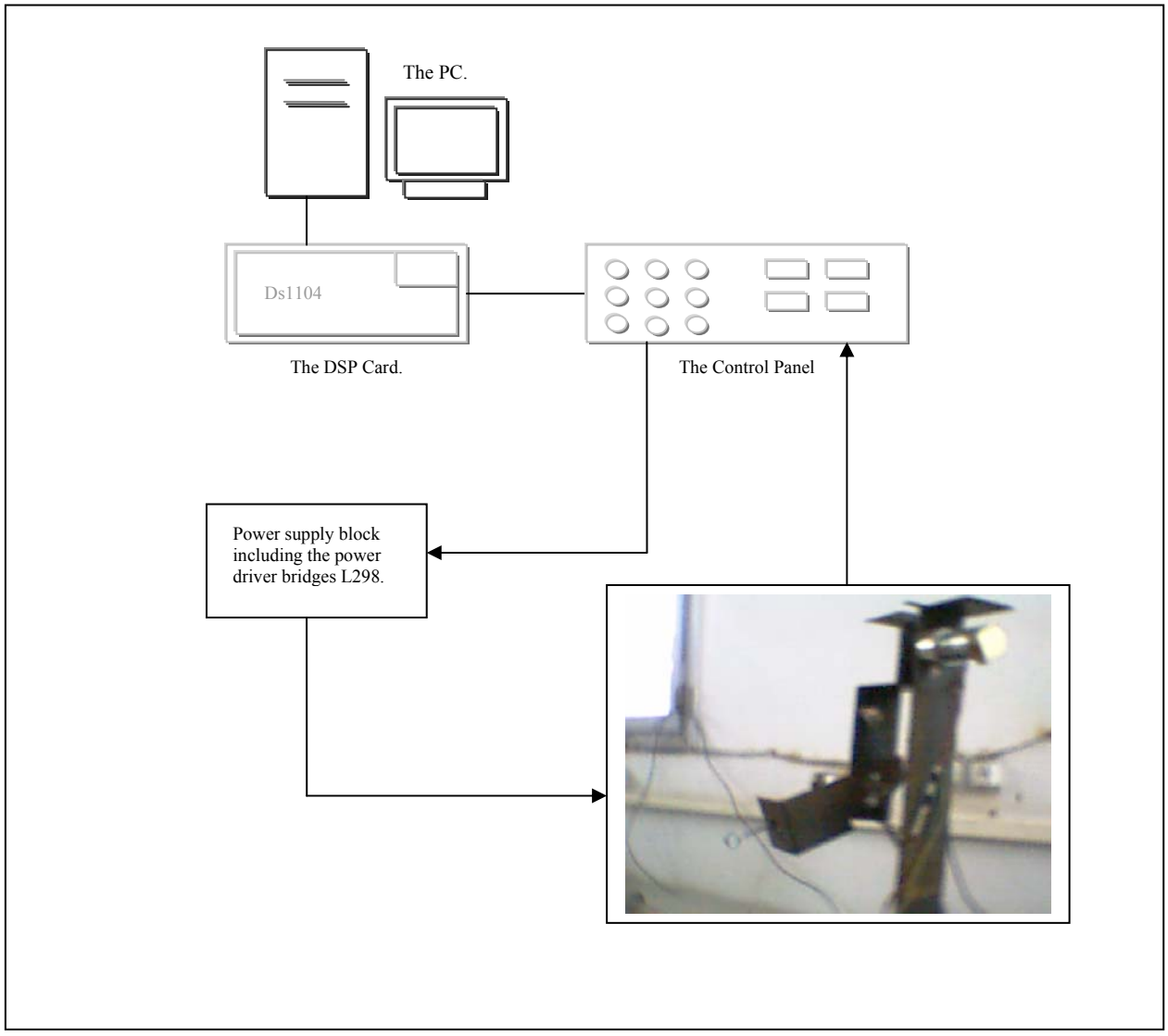


Figure A2.1: General hardware structure.

The LM298 is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. The LM298 driver is used to supply the DC Motors with the required PWM voltage at the supply voltage rating corresponding to the TTL DSP Card PWM output. The power supply block contains in addition to the LM298, the necessary circuitry to control the sense of rotation.

Its features:

- Power supply voltage up to 46V.
- 2A output per channel.
- Low saturation voltage.
- Thermal shutdown protection.
- Logical ``0'' input voltage up to 1.5V (High noise immunity).
- Provides two pins to connect two current sensing resistors (the resistors used are of values 2.7 ohm).

Data sheet of the LM298 can be obtained from the website:
http://me118.stanford.edu/pictures/Win01Projects/TheBeast.

2. The Actuators

The actuators used for the application are DC servo Gearmotors of maximum reference voltage of 24v and a reduction ratio of the integrated gear of 65.5; it integrates also an Encoder with a resolution of 500 counts per revolution (CPR).

Data sheet of the DC motor in: http://www.pennmotion.com.

Data sheet of the H.P.Encoder Module 9100 in: http://www.secomtel.com.

3. DSP Card Practical Aspects

- The output PWM signal channels when not used are defined logical high, in order to force them to a defined TTL logical low level, an external pull-down resistor of about 1Kohm can be connected to GND.
- The ADCs used are 3 of the 4 separated channels.

Data sheet of the DSP Card in: http://www.dspace.fr.

Appendix 3: The Pseudo Inertia Matrix

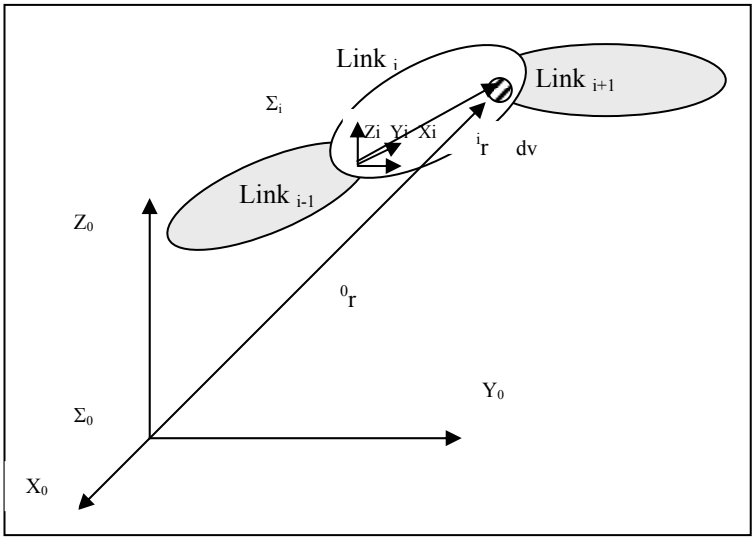


Figure A3.1: Point ^{i}r on link i.

The pseudo inertia matrix is defined by:

$$\hat{H}_i = \int_{Link\ i} {}^{i}r\ {}^{i}r^{T}\ \rho\ d\upsilon \qquad \textit{(A3-1)}$$

$$\hat{H}_i = \begin{bmatrix} \dfrac{-\hat{I}_{ixx}+\hat{I}_{iyy}+\hat{I}_{izz}}{2} & \hat{H}_{ixy} & \hat{H}_{ixz} & m_i\hat{s}_{ix} \\ \hat{H}_{ixy} & \dfrac{\hat{I}_{ixx}-\hat{I}_{iyy}+\hat{I}_{izz}}{2} & \hat{H}_{iyz} & m_i\hat{s}_{iy} \\ \hat{H}_{ixz} & \hat{H}_{iyz} & \dfrac{\hat{I}_{ixx}+\hat{I}_{iyy}-\hat{I}_{izz}}{2} & m_i\hat{s}_{iz} \\ m_i\hat{s}_{ix} & m_i\hat{s}_{iy} & m_i\hat{s}_{iz} & m_i \end{bmatrix} \qquad \textit{(A3-2)}$$

$d\upsilon$ is The volume of the differential element, ρ is its density, $^{i}\hat{s}_i = \left[\hat{s}_{ix}, \hat{s}_{iy}, \hat{s}_{iz}\right]^T$ is the vector from O_i to the centre of mass of *link* $_i$ expressed in Σ_i and $^{i}r = \left[{}^{i}r_x, {}^{i}r_y, {}^{i}r_z\right]$ is its position, the elements of $\hat{H}_i$ are given by:

$$\hat{I}_{ixx} = \int_{Link\ i} ({}^{i}r_y^2 + {}^{i}r_z^2)\ \rho\ d\upsilon \quad \text{(Moment of inertia),} \qquad \textit{(A3-3a)}$$

$$\hat{H}_{ixy} = \int_{Link\ i} {}^{i}r_x\ {}^{i}r_y\ \rho\ d\upsilon \quad \text{(Product of inertia),} \qquad \textit{(A3-3b)}$$

$$m_i = \int_{Link\ i} \rho\ d\upsilon \quad \text{(Mass of link i),} \qquad \textit{(A3-3c)}$$

$$\hat{s}_{ix} = \int_{Link\ i} {}^{i}r_x\ \rho\ d\upsilon / m_i \quad \text{(Center of mass),} \qquad \textit{(A3-3d)}$$

and by similar equations for $\hat{I}_{iyy}, \hat{I}_{izz}, \hat{H}_{ixz}, \hat{H}_{iyz}, \hat{s}_{iy},$ and $\hat{s}_{iz}$. Since $\hat{H}_i$ is defined with respect to Σ_i fixed to *link* $_i$, the value of $\hat{H}_i$ is a constant and independent of $\boldsymbol{q}$.

The inertia tensor of *link* $_i$ with respect to Σ_i is given by:

$$ {}^{i}\hat{I}_i = \begin{bmatrix} \hat{I}_{ixx} & -\hat{H}_{ixy} & -\hat{H}_{ixz} \\ -\hat{H}_{ixy} & \hat{I}_{iyy} & -\hat{H}_{iyz} \\ -\hat{H}_{ixz} & -\hat{H}_{iyz} & \hat{I}_{izz} \end{bmatrix} \qquad \textit{(A3-4)} $$

The inertia tensor of *link* $_i$ with respect to the frame with its origin at the centre of mass (which is generally different from the origin of Σ_i) and with its coordinate axes parallel to those of Σ_i . This tensor is denoted by:

$$ {}^{i}I_i = \begin{bmatrix} I_{ixx} & -H_{ixy} & -H_{ixz} \\ -H_{ixy} & I_{iyy} & -H_{iyz} \\ -H_{ixz} & -H_{iyz} & I_{izz} \end{bmatrix} \qquad \textit{(A3-5)} $$

The following relations hold between elements of ${}^{i}\hat{I}_i$ and ${}^{i}I_i$:

$$ \hat{I}_{ixx} = I_{ixx} + m_i(\hat{s}_{iy}^2 + \hat{s}_{iz}^2), \qquad \textit{(A3-6a)} $$

$$ \hat{H}_{ixy} = H_{ixy} + m_i \hat{s}_{ix} \hat{s}_{iy} . \qquad \textit{(A3-6b)} $$

Similar relations also hold for $\hat{I}_{iyy}, \hat{I}_{izz}, \hat{H}_{iyz},$ and $\hat{H}_{ixz}$.

Values concerning the manipulator used can be derived from the pages 50, 52.

BIBLIOGRAPHY

[1] Tsuneo Yoshikawa. “Foundation of Robotics: Analysis and Control”. The MIT Press, Cambridge, Massachusetts London, England 1990.

[2] Wisama Khalil, Etienne Dombre. " Modélisation, Identification et commande des robots". 2nd Edition, Hermes Science publications, Paris 1988, 1999.

[3] John J.D’azzo, Constantine H. Houpis. “Linear Control System Analysis and Design», McGraw-HILL International Book Company 1981.

[4] Nasser Kehtarnavaz, Mansour Keramat. “DSP System design using the TMS320C6000“, Prentice Hall, USA 2001.

[5] Gérard Blanchet et Patrick Devriendt "Processeur de traitement numérique du signal (DSP)". Technique de l'ingénieur, traité Electronique 2002.

[6] The web site: www.cat.csiro.au/cmst/staff/pic/robot

[7] The web site: www.srl.gatech.edu

[8] ‘Understanding data converter’ Texas Instruments Application report 1995.

[9] Matlab 6.0 - Help file.

[10] Hamdani, Rouiki ‘Conception et réalisation d’une carte de commande à base de microcontrôleur pour Robot scara’. ENP-LCP-, final year project 2003.

[11] Digital Control Systems, fall 2001, Charles Brice, University of South Carolina, a University course.

[12] The web site: www.engin.umich.edu.

[13] The web site: www.howstuffworks.com

[14] Nicolas Séguy “Système de commande d’un manipulateur”. Technique de l'ingénieur, Traité Informatique industrielle 2002.

[15] The web site: www.umich.edu

[16] K.A. Connor “Introduction to Engineering Electronics-Motors and Actuators-”, a University lecture 2003.

[17] Leon Zlajpah and Ivan Godler “Robot control design by using -Manipulator In the Loop-Simulation”, Jozef Stefan Institute, Slovenia 1998.

[18] DSP Based Electric Drives Laboratory-User Manual-, Department of Electrical and Computer Engineering, university of Minnesota 2003.

[19] Elio Abiakel "Development of an undergraduate laboratory course in control systems", a master of science thesis, The Ohio State University -2003.

[20] dSPACE "ds1104 " manuals and documents, Release 3.4 – May 2002.

[21] R. P. Paul, "Robot manipulators: mathematics, programming, and control. The Computer Control of Robot Manipulators", The MIT Press, Cambridge, 1983.

[22] O. Khatib, K. Yokoi, O. Brock, K.-S. Chang, and A. Casal: "Robots in human environments: Basic autonomous capabilities." International Journal of Robotics Research, 18(7):684-696, 1999.

[23] M.W. Spong, and M. Vidyasagar, Robot Dynamics and Control, John Wiley & Sons, Inc., New York, NY, 1989.

[24] John J. Craig. Introduction to robotics mechanics and control, 2nd Ed. Addison Wesley Publishing Company, 1989.

[25] John J. Craig. Adaptive control of mechanical Manipulators. Addison Wesley Publishing Company, 1988.

[26] Etienne Dombre, Wisama Khalil. Modélisation et commande des robots. Addition HRMES, Paris, 1988.

[27] Khalil W., Dombre E., « Modeling, Identification and Control of Robots », *Hermès Penton Science,* 2002.

[28] P. I. Corke. A Robotics Toolbox for MATLAB. *IEEE Robotics & Automation Magazine*, 3(1):24 – 32, 1996.

[29] R. P. Paul, Robot Manipulators (MIT Press, 1981).

[30] O.Khatib 1987 "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation." *IEEE J. Robotics and Automation* vol. RA-3, no. 1, pp 43-53.

[31] R. H. Middleton and G. C. Goodwin, "Adaptive Computed Torque Control for Rigid Link Manipulators" in Proceedings of the 22nd IEEE Conference on Decision and Control (1986), pp. 68-73.

[32] M. W. Spong and M. Vidyasagar, "Robust Linear Compensator Design for Nonlinear Robotic Control" IEEE Journal of Robotics and Automation 3, no. 1 (1987): pp 345-351.

[33] Mason M.T., "Compliance and Force Control for Computer Controlled Manipulators", *IEEE Trans. on Systems, Man and Cybernetics,* vol. 11(6), 1981, pp. 418-432.

[34] J. Nethery, and M. Spong, "Robotica: A Mathematica Package for Robot Analysis," IEEE Robotics & Automation Magazine v1, n1, pp13-20, 1996.

[35] E. G. Gilbert and I. J. Ha, "An Approach to Nonlinear Feedback Control with

applications to Robotics" in Proceedings of the 22nd IEEE Conference on Decision and Control (1983), pp. 134-138.