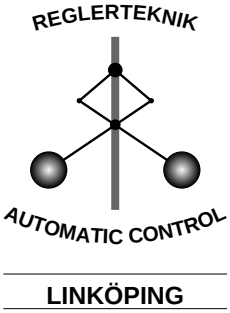
Modeling of industrial robots

Mikael Norrlöf

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
WWW: http://www.control.isy.liu.se
Email: mino@isy.liu.se

December 16, 1999



Report no.: LiTH-ISY-R-2208

Technical reports from the Automatic Control group in Linköping are available by anonymous ftp at the address ftp.control.isy.liu.se. This report is contained in the compressed postscript file 2208.ps.gz.

Contents

Modeling of industrial robots*

Mikael Norrlöf
Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
Email: mino@isy.liu.se

Abstract

A general theory for modeling of industrial robots is presented and, based on the theory, some algorithms for the actual modeling process are derived and described. As an example of an application of the theory an industrial manipulator, an IRB1400 from ABB Robotics Products, is used. A kinematic model and a dynamical model for the first three joints of the IRB1400 manipulator is found. One part of the report is also devoted to flexible manipulators, especially manipulators with joint flexibilities. Finally, an identification experiment using data from the IRB1400 is discussed.

1 Introduction

In this report the robot modeling problem will be discussed. The robot that will be used as an example throughout this report is depicted in Figure 1.1. This is a classical industrial manipulator, an ABB IRB 1400. Some of the important

Figure 1.1: The ABB IRB 1400 manipulator.

*This work was supported by ABB Robotics within ISIS at Linköping University.

properties of this type of robot will be examined in this document. We will start by looking at the kinematics, i.e. the geometrical description of the manipulator. Many survey articles and books have been written on the kinematics, dynamics, and control of robots, e.g. [2, 10, 11].

2 Kinematics

The *kinematics* of a robot refers to the geometric relationship between the motion of the robot in joint space and the motion of the tool frame relative to the base frame of the robot. In this section the concept of forward kinematics will be described and the procedure for building the forward kinematic model of an ABB IRB1400 will be described. The actual calculations will be done using the Robotics Toolbox [1] in MATLAB™.

2.1 Some facts about rigid motions and homogeneous transformations

Before introducing the actual kinematic model for the IRB1400 some of the mathematical tools used in the modeling will be presented. What we want to achieve is to describe the relation between different coordinate systems rotated and translated in a three-dimensional space. To achieve this we will use the notion of homogenous transformations.

2.1.1 Homogenous transformations

Consider the two coordinate systems depicted in Figure 2.1. The coordinate

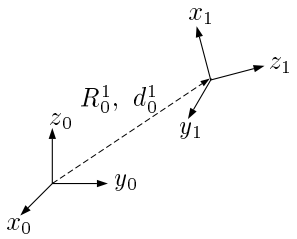


Figure 2.1: Translated and rotated frame.

system $o_1x_1y_1z_1$ is obtained from $o_0x_0y_0z_0$ by a translation and rotation described by the rotation matrix R_0^1 and the translation $d_0^1 \in \mathbb{R}^3$. This kind of transformation is the most general that we can consider in a 3 dimensional space and it is called *rigid motion*. It can be expressed as

$$p_0 = R_0^1 p_1 + d_0^1 \tag{2.1}$$

where p_0 is a coordinate in $o_0x_0y_0z_0$ and R_0^1 is the orientation matrix describing the orientation of $o_0x_0y_0z_0$ with respect to $o_1x_1y_1z_1$, and d_0^1 is the translation of frame 1 relative to frame 0. An important property of an orientation matrix is that $\det R = +1$ and the set of all 3×3 matrices having this property are generally referred to as $SO(3)$ where SO stands for *Special Orthogonal group*, here of order 3. By introducing *homogenous transformations* (2.1) can be expressed in a short form

$$\begin{bmatrix} p_0 \\ 1 \end{bmatrix} = H \begin{bmatrix} p_1 \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} R_0^1 & d_0^1 \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

where R_0^1 is the rotation matrix and d_0^1 gives the translation exactly as in (2.1).

The condition for the orientation matrix to be in $SO(3)$ gives actually a strong limitation of the elements in the matrix. Even though the orientation matrix has 9 elements it is, in fact, possible to express all matrices in $SO(3)$ using only 3 elements. One possible realization of the orientation matrix using only 3 parameters is the Euler angle representation. The Euler angles specify the orientation of a coordinate system, $o_0x_0y_0z_0$, relative to another coordinate system, $o_1x_1y_1z_1$, by using three angles, (α, β, γ). The orientation of $o_1x_1y_1z_1$ relative to $o_0x_0y_0z_0$ is found by first rotate about the z_0 axis by the angle β. Next rotate about the current y axis by the angle α. Finally rotate about the current z (actually z_1) axis by the angle γ. Another way to express the orientation is using a quaternion. This will be described in the next section.

2.1.2 Using quaternions for representing orientation

A quaternion consists of four elements, $q_1, \ldots, q_4$ and it is actually an extension to complex numbers. A quaternion can be written as a mathematical object as,

$$q = q_1 + iq_2 + jq_3 + kq_4 \tag{2.3}$$

where $q_n \in \mathbb{R}$, $n \in [1, 4]$, and i, j, k are mutually orthogonal imaginary units having the property,

$$i^2 = j^2 = k^2 = ijk = -1, \quad ij = k, \quad jk = i, \quad ki = j \tag{2.4}$$

A quaternion can also, as suggested in [4], be written as a tuple, $q = [s, < x, y, z >] = [s, v]$. Where v is a vector with three elements. Addition of two quaternions can be defined as,

$$q = q_1 + q_2 = [s_1 + s_2, v_1 + v_2] \tag{2.5}$$

and multiplication is defined as,

$$[s_1, v_1] * [s_2, v_2] = [s_1 s_2 - v_1 \cdot v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2] \tag{2.6}$$

The cross product involved in the calculations implies that the multiplication operation in general is not commutative. It is also possible to define the conjugate $\bar{q}$, the norm $\|q\|$, and the inverse q^{-1} according to,

$$\bar{q} = [s, -v] \tag{2.7}$$

$$\|q\| = \sqrt{q * \bar{q}} = \sqrt{s^2 + |v|^2} \tag{2.8}$$

$$q^{-1} = \frac{1}{q} * \frac{\bar{q}}{\bar{q}} = \frac{\bar{q}}{\|q\|} \tag{2.9}$$

A quaternion with norm 1 is called a *unit quaternion* and for a unit quaternion it is true that $q^{-1} = \bar{q}$. As noted in [4] quaternions can also be expressed in trigonometric or exponential form. If we have $q = [s, v]$ given as an arbitrary quaternion and let

$$r = \|q\|, \quad \phi = \arctan\frac{|v|}{s}, \quad k = \frac{v}{|v|} \tag{2.10}$$

The quaternion q can now be written as

$$q = r(cos(\phi) + ksin(\phi)) \tag{2.11}$$

It is also possible to extend Euler's formula for complex numbers and write,

$$q = [s, v] = r(cos(\phi) + ksin(\phi)) = re^{k\phi} \tag{2.12}$$

where r, k, and ϕ are given by (2.10).

Now we can show how quaternions can be used to represent an arbitrary rotation in a very compact manner. Assume that two vectors a and b are separated by an angle θ. This means that there is a homogenous transform $M = Rot_{k,\theta}$ that makes

$$b = Ma \tag{2.13}$$

and k is the axis around which the rotation is done. To express (2.13) using quaternions we simply define $q = [cos(\theta/2), ksin(\theta/2)]$ and then write

$$b = q * a * \bar{q} \tag{2.14}$$

which is equivalent to (2.13).

To give a sloppy proof for (2.14), it is possible to establish the relation between quaternions and the corresponding rotational matrix and vice versa, see also e.g. [5]. Given a unit quaternion $q = [cos(\theta/2), ksin(\theta/2)] = [s, < x, y, z >]$ the rotational matrix R can be found as,

$$R = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz+2sy \\ 2xy+2sz & 1-2x^2-2z^2 & 2yz-2sx \\ 2xz-2sy & 2yz+2sx & 1-2x^2-2y^2 \end{bmatrix} \tag{2.15}$$

and, for example, if k is chosen as $k = [1\ 0\ 0]$ then

$$Rot_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1-2\sin^2(\theta/2) & -2\cos(\theta/2)\sin(\theta/2) \\ 0 & 2\cos(\theta/2)\sin(\theta/2) & 1-2\sin^2(\theta/2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

which is a rotational matrix for rotating an angle θ around the x-axis. To go the other way around, i.e. from a rotational matrix to a quaternion, (2.15) can be used,

$$s = \frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1} = \frac{1}{2}\sqrt{\mathrm{tr}\,R+1} \tag{2.16}$$

where tr is the trace function and the fact that the quaternion is unit is used. To find x, y, and z we see that we can find a vector parallel to k by the following calculations,

$$\begin{aligned} k'_x &= 4sx = r_{32} - r_{23} \\ k'_y &= 4sy = r_{13} - r_{31} \\ k'_z &= 4sz = r_{21} - r_{12} \end{aligned} \tag{2.17}$$

The vector k_x can not be calculated only using this expression because when θ approaches 0 or π the right hand side of (2.17) becomes 0. By considering the diagonal of the matrix R, however, we can get an idea of the primary direction of the rotation axis. This gives three possibilities and here we will show one of these, namely when the x component is the largest among x, y, and z. The other two cases are similar. We have

$$r_{11} - r_{22} - r_{33} + 1 = 4x^2 \tag{2.18}$$

and a new vector k''_x, colinear with k'_x, can be found as

$$\begin{aligned} k''_x &= 4xx = r_{11} - r_{22} - r_{33} + 1 \\ k''_y &= 4xy = r_{21} + r_{12} \\ k''_z &= 4xz = r_{23} + r_{13} \end{aligned} \tag{2.19}$$

Now if $k'_x \geq 0$ then the x component is greater than zero, since s by definition is positive (cf. (2.16)). This also means that k'_x and k''_x are parallel and having the same direction. If $k'_x < 0$ we will have the opposite relation and k''_x points in the opposite direction. To combine the two vectors we can do as follows

$$\begin{aligned} k'''_x &= k'_x + \text{sign}(k'_x)k''_x \\ k'''_y &= k'_y + \text{sign}(k'_x)k''_y \\ k'''_z &= k'_z + \text{sign}(k'_x)k''_z \end{aligned} \tag{2.20}$$

The resulting quaternion should be a unit quaternion so we must have

$$|k|^2 = x^2 + y^2 + z^2 = 1 - s^2 \tag{2.21}$$

and this means that

$$k = \left(\frac{\sqrt{1 - s^2}}{|k'''|} \right) k''' \tag{2.22}$$

where $q = [s, k]$ is the quaternion that corresponds to the given rotational matrix R. In the Robotics toolbox for MATLAB™ the above transformations are implemented as two commands, `q2tr` to go from a quaternion to a homogenous transformation matrix, and `tr2q` to go from a homogenous transformation matrix to a quaternion.

More on quaternions and the computational aspects of quaternions can be found in, e.g., [4, 9, 5, 3].

2.2 Position kinematics

We are now going to establish the relation between the motion of the robot in joint space and the motion of the tool frame relative to another frame, for example the base frame. We will use the Denavit-Hartenberg representation to build the forward kinematic model. In the second part of this section the inverse kinematics problem will be considered.

2.2.1 Forward kinematics

When modeling the forward kinematics of a robot manipulator the idea of kinematic chains is used. This means that the manipulator is considered to consist of a set of rigid links connected together in a chain. Suppose the robot has $n+1$ links connected with n joints. The links are numbered from 0 to n, starting with 0 for the base of the robot. Let the joints be numbered 1 to n where joint i connects link $i-1$ and i. The i-th joint variable is called ϕ_i and denotes an angle for a revolute joint and a displacement for a prismatic joint.

Now attach a coordinate system to all links in such a way that the positions of all points on link i are constant when expressed in coordinate system $o_i x_i y_i z_i$ whatever motion the robot performs. The Denavit-Hartenberg, or D-H convention is a systematic method to choose these coordinate systems. Let A_i denote the homogenous matrix that transforms the coordinates of a point from frame i to $i-1$. This transformation is not constant but varies as a function of the corresponding joint variable ϕ_i. In the D-H convention the transformation matrix, A_i, is chosen as a combination of four basic transformations

$$A_i = Rot_{z,\theta_i}\ Trans_{z,d_i}\ Trans_{x,a_i}\ Rot_{x,\alpha_i} \tag{2.23}$$

The variables θ_i, d_i, a_i, and α_i are parameters of link i and joint i. When considering the case where each joint only has one degree of freedom, three of the four parameters are constant. The fourth is a variable, this means θ_i for a revolute and d_i for a prismatic joint. The expression in (2.23) can also be written as a multiplication of homogenous transformation matrices,

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c_{(\cdot)}$ and $s_{(\cdot)}$ represents $\cos(\cdot)$ and $\sin(\cdot)$ respectively.

The parameters in the transform are generally given the following names: a_i is called the *length*, α_i is called the *twist*, d_i is called the *offset*, and θ_i the *angle*. In Figure 2.2 the positive sense for the angles α_i and θ_i are shown.

We are now going to illustrate how the Denavit-Hartenberg representation can be applied to an industrial robot manipulator. First a slightly modified version of an algorithm from [11] for deriving the forward kinematics is presented:

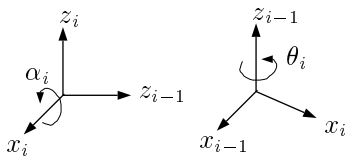


Figure 2.2: Positive direction for α_i and θ_i.

Algorithm 1 (D-H representation of forward kinematics) *The following steps give a systematic approach to building the forward kinematic model.*

1. *Locate and label the joint axes $z_0, \ldots, z_{n-1}$ in such a way that the positive direction for joint motion is correct with respect to the definition in Figure 2.2.*

2. *Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-hand coordinate system, $o_0x_0y_0z_0$.*

   Repeat step 3 to 5 for $i = 1, \ldots, n-1$

3. *Place the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i at joint i.*

4. *Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the plane spanned by z_{i-1} and z_i if z_{i-1} and z_i intersect.*

5. *Establish y_i to complete a right hand coordinate system.*

6. *Establish the end-effector frame $o_nx_ny_nz_n$.*

7. *Create the table of link parameters α_i, a_i, θ_i, d_i*
   - α_i *the angle between z_{i-1} and z_i measured as shown in Figure 2.2.*
   - a_i *distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i.*
   - θ_i *the angle between x_{i-1} and x_i measured as shown in Figure 2.2. If the joint is revolute this parameter is variable.*
   - d_i *distance along z_{i-1} from o_{i-1} to the intersection of the x_i axis and the z_{i-1} axes. This parameter is variable if the joint is prismatic.*

Algorithm 1 can now be applied to the industrial robot, IRB1400. The robot in Figure 1.1 can be considered as an open mechanical chain. Figure 2.3 shows a blueprint of the IRB1400 robot including the measures of the length of the different links.

Following the algorithm we start by locating and labeling the joint axes z_0 to z_5. Since all the joints are revolute we simply set the joint axis, z_i, such that it corresponds to the axis of rotation of joint $i+1$. This is shown in Figure

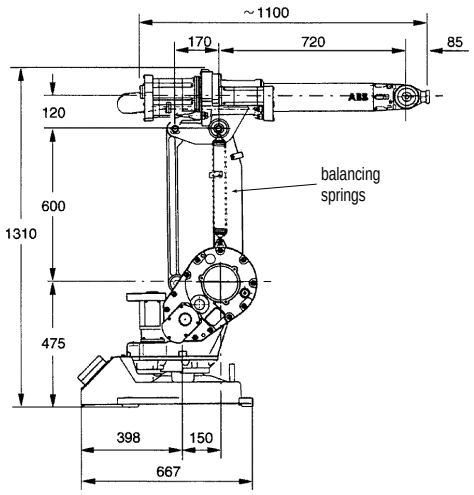


Figure 2.3: A blueprint of the ABB IRB1400 robot.

2.4. The axes z_1, z_2, and z_4 are all parallel to y_0 while z_3 and z_5 are parallel to x_0. The step 2 in the algorithm is already done in Figure 2.4 where the robot

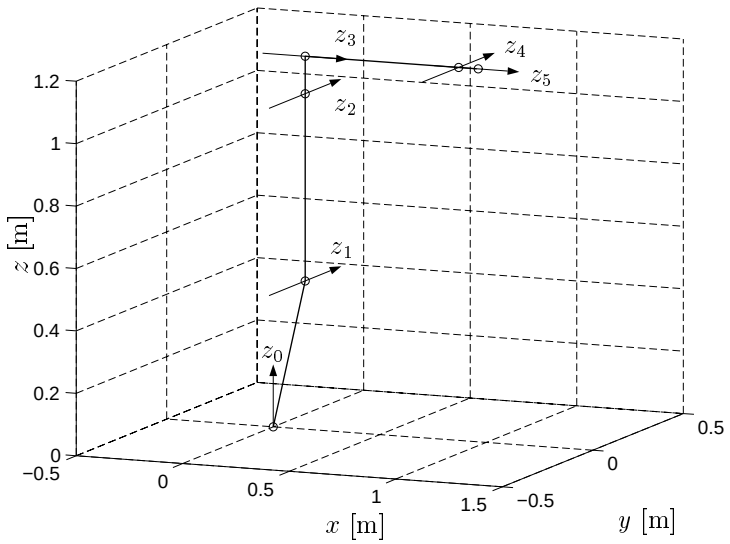


Figure 2.4: The result from step 1 in Algorithm 1.

is depicted in the base coordinate system, $o_0x_0y_0z_0$. Note that the robot is in the plane spanned by the y_0 and z_0 axes. Now steps 3, 4, and 5 can be done to establish frame 1 to 5. By following the algorithm we see that one possible result of step 3, placing the origin for the different coordinate systems, is simply to place o_i at the joint $i+1$. The joints 1 to 6 are all marked with small circles

in Figure 2.4. This idea of choosing the origin works for all o_i except o_5. For this joint the origin should be chosen as the point where z_4 and z_5 intersects, i.e. o_4. Step 4 can also be done quite easily but it is important to stress that there is no unique solution. In Figure 2.5 the result of applying step 3 to 5 is displayed. Here the direction of x_i has been chosen in such a way that the parameter a_i is positive for all i.

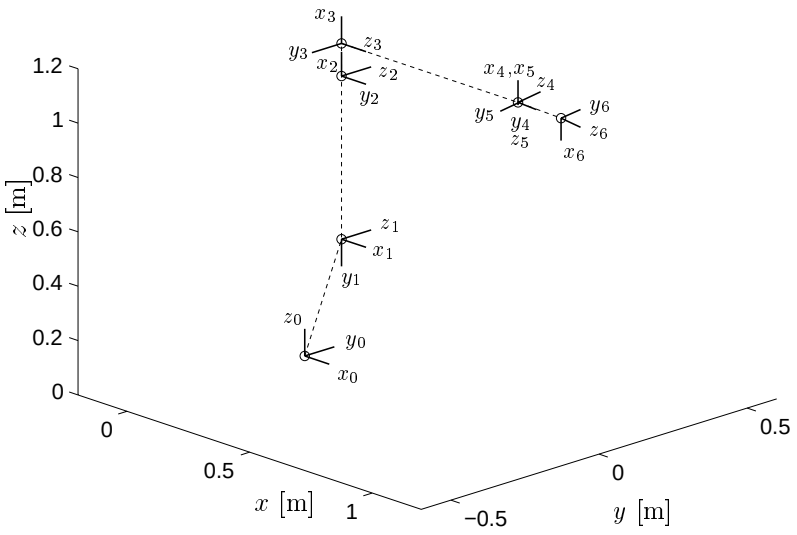


Figure 2.5: The result from step 3 to 5 in Algorithm 1.

The end-effector frame, $o_6x_6y_6z_6$, shown in Figure 2.5, is for the current application chosen to be the rotated (around z_5) and translated (along z_5) version of frame 5.

It is now possible to create the table of joint/link parameters for the IRB1400 according to Table 2.1.

Joint/Link i	α_i [rad]	a_i [m]	θ_i [rad]	d_i [m]
1	$-\pi/2$	0.15	0	0.475
2	0	0.6	$-\pi/2$	0
3	$-\pi/2$	0.12	0	0
4	$\pi/2$	0	0	0.72
5	$-\pi/2$	0	0	0
6	0	0	π	0.085

Table 2.1: The D-H link parameters for the ABB IRB1400

Using the following definition of the zero pose,

$$\theta^0 = \begin{bmatrix} 0 & -\pi/2 & 0 & 0 & 0 & \pi \end{bmatrix}^T \tag{2.24}$$

it is possible to calculate the position and orientation of the end-effector expressed in the base coordinate system. In MATLAB™ syntax with the Robotics Toolbox this is expressed as

```
 >> abb1400
 >> T = fkine(irb1400,qz)

T =

   -0.0000   -0.0000    1.0000    0.9550
   -0.0000    1.0000    0.0000    0.0000
   -1.0000   -0.0000   -0.0000    1.1950
         0         0         0    1.0000
```

The first command defines the kinematic model having the name `irb1400`. From the resulting matrix, T, we can see that the position of the end-effector relative to the base frame of the robot is 955 mm in the x_0 direction, 0 mm in the y_0 direction and 1195 mm in the z_0 direction. The orientation is given by the upper left 3×3 sub-matrix of T and this orientation can also be expressed as a quaternion using the following command in the Robotics Toolbox,

```
>> tr2q(T)

ans =

    0.7071   -0.0000    0.7071    0.0000
```

Both the x-y-z-coordinates and the quaternion corresponds exactly with the one presented on the teach pendant when the robot is in the joint position θ^0.

To complete the kinematic model of the IRB1400 one particularity in the design of the robot has to be examined. If we take a closer look at Figure 2.3 we can see that there is a mechanical link between motor 3 on the foot of the robot, to the actual link 3. This means that when motor 2 is moved and motor 3 is not moved the angle between the upper arm, i.e. $x_3 - axis$, and $z_0 - axis$ is constant. The coupling between joint two and joint three actually violate the kinematic chain assumption but since this coupling can be solved using an algebraic relation, it does not cause any problem.

We can now write down the following relation, describing how the D-H parameter θ can be calculated from the joint angles ϕ, specified for the ABB IRB1400.

$$\theta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \phi + \theta^0 \tag{2.25}$$

Using the commands `abb2dh` and `dh2abb` in our extension to the Robotics Toolbox it is possible to calculate the D-H joint angles given IRB1400 joint angles and vice versa.

2.2.2 Inverse kinematics

We are now going to consider the inverse problem, i.e. calculating the robot joint configuration from the position and orientation of the end-effector. This problem is in general much more difficult than the forward kinematics problem.

For a 6-DOF robot, as the IRB1400, having a spherical wrist it is possible to simplify the inverse kinematics problem by dividing it into an *inverse position kinematics* problem and an *inverse orientation kinematics* problem. Each of these problems having a much lower complexity than the original general 6-DOF inverse kinematics problem. We will now illustrate the method of dividing the inverse kinematic problem by considering the ABB IRB1400 robot.

First we can establish that the IRB1400 robot has a spherical wrist by noting that the z-axes of the last three joints (4, 5, and 6) intersect in one point, the origin of the coordinate systems $o_4x_4y_4z_4$ and $o_5x_5y_5z_5$. This point will from now on be referred to as the *wrist center point*, d_{wcp}, and the position of this point is independent of the orientation of the last three joints. Its position, therefore, depends only of the orientation of the first three joints (or the main axes).

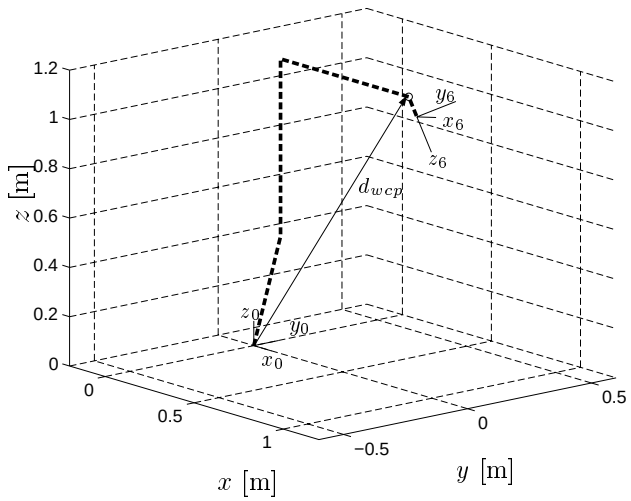


Figure 2.6: Illustration of the wrist center point of the ABB IRB1400 robot.

In Figure 2.6 the wrist center point is shown for the ABB IRB1400 and we can also see that the position of this point is not dependent of the joint position for joint 4 to 6. We can now consider the case when we have a position and orientation given for the coordinate system $o_6x_6y_6z_6$ as a homogenous transformation

relative the base frame, $o_0x_0y_0z_0$,

$$T_0^6 = \begin{bmatrix} R_0^6 & d_0^6 \\ 0 & 1 \end{bmatrix} \tag{2.26}$$

Using this homogenous transformation matrix it is possible to calculate the position of the wrist center point using the following relation,

$$d_0^6 = d_{wcp} + R_0^6 \begin{bmatrix} 0 \\ 0 \\ d_6 \end{bmatrix} = d_{wcp} + d_6 \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \tag{2.27}$$

where d_6 is the D-H parameter measuring the distance along z_5 from o_5 to o_6. From (2.27) the wrist center point can now readily be calculated,

$$d_{wcp} = d_0^6 - d_6 \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \tag{2.28}$$

Given the position of the wrist center point we can start calculating the position of joints 1, 2, and 3. Note however that the solution to this problem is not unique.

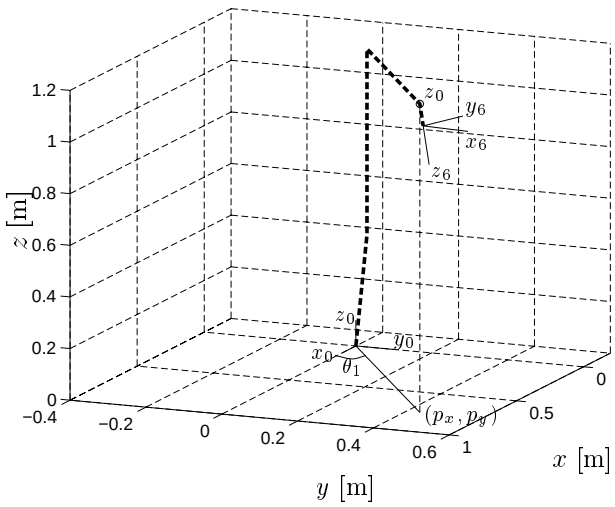


Figure 2.7: Calculation of joint 1 position based on projection.

In Figure 2.7 it is shown how the position of joint 1 can be calculated by using projection of the wrist center point in the x_0 - y_0 plane. This projection is very simple since it is just to remove the z coordinate from the vector d_{wcp}. The solution is found as

$$\theta_1 = \arctan_2 (p_y, p_x) \tag{2.29}$$

where the $\arctan_2$ function is the two argument arctangent function. In MATLAB™ the function is implemented as `atan2`. Note that this function is defined for all

$(p_x, p_y) \neq 0$. The configuration where $(p_x, p_y) = 0$ is called a kinematic singularity and it arises when the wrist center point intersects the z_0-axis.

Even though not always realizable, there is a second possible solution for the joint angle, namely $\theta_1 = \arctan_2(p_x, p_y) + \pi$. Obviously this will also lead to other solutions of the joint angles θ_2 and θ_3.

Now it is possible to find the solution for the joints θ_2 and θ_3. Consider the plane spanned by the z_0-axis and the projection of d_{wcp} in the x_0-y_0 plane, shown in Figure 2.7. First we should notice that since the joint axis of joint 2 has an offset relative to the base coordinate system it is easier to use d_1^{wcp} instead of d_{wcp}. The vector d_1^{wcp} can be calculated from d_{wcp} by

$$d_1^{wcp} = d_{wcp} - R_0^1 \begin{bmatrix} a_1 \\ 0 \\ d_1 \end{bmatrix} \tag{2.30}$$

When we have calculated the new vector d_1^{wcp} it is possible to consider the problem of finding the angles for joints 2 and 3. In Figure 2.8 an equivalent 2-DOF robot is shown. Using the law of cosines we can state that,

$$\frac{a_2^2 + b_3^2 - c^2}{2a_2b_3} = \cos(\pi - \tilde{\theta}_3) = -\cos(\tilde{\theta}_3) \tag{2.31}$$

Now let

$$D = \frac{c^2 - a_2^2 - b_3^2}{2a_2b_3} \tag{2.32}$$

We get the following expression for $\tilde{\theta}_3$,

$$\tilde{\theta}_3 = \pm \arccos(D) \tag{2.33}$$

where the $\pm$ comes from the fact that we have two solutions, elbow up or elbow down. In Figure 2.8 the elbow up configuration is shown. It is now possible to write down an equation for $\tilde{\theta}_2$

$$\tilde{\theta}_2 = \arctan\left(\frac{p_x}{p_y}\right) - \arctan\left(\frac{b_3 \sin(\tilde{\theta}_3)}{a_2 + b_3 cos(\tilde{\theta}_3)}\right) \tag{2.34}$$

Finally we have to calculate the joint angles corresponding to the D-H parameterization. First we have to note that the two solutions for θ_1 give rise to two different kinds of solutions because of the offset from joint 3 to joint 4. The first solution is given by

$$\theta_1 = \arctan_2(p_y, p_x) \tag{2.35}$$

$$\theta_2 = \tilde{\theta}_2 - \frac{\pi}{2} \tag{2.36}$$

$$\theta_3 = \tilde{\theta}_3 - \arctan\left(\frac{d_4}{a_3}\right) \tag{2.37}$$

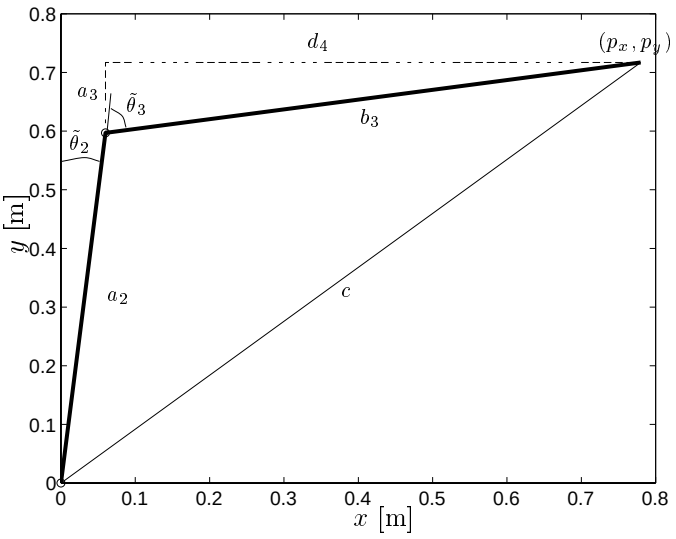


Figure 2.8: The equivalence between the IRB1400 joints 2 and 3, and a simple 2-DOF robot arm.

and the second solution is

$$\theta_1 = \arctan_2(-p_y, -p_x) \tag{2.38}$$

$$\theta_2 = -\tilde{\theta}_2 - \frac{\pi}{2} \tag{2.39}$$

$$\theta_3 = -\tilde{\theta}_3 - \arctan\left(\frac{d_4}{a_3}\right) \tag{2.40}$$

Note that both solutions give rise to two solutions for θ_2 and θ_3 given a value of θ_1. This means that in the general case there is a total of 4 solutions. Because of the limited range of the robot it is of course possible that some coordinates will give only 2 or even no solutions. The inverse kinematics as it is described here is implemented in the MATLAB™ function `ikin1`. This function calculates all the possible solutions for joints θ_1, θ_2, and θ_3 for the IRB1400 to reach a given p_{wcp}.

Example 1 (The ikin1 command) *The* `ikin1` *command in MATLAB™ solves the inverse position kinematics problem.*

```
>> pwcp

pwcp =

    1.0877
    0.1091
    0.8235

>> theta = ikin1(pwcp')
```

```
theta =

    0.1000   -1.1567    0.0297
    0.1000    0.4488   -2.8410
   -3.0416   -2.6007   -1.8924
   -3.0416   -3.1359   -0.9189
```

We can see that there are four solutions for the given p_{wcp}. One row in the `theta` *matrix corresponds to a solution of θ_1, θ_2, and θ_3 respectively.*

We are now left with the problem of calculating the joint position for the last three joints from the orientation matrix R_0^6. First we can calculate the rotation matrix R_0^3 as

$$R_0^3 = Rot_{z,\theta_1} Rot_{x,\alpha_1} Rot_{z,\theta_2} Rot_{x,\alpha_2} Rot_{z,\theta_3} Rot_{x,\alpha_3} \tag{2.41}$$

which can be expressed in matrix form as

$$R_0^3 = \begin{bmatrix} c_1 & 0 & -s_1 \\ s_1 & 0 & c_1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & 0 & -s_3 \\ s_3 & 0 & c_3 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} c_1 c_{23} & s_1 & -c_1 s_{23} \\ s_1 c_{23} & -c_1 & -s_1 s_{23} \\ -s_{23} & 0 & -c_{23} \end{bmatrix} \tag{2.42}$$

where c_i is $\cos\theta_i$, s_i is $\sin\theta_i$, c_{ij} is $\cos(\theta_i + \theta_j)$, and s_{ij} is $\sin(\theta_i + \theta_j)$. We know that $R_0^6 = R_0^3 R_3^6$ and if we want to solve for the last three joint angles we should solve the system of equations,

$$R_3^6 = (R_0^3)^T R_0^6 \tag{2.43}$$

The rotation matrix R_3^6 can be calculated in the same way as in (2.41) using the definitions of the D-H parameters in Table 2.1. The resulting matrix becomes,

$$R_3^6 = \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & -c_4 s_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & -s_4 s_5 \\ s_5 c_6 & -s_5 s_6 & c_5 \end{bmatrix} \tag{2.44}$$

If we let

$$R_0^6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.45}$$

we get $U = (R_0^3)^T R_0^6$ to be

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} =$$

$$\begin{bmatrix} c_1 c_{23} r_{11} + s_1 c_{23} r_{21} - s_{23} r_{31} & c_1 c_{23} r_{12} + s_1 c_{23} r_{22} - s_{23} r_{32} & c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33} \\ -c_1 r_{21} + s_1 r_{11} & -c_1 r_{22} + s_1 r_{12} & -c_1 r_{23} + s_1 r_{13} \\ -c_{23} r_{31} - c_1 s_{23} r_{11} - s_1 s_{23} r_{21} & -c_{23} r_{32} - c_1 s_{23} r_{12} - s_1 s_{23} r_{22} & -c_{23} r_{33} - c_1 s_{23} r_{13} - s_1 s_{23} r_{23} \end{bmatrix} \tag{2.46}$$

and we can see that it is possible to solve for θ_5 by using

$$\theta_5 = \pm \arccos\big(- c_{23}r_{33} - c_1 s_{23} r_{13} - s_1 s_{23} r_{23}\big) = \pm \arccos(u_{33}) \tag{2.47}$$

When c_5 does not equal ± 1 it is also possible to calculate θ_4 from

$$\begin{aligned}\theta_4 &= \arctan_2\big(- \text{sign}(\theta_5)(-c_1 r_{23} + s_1 r_{13}), -\text{sign}(\theta_5)(c_1 c_{23} r_{11} + s_1 c_{23} r_{21} - s_{23} r_{31})\big) \\ &= \arctan_2(-\text{sign}(\theta_5) u_{23}, -\text{sign}(\theta_5) u_{13})\end{aligned} \tag{2.48}$$

and finally θ_6 from

$$\begin{aligned}\theta_6 = \arctan_2\big(&- \text{sign}(\theta_5)(-c_{23}r_{32} - c_1 s_{23} r_{12} - s_1 s_{23} r_{22}), \\ &\text{sign}(\theta_5)(-c_{23} r_{31} - c_1 s_{23} r_{11} - s_1 s_{23} r_{21})\big) \\ &= \arctan_2(-\text{sign}(\theta_5) u_{32}, \text{sign}(\theta_5) u_{31})\end{aligned} \tag{2.49}$$

When $c_5 = \pm 1$ it is not possible to calculate uniquely the values of θ_4 and θ_6. In this case we will only be able to calculate $\theta_4 + \theta_6$ or $\theta_4 - \theta_6$. This situation occurs when the z-axes of joints 4 and 6 are parallel. From, for example element $(1,1)$ of R_3^6 and u_{11}, it follows that

$$\begin{aligned}\theta_4 + \text{sign}(c_5)\theta_6 &= \arccos\big(\text{sign}(c_5)(c_1 c_{23} r_{11} + s_1 c_{23} r_{21} - s_{23} r_{31})\big) \\ &= \arccos(\text{sign}(c_5) u_{11})\end{aligned} \tag{2.50}$$

One possible solution in this situation is of course to specify one of the two joint positions and solve for the other algebraically. The algorithm to find the joint angles θ_4, θ_5, and θ_6 is implemented in a MATLAB™-function, `ikin2`.

Example 2 (The ikin2 command) *The* `ikin2` *command in MATLAB™ solves the inverse rotation kinematic problem for a robot with a spherical wrist.*

```
>> R  % The rotation relative to the base frame

R =

   -0.0003   -0.1983    0.9801
    0.0007    0.9801    0.1983
   -1.0000    0.0007   -0.0001

>> R03 % The orientation of coordinate system 3 relative the base-frame

R03 =

    0.4272    0.0998    0.8986
    0.0429   -0.9950    0.0902
    0.9031   -0.0000   -0.4294

>> R36 = R03'*R; % Calculation of the wrist orientation matrix
>> [ths, s] = ikin2(R36)
```

```
ths =

    2.9128    0.4540    0.2070
   -0.2288   -0.4540   -2.9346


s =

     0
     0
```

The command `ikin2` *calculates the solution for joints* θ_4, θ_5, *and* θ_6 *respectively. As we can see from above there are two solutions for the given configuration of joints 1 to 3. The second returned value,* s, *takes the values* 0, $+1$, *or* -1. *The value 0 means that the wrist is not in a singular point,* $+1$ *means that only* $\theta_4 + \theta_6$ *can be calculated and* -1 *means that only* $\theta_4 - \theta_6$ *can be calculated.*

This concludes the part on position kinematics, next we will take a closer look at the velocity kinematics of the ABB IRB1400 robot.

2.3 Velocity kinematics

We have now studied the position kinematics and the rotational kinematics. To be able to produce smooth trajectories for the robot we will also need to have a relation between the joint velocities and the end-effector velocity in a Cartesian space. The relation from joint space of the robot to the Cartesian position is given by the forward kinematics and now we seek the derivative or the Jacobian of this relation. It can be expressed as

$$\begin{aligned} v_0^n &= \dot{d}_0^n = J_v \dot{\theta} \\ \omega_0^n &= J_\omega \dot{\theta} \end{aligned} \tag{2.51}$$

and we have to find the matrices J_v and J_ω.

2.3.1 Angular velocity

It is a fact that angular velocities [8, 11] can be added if they are expressed in the same coordinate system. The local description of the angular velocity for each joint is that

$$\omega_{i-1}^i = \dot{\theta}_i z_{i-1} \tag{2.52}$$

and if the i-th joint is prismatic then $\omega_{i-1}^i = 0$. The vector z_{i-1} is $R_0^{i-1}k$ where $k = [0\ 0\ 1]^T$. This means that we can express the angular velocity of the end-effector as

$$\omega_0^n = \sum_{i=1}^{n} \rho_i \dot{\theta}_i z_{i-1} \tag{2.53}$$

where ρ_i is 1 for a revolute joint and 0 for a prismatic joint. This means that the J_ω can be written as

$$J_\omega = [\rho_1 z_0 \ \dots \ \rho_n z_{n-1}] \tag{2.54}$$

This concludes the part on angular velocity, next we will consider the linear velocity, i.e. $\dot{d}_0^n$.

2.3.2 Linear velocity

The linear velocity of the end-effector can be calculated directly from $\dot{d}_0^n$ by using the chain rule. We get

$$\dot{d}_0^n = \sum_{i=1}^{n} \frac{\partial d_0^n}{\partial \theta_i} \dot{\theta}_i \tag{2.55}$$

and we notice that the i-th column of J_v is $\frac{\partial d_0^n}{\partial \theta_i}$. This is exactly the velocity that we will have if we let $\dot{\theta}_i$ equals unity and all other joint velocities equal zero. As in the case of angular velocity we will here have two different cases, one when the joint is of prismatic type and one when the joint is of revolute type. Let us start with the prismatic case. For a prismatic joint we have that

$$d_{i-1}^i = d_i \bar{z} + R_{i-1}^i a_i \bar{x} \tag{2.56}$$

and when all the other joints are fixed we have that

$$\dot{d}_0^n = R_0^{i-1} \dot{d}_{i-1}^i = \dot{d}_i R_0^{i-1} \bar{z} = \dot{d}_i z_{i-1} \tag{2.57}$$

This implies that

$$\frac{\partial d_0^n}{\partial \theta_i} = z_{i-1} \tag{2.58}$$

for a prismatic joint. If we instead consider a revolute joint

$$d_0^n = d_0^{i-1} + R_0^{i-1} d_{i-1}^n \tag{2.59}$$

Now, since all joint angels are kept constant except θ_i we will have that d_0^{i-1} and R_0^{i-1} are constant and thus

$$\dot{d}_0^n = R_0^{i-1} \dot{d}_{i-1}^n \tag{2.60}$$

Since the joint is revolute it is also true that

$$\dot{d}_{i-1}^n = \dot{\theta}_i \bar{z} \times d_{i-1}^n \tag{2.61}$$

and therefore

$$\dot{d}_0^n = R_0^{i-1} \dot{\theta}_i \bar{z} \times d_{i-1}^n = \dot{\theta}_i R_0^{i-1} \bar{z} \times R_0^{i-1} d_{i-1}^n = \dot{\theta}_i z_{i-1} \times (d_0^n - d_0^{i-1}) \tag{2.62}$$

which means that

$$\frac{\partial d_0^n}{\partial \theta_i} = z_{i-1} \times (d_0^n - d_0^{i-1}) \tag{2.63}$$

Now we can conclude that the Jacobian J_v is given as

$$J_v = [J_{v1}\ J_{v2}\ \ldots\ J_{vn}] \tag{2.64}$$

where J_{vi} is given by (2.63) if the i-th joint is revolute or (2.58) if it is a prismatic joint.

Given the forward kinematics it is easy to calculate the Jacobian. We can see that the only quantities we need are the unit vectors z_i and the vectors $d_0^1\ \ldots\ d_0^n$. When we have calculated the homogenous transform T_0^i as

$$T_0^i = \begin{bmatrix} x_i & y_i & z_i & d_0^i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.65}$$

where $x_i,\ y_i,\ z_i,\ d_0^i \in \mathbb{R}^3$. We see that it is possible to get z_i as the first three elements in the third column and d_0^i as the first three elements in the fourth column.

It is worth noting that the above presented procedure can be used to calculate the speed of any point of the manipulator. Simply change the position d_0^n to the position of which the speed should be calculated and involve just the joints that this position depends on. For example, the speed of a position on link 3 depends only of the speed of joint 1, 2, and 3.

2.3.3 Example IRB1400

We will now show the calculations of end-effector speed for the ABB IRB1400 manipulator. The D-H representation will be used according to Table 2.1. The calculation is done in MATLAB™ using the Robotics Toolbox. Actually the same calculations can be done also in the toolbox using the command `jacob0`. The following lines of code solves the problem of calculating the Jacobian for the IRB1400.

```
>> % Calculate the homogenous transforms for each link of the robot
>> q = qz;              % The zero position
>> n = irb1400.n;       % n = number of joints
>> L = irb1400.link;    % L is the link objects (in the robotics toolbox)
>> t = irb1400.base;    % t = the homogenous transformation matrix for
                     % the base frame
>> ts = [];
>> Jw = [];
>> t0n = fkine(irb1400,q); % The homogenous transformation matrix of the
                          % end-effector
>> d0n = t0n(1:3,4);       % The position of the end-effector
>> zim1 = [0 0 1]';
>> d0im1 = [0 0 0]';
>> Jv = [];
>> for j=1:n,              % For every joint do...
>>   t = t * L{j}(q(j));
```

```
>>   ts{j} = t;
>>   % the Jacobian for calculation of the angular velocity
>>  Jw = [Jw zim1];

>>   % the Jacobian for the linear velocity
>>   Jv = [Jv cross(zim1,d0n - d0im1)];
>>   zim1 = t(1:3,3);
>>   d0im1 = t(1:3,4);
>> end
>>
>> Jv

Jv =

   -0.0000    0.7200    0.1200    0.0000   -0.0000    0.0000
    0.9550    0.0000    0.0000   -0.0000    0.0000   -0.0000
         0   -0.8050   -0.8050    0.0000   -0.0850    0.0000

>> Jw

Jw =

         0         0         0    1.0000         0    1.0000
         0    1.0000    1.0000    0.0000    1.0000    0.0000
    1.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000

>> jacob0(irb1400,qz)

ans =

   -0.0000    0.7200    0.1200         0   -0.0000         0
    0.9550    0.0000    0.0000         0    0.0000         0
   -0.0000   -0.8050   -0.8050         0   -0.0850         0
   -0.0000         0         0    1.0000         0    1.0000
   -0.0000    1.0000    1.0000    0.0000    1.0000    0.0000
    1.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000
```

The above calculation follows the method described in the previous two sections. In order to test the given algorithm a comparison is done with the resulting Jacobian of the function included in the Robotics Toolbox. As can be seen above the two calculations give the same result for the IRB1400.

2.3.4 Inverse velocity and acceleration

The inverse velocity and acceleration can be calculated using the inverse position kinematics and the results from the forward velocity kinematics in the previous sections. Recall that

$$\dot{x} = J(\theta)\dot{\theta} \tag{2.66}$$

and differentiating gives the acceleration

$$\ddot{x} = J(\theta)\ddot{\theta} + \big(\frac{d}{dt}J(\theta)\big)\dot{\theta} \tag{2.67}$$

The inverse velocity kinematics is now simple to solve as

$$\dot{\theta} = J^{-1}(\theta)\dot{x} \tag{2.68}$$

under the assumption that $J(\theta)$ is non-singular. If the Jacobian is singular or if the robot has less than 6 DOF, it is only possible to solve (2.68) if $\dot{x}$ is in the range of $J(\theta)$, i.e. $rank J(\theta) = rank[J(\theta)\ \dot{x}]$.

The acceleration can be solved from (2.67) by the following relation

$$\ddot{\theta} = J^{-1}(\theta)(\ddot{x} - \big(\frac{d}{dt}J(\theta)\big)\dot{\theta}) \tag{2.69}$$

again under the assumption that $J(\theta)$ is non-singular. If the Jacobian is singular it is only possible to solve the equations if $\ddot{x} - \big(\frac{d}{dt}J(\theta)\big)\dot{\theta}$ is in the range of $J(\theta)$ which can be checked using a rank test as above.

3 Dynamics

Following the exposition in [11] with some clarifications on the derivation of the Euler-Lagrange equations from [6] the dynamic equations for a rigid body robot is derived in this chapter.

We will start by considering the derivation of the equations of motion using the *Euler-Lagrange equations*. Another way to find the dynamic equations of a robot manipulator that will be reviewed is the *Newton-Euler* formulation.

3.1 Euler-Lagrange formulation

The Euler-Lagrange equations is a tool in analytical mechanics that can be used to derive the equations of motion for a mechanical system. The mechanical system has to meet some assumptions; Firstly the constraint forces has to satisfy the principle of virtual work, secondly the system is subjected to holonomic constraints. The meaning of these assumptions will be explained in the next sections.

3.1.1 Background and definitions

The Euler-Lagrange equations can be derived in at least two distinct ways. Here the presentation will be focused on a method based on virtual displacements [11] but it is also possible to find the same relation using Hamilton's principle [6].

First we have to establish the limitation on the motion introduced by the fact that the different links in a robot are all fixed to each other. Consider

a simplified case where we have k particles, with the corresponding coordinates $r_1,\ r_2, \ldots,\ r_k$. The constraints that we assume to have on the k coordinates are called *holonomic* when they are equality constraints of the following form,

$$g_i(r_1, \ldots, r_k) = 0, \quad i = 1, \ldots, l \tag{3.1}$$

and *nonholonomic* if there are other types of constraints. An example where we have holonomic constraints is a simple system [11] consisting of two particles joined by a massless link of length l. The coordinates for the two particles must then satisfy,

$$\|r_1 - r_2\| = l \tag{3.2}$$

When an external force is applied on one of the particles the other particle will also experience a force (in the direction $r_2 - r_1$).

Now, if a system has l holonomic constraints it will have a reduced degree of freedom (with a factor l) compared to the unconstraint system. This implies, in turn, that the coordinates of the k particles can be expressed using fewer than k coordinates. The *generalized coordinates* $\phi_1, \ldots, \phi_n$ are introduced, having the property of being all *independent*. The coordinates of the particles can, of course, be calculated from the generalized coordinates as,

$$r_i = r_i(\phi_1, \ldots, \phi_n), \quad i = 1, \ldots, k \tag{3.3}$$

Important are also the *virtual displacements*, defined as any set $\delta r_1, \ldots, \delta r_k$ of infinitesimal displacements that are consistent with the constraints in (3.1). For the example with the two particles joined by the massless link we get

$$(r_1 + \delta r_1 - r_2 - \delta r_2)^T (r_1 + \delta r_1 - r_2 - \delta r_2) = l^2 \tag{3.4}$$

and if we expand (3.4), neglecting the quadratic terms in δr_1 and δr_2, and using (3.2), we get

$$(r_1 - r_2)^T (\delta r_1 - \delta r_2) = 0 \tag{3.5}$$

For this example any perturbation in the position of the two particles has to satisfy this equation in order to continue to satisfy (3.2).

The general set of virtual displacements can now be expressed using the generalized coordinates according to,

$$\delta r_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial \phi_j} \delta \phi_j, \quad i = 1, \ldots, k \tag{3.6}$$

where the virtual displacements $\delta\phi_1, \ldots, \delta\phi_n$ of the generalized coordinates are unconstraint since the generalized coordinates are independent. Now consider a constrained system in equilibrium. Equilibrium means that the sum of all forces acting on each particle equal zero. Since the net force on one particle r_i is zero ($F_i = 0$) also the virtual work done in the displacement δr_i is zero. The virtual

work equals $F_i \cdot \delta r_i$. From this it also follows that the sum of the work done by all the virtual displacements equals zero,

$$\sum_{i=1}^{k} F_i^T \delta r_i = 0 \tag{3.7}$$

The total force acting on a particle F_i can be divided into two components. The first being the externally applied force f_i and, second, the force caused by the constraints $f_i^{(c)}$, i.e.

$$F_i = f_i + f_i^{(c)} \tag{3.8}$$

Equation (3.7) therefore becomes

$$\sum_{i=1}^{k} f_i^T \delta r_i + \sum_{i=1}^{k} (f_i^{(c)})^T \delta r_i = 0 \tag{3.9}$$

We can now make the restriction that we will only consider systems where the total virtual work of the constraint forces is equal to zero [6]. This restriction hold for rigid bodies which will be the type of systems considered here. Now it follows that

$$\sum_{i=1}^{k} f_i^T \delta r_i = 0 \tag{3.10}$$

which is referred to as the *principle of virtual work*. We see that when this principle applies it is actually possible to analyze the dynamics of the system without having to evaluate the constraint forces $f_i^{(c)}$. Note that from (3.10) it is not possible to draw the conclusion that all f_i equals zero. This is not possible since the virtual displacements δr_i are not independent.

3.1.2 The Lagrange equation

First consider the following result developed by D'Alembert (but first thought of by James Bernoulli). The equation of motion

$$F_i = \dot{p}_i \tag{3.11}$$

can be rewritten as

$$F_i - \dot{p}_i = 0 \tag{3.12}$$

This means that the particles in the system will be in equilibrium when an additional force $-\dot{p}_i$ is introduced. We will now restrict ourselves to such systems where the virtual work of the constraint forces is zero. This means that, using (3.10) and (3.12), we arrive at

$$\sum_{i=1}^{k} (f_i - \dot{p}_i)^T \delta r_i = 0 \tag{3.13}$$

which is often referred to as the *D'Alembert's principle*. Since the different δr_i are not independent the result in (3.13) is not so useful. It would be more useful if we could transform the virtual displacement into the generalized virtual displacement as in (3.6). The virtual work done by the external forces can be written as

$$\sum_{i=1}^{k} f_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} f_i^T \frac{\partial r_i}{\partial \phi_j} \delta \phi_j = \sum_{j=1}^{n} \psi_j \delta \phi_j \tag{3.14}$$

where ψ_j is called the components of the *generalized force* and it is defined as

$$\psi_j = \sum_{i=1}^{k} f_i^T \frac{\partial r_i}{\partial \phi_j} \tag{3.15}$$

Note that ψ_j does not need to have the dimension of force, it is only necessary that the product $\psi_j \delta \phi_j$ has dimension of work.

We can now return to the second term in (3.13) and use that $p_i = m_i \dot{r}_i$,

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{i=1}^{k} m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial \phi_j} \delta \phi_j \tag{3.16}$$

and using the product rule of differentiation

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial \phi_j} = \sum_{i=1}^{k} \Big\{ \frac{d}{dt} \Big[m_i \dot{r}_i^T \frac{\partial r_i}{\partial \phi_j} \Big] - m_i \dot{r}_i^T \frac{d}{dt} \Big[\frac{\partial r_i}{\partial \phi_j} \Big] \Big\} \tag{3.17}$$

Now (3.3) can be differentiated according to

$$v_i = \dot{r}_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial \phi_j} \dot{\phi}_j, \quad \frac{\partial v_i}{\partial \dot{\phi}_j} = \frac{\partial r_i}{\partial \phi_j} \tag{3.18}$$

and

$$\frac{d}{dt} \Big[\frac{\partial r_i}{\partial \phi_j} \Big] = \sum_{l=1}^{n} \frac{\partial^2 r_i}{\partial \phi_j \partial \phi_l} \dot{\phi}_l = \frac{\partial v_i}{\partial \phi_j} \tag{3.19}$$

where the last equality follows from (3.18). Using (3.18) and (3.19) in (3.17) gives us

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial \phi_j} = \sum_{i=1}^{k} \Big\{ m_i \frac{d}{dt} \Big[v_i^T \frac{\partial v_i}{\partial \dot{\phi}_j} \Big] - m_i v_i^T \frac{\partial v_i}{\partial \phi_j} \Big\} \tag{3.20}$$

where also the fact that $v_i = \dot{r}_i$ has been used. Now the second term in (3.13) can be expanded into

$$\sum_{j=1}^{n} \Big\{ \frac{d}{dt} \Big(\frac{\partial}{\partial \dot{\phi}_j} \Big(\sum_{i=1}^{k} \frac{1}{2} m_i v_i^2 \Big) \Big) - \frac{\partial}{\partial \phi_j} \Big(\sum_{i=1}^{k} \frac{1}{2} m_i v_i^2 \Big) \Big\} \delta \phi_j \tag{3.21}$$

The term $\sum_i \frac{1}{2} m_i v_i^2$ is the system kinetic energy K and therefore

$$\sum_{j=1}^{n} \Big\{ \frac{d}{dt} \frac{\partial K}{\partial \dot{\phi}_j} - \frac{\partial K}{\partial \phi_j} - \psi_j \Big\} \delta\phi_j = 0 \tag{3.22}$$

From the definition of the virtual displacement $\delta\phi_j$ it now follows that each individual term in the sum above equals zero which means that

$$\frac{d}{dt} \frac{\partial K}{\partial \dot{\phi}_j} - \frac{\partial K}{\partial \phi_j j} = \psi_j, \quad j = 1, \ldots, n \tag{3.23}$$

Now, assume that ψ_j can be written as

$$\psi_j = -\frac{\partial V}{\partial \phi_j} + \tau_j \tag{3.24}$$

Using (3.24) in (3.23) gives the result

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\phi}_j} - \frac{\partial L}{\partial \phi_j} = \tau_j \tag{3.25}$$

where $L = K - V$ is called the *Lagrangian* and the function V is called the potential energy. Equation (3.25) is referred to as the *Lagrange equation* or the *Euler-Lagrange equation of motion*. In (3.25) the term τ_j is assumed to be an externally applied torque on the system, e.g. caused by a motor in a joint of a robot.

3.1.3 Special structure for the kinetic and potential energy in the robotics case

To be able to use the result in the previous section it is necessary to find expressions for the kinetic and potential energy. First suppose that we have a rigid object for which we want to find the kinetic energy. Assume that we attach a coordinate system rigidly to the center of mass of the object. The velocity of a point r on the object can now be expressed as

$$v = v_c + \omega \times r \tag{3.26}$$

Consider the definition of kinetic energy,

$$K = \frac{1}{2} \int_B v^T(x, y, z) v(x, y, z) dm \tag{3.27}$$

and the fact that the cross product can be written as a skew symmetric matrix multiplying a vector [11], $\omega \times r = S(\omega) r$, where the matrix $S(\omega)$ is given by

$$S(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ \omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.28}$$

Using (3.26), (3.27) and the above result on the cross product we get

$$K = \frac{1}{2} \int_B [v_c + S(\omega) r]^T [v_c + S(\omega) r] dm \tag{3.29}$$

The integral can now be expanded and the first term becomes

$$\frac{1}{2}\int_B v_c^T v_c dm = \frac{1}{2} m v_c^T v_c \tag{3.30}$$

This is just the kinetic energy of a particle with mass m located at the center of mass and moving with the velocity v_c. This part is referred to as the *translational* part of the kinetic energy.

The second term equals

$$\frac{1}{2}\int_B v_c^T S(\omega) r dm = \frac{1}{2} v_c^T S(\omega) \int_B r dm = 0 \tag{3.31}$$

The result in (3.31) follows from the fact that the center of gravity is chosen as the origin of our coordinate system. The third term becomes zero for the same reason. For the last term we need a result from linear algebra. For any two vectors a and b it is true that,

$$a^T b = \mathrm{tr}\, b a^T \tag{3.32}$$

The fourth term can now be written according to

$$\begin{aligned} &\frac{1}{2}\int_B r^T S^T(\omega) S(\omega) r dm = \frac{1}{2}\int_B \mathrm{tr}\left\{S(\omega) r r^T S^T(\omega)\right\} dm \\ &= \frac{1}{2}\mathrm{tr}\left\{S(\omega)\int_B r r^T dm S^T(\omega)\right\} \end{aligned} \tag{3.33}$$

where in the first equality (3.32) is used. Now, using the definition of $S(\omega)$ from (3.28), and doing the trace calculation, gives the fourth term of (3.29) as

$$\frac{1}{2}\omega^T I \omega \tag{3.34}$$

where I is the 3×3 *inertia matrix*, defined as

$$I = \begin{bmatrix} \int y^2 + z^2 dm & -\int xy dm & -\int xz dm \\ -\int xy dm & \int x^2 + z^2 dm & -\int yz dm \\ -\int xz dm & -\int yz dm & \int x^2 + y^2 dm \end{bmatrix} \tag{3.35}$$

The fourth term of (3.29) is referred to as the *rotational* part of the kinetic energy. The total kinetic energy can therefore be written as

$$K = \frac{1}{2} m v_c^T v_c + \frac{1}{2}\omega^T I \omega \tag{3.36}$$

Is it important to know in which coordinate system the velocities, angular velocities, etc., are calculated? For the first term in (3.36) it will not matter in which coordinate system the vector v_c is expressed since the length will not change when transforming the vector from one coordinate system to another. Actually, also the second term will be the same irrespective of in which coordinate system the vector ω and the inertia matrix is calculated. Note, however, that by using the coordinate frame fixed in the center of mass of the object the inertia matrix will be constant. This will in general not be true for an inertial frame.

It is now possible to formulate the kinetic energy for an n-link manipulator. Using the velocity kinematic results from Section 2.3 we have

$$v_{cj} = J_{v_{cj}}(\phi)\dot{\phi}, \quad \omega_j = (R_0^j(\phi))^T J_{\omega_j}(\phi)\dot{\phi} \tag{3.37}$$

where the matrix $(R_0^j)^T$ transforms the angular velocity from the inertial frame to the frame attached to the link j.

Now, assuming that the mass of link i is m_i and the inertia matrix, expressed in the frame fixed in the center of mass of the object (but parallel to frame i of the manipulator), equals I_i. Then the kinetic energy can be expressed as,

$$\begin{aligned} K &= \frac{1}{2}\dot{\phi}^T \sum_{j=1}^{n} \Big[m_j J_{v_{cj}}^T(\phi) J_{v_{cj}}(\phi) + J_{\omega_j}^T(\phi)(R_0^j(\phi))^T I_j J_{\omega_j}(\phi) R_0^j(\phi) J_{\omega_j}(\phi) \Big] \dot{\phi} \\ &= \frac{1}{2}\dot{\phi}^T D(\phi)\dot{\phi} \end{aligned} \tag{3.38}$$

where $D(\phi) > 0$, $D(\phi) = D^T(\phi)$ is referred to as the *inertia matrix*.

The potential energy term, when considering only rigid body dynamics, is only caused by gravity and can be expressed as

$$V = \int_B g^T r dm = g^T \int_B r dm = g^T r_c m \tag{3.39}$$

Here a frame different from the frame fixed in the center of mass is used. Note that the potential energy of the manipulator depends only on ϕ and not on $\dot{\phi}$, this will be used in the next section.

3.1.4 The equations of motion of a manipulator

We will now use the Euler-Lagrange equation of motion to derive the equations of motion of a rigid body model of a robotic manipulator. The assumptions that will follow throughout this sections is that the kinetic energy is a quadratic function in $\dot{\phi}$ according to (3.38) with $D(\phi)$ symmetric and positive definite for all $\phi \in \mathbb{R}^n$. Next the potential energy is assumed to be independent of $\dot{\phi}$ but dependent of ϕ.

The Lagrangian (3.25) for this kind of system can now be expressed as

$$L = K - V = \frac{1}{2}\sum_{i,j} d_{ij}(\phi)\dot{\phi}_i\dot{\phi}_j - V(\phi) \tag{3.40}$$

and

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}_p} &= \frac{d}{dt}\sum_j d_{pj}(\phi)\dot{\phi}_j = \sum_j d_{pj}(\phi)\ddot{\phi}_j + \sum_j \frac{d}{dt}\Big\{ d_{pj}(\phi) \Big\}\dot{\phi}_j \\ &= \sum_j d_{pj}(\phi)\ddot{\phi}_j + \sum_{i,j} \frac{\partial d_{pj}}{\partial \phi_i}\dot{\phi}_i\dot{\phi}_j \end{aligned} \tag{3.41}$$

We need also

$$\frac{\partial L}{\partial \phi_p} = \frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial \phi_p}\dot{\phi}_i\dot{\phi}_j - \frac{\partial V}{\partial \phi_p} \tag{3.42}$$

Now the Euler-Lagrange equations can be written down as

$$\sum_j d_{pj}(\phi)\ddot{\phi}_j + \sum_{i,j}\Big\{\frac{\partial d_{pj}}{\partial \phi_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial \phi_p}\Big\}\dot{\phi}_i\dot{\phi}_j - \frac{\partial V}{\partial \phi_p} = \tau_p, \quad p = 1, \ldots, n \tag{3.43}$$

On matrix form it is possible to write (3.43) as

$$D(\phi)\ddot{\phi} + C(\phi,\dot{\phi})\dot{\phi} + g(\phi) = \tau \tag{3.44}$$

where the elements of the matrix $C(\phi,\dot{\phi})$ are defined as

$$c_{pj} = \sum_{i=1}^{n}\frac{1}{2}\Big\{\frac{\partial d_{pj}}{\partial \phi_i} + \frac{\partial d_{pi}}{\partial \phi_j} - \frac{\partial d_{ij}}{\partial \phi_p}\Big\} \tag{3.45}$$

The terms

$$c_{ijp} = \frac{1}{2}\Big\{\frac{\partial d_{pj}}{\partial \phi_i} + \frac{\partial d_{pi}}{\partial \phi_j} - \frac{\partial d_{ij}}{\partial \phi_p}\Big\} \tag{3.46}$$

are called *Christoffel symbols*. Note that $c_{ijp} = c_{jip}$ which can be used to reduce the number of calculations. For a more thorough discussion of the C matrix see e.g. [11].

To conclude this part with the theoretical background for modeling of the dynamics of the manipulator we will give an algorithm that can be followed to state the equations of motion for a general mechanical system.

Algorithm 2 (Derivation of the equations of motion using the Euler-Lagrange method)

1. *Find an expression for the* **kinetic energy** K *of the different links on the manipulator. This involves finding forward position kinematics as well as the forward velocity kinematics, needed parameters are also the mass* m_i *and the inertia matrix* I_i *for each link. The expression for the kinematic energy becomes*

$$\begin{aligned} K &= \frac{1}{2}\sum_{j=1}^{k}\Big\{m_j v_{cj}^T v_{cj} + \omega_j^T I_j \omega_j\Big\} \\ &= \frac{1}{2}\dot{\phi}^T\sum_{i=1}^{n}\Big[m_i J_{v_{ci}}^T(\phi)J_{v_{ci}}(\phi) + J_{\omega_i}^T(\phi)(R_0^i(\phi))^T I_i J_{\omega_i}(\phi)R_0^i(\phi)J_{\omega_i}(\phi)\Big]\dot{\phi} \\ &= \frac{1}{2}\dot{\phi}^T D(\phi)\dot{\phi} \end{aligned}$$

   where the second equality can be used if the manipulator is of an open kinematic chain type. The result from this step is the matrix $D(\phi)$.

2. *Find an expression for the* **potential energy** V *of the manipulator.*

$$V = g^T \sum_{i=1}^{k} m_i r_{ci}$$

3. *Calculate the* **Christoffel symbols**,

$$c_{ijp} = \frac{1}{2}\left\{ \frac{\partial d_{pj}}{\partial \phi_i} + \frac{\partial d_{pi}}{\partial \phi_j} - \frac{\partial d_{ij}}{\partial \phi_p} \right\}$$

taking into account that $c_{ijp} = c_{jip}$. *Find the matrix* $C(\phi, \dot{\phi})$ *with the elements defined as*

$$c_{pj} = \sum_{i=1}^{n} c_{ijp}\dot{\phi}_i$$

4. *Derive the part of the dynamics that depend of the potential energy,*

$$g_p(\phi) = \frac{\partial V}{\partial \phi_p}, \quad p = 1, \ldots, n$$

and let $g(\phi) = [g_1, \ldots, g_n]^T$.

5. *Finally write down the equations of motion,*

$$D(\phi)\ddot{\phi} + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) = \tau$$

The algorithm will now be applied on the IRB1400.

3.1.5 Example, the main axes of the IRB1400

Using Algorithm 2 we first need to establish the kinetic energy for the different links of the manipulator. In fact we want to express the kinetic energy as a function of the generalized coordinates. In the following these coordinates will be referred to as θ for the vector of all the six coordinates and θ_i for the individual joint coordinates. We will now consider each link of the manipulator and calculate the kinetic energy as a function of the link parameters, i.e. perform step 1 in Algorithm 2.

Step 1: First we need to find the vectors that describe the center of mass for the different links as a function of the generalized coordinates. By differentiating these coordinates with respect to time the linear velocity of the center of mass is found. The center of mass of link 1 coincides with the axis of rotation and the velocity is therefore zero.

$$v_{c1} = 0 \tag{3.47}$$

We can now establish the vector describing link 1,

$$l_1 = Rot_{z,\theta_1} \begin{bmatrix} l_{1,x} \\ 0 \\ l_{1,z} \end{bmatrix} \tag{3.48}$$

where Rot_{z,θ_1} is a matrix that describe a rotation around the z-axis of θ_1. Note that here the fact that the y component of the vector is zero when $\theta_1 = 0$ has been used. Before going on we have to consider the very much simplified picture of the first three axis of the ABB IRB1400 manipulator depicted in Figure 3.1. Both the actuators for the manipulator links 2 and

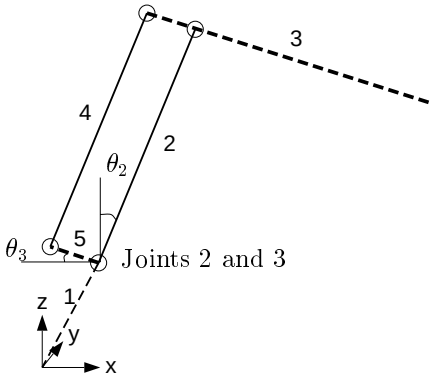


Figure 3.1: A simplified view of axes 1 to 3 of the IRB1400.

3 are positioned in the joint between link 1 and 2. With this mechanical structure a parallelogram is achieved in the mechanical chain created by links 2,3,4 and 5. This means that when actuating joint 2, link 5 will be still and the joint angle θ_3 will obviously be constant. Actuating joint 2 will therefore create a rotation of link 2 but the same rotation will also be done by link 4 since these two are always parallel. When actuating joint 3 we will instead have a rotation of link 3 and 5 while link 2 will be still and link 4 will do a linear motion but no rotation since link 2 is still.

Now we can go on with the modeling of the position of the center of mass for the links 2 to 5. For link 2 we have,

$$r_0^{c2} = l_1 + Rot_{z,\theta_1} Rot_{y,\theta_2} r_{c2} \tag{3.49}$$

and if we continue we have

$$r_0^{c3} = l_1 + Rot_{z,\theta_1} Rot_{y,\theta_2} \begin{bmatrix} 0 \\ 0 \\ l_{2,z} \end{bmatrix} + Rot_{z,\theta_1} Rot_{y,\theta_3} r_{c3} \tag{3.50}$$

$$r_0^{c4} = l_1 + Rot_{z,\theta_1} Rot_{y,\theta_3} \begin{bmatrix} l_{5,x} \\ 0 \\ 0 \end{bmatrix} Rot_{z,\theta_1} Rot_{y,\theta_2} r_{c4} \tag{3.51}$$

$$r_0^{c5} = l_1 + Rot_{z,\theta_1} Rot_{y,\theta_3} r_{c5} \tag{3.52}$$

Now we can calculate the respective velocity Jacobians for the different links according to,

$$J_{vci} = [\frac{\partial r_0^{ci}}{\partial \theta_1}, \frac{\partial r_0^{ci}}{\partial \theta_2}, \frac{\partial r_0^{ci}}{\partial \theta_3}] \tag{3.53}$$

The results are not presented here but can be reviewed in a Mathematica notebook distributed on request by the author. Now the inertia matrix $D(\theta)$ can be calculated,

$$D(\theta) = \sum_{i=2}^{5} m_i J_{vci}^T J_{vci} + \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 + I_4 & 0 \\ 0 & 0 & I_3 + I_5 \end{bmatrix} \tag{3.54}$$

where I_1 is the moment of inertia around the z-axis for link 1 and I_2 up to I_5 are the moments of inertia around the y-axis for links 2 to 5. The resulting matrix becomes

$$D(\theta) = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \tag{3.55}$$

where

$$\begin{aligned}
d_{11} &= I_1 + m_4\Big(l_{1,x} + l_{5,x}\cos(\theta_3) + r_{c4,z}\sin(\theta_2))^2 + \frac{1}{2}m_2(2l_{1,x}^2 + |r_{c2}|^2 \\
&\quad + 4l_{1,x}r_{c2,x}\cos(\theta_2) + (r_{c2,x}^2 - r_{c2,z}^2)\cos(2\theta_2) + 4l_{1,x}r_{c2,z}\sin(\theta_2) \\
&\quad + 2r_{c2,x}r_{c2,z}\sin(2\theta_2)\Big) + m_3\Big(l_{1,x} + r_{c3,x}\cos(\theta_3) + l_{2,z}\sin(\theta_2) \\
&\quad + r_{c3,z}\sin(\theta_3)\Big)^2 + \frac{1}{2}m_5\Big(2l_{1,x}^2 + |r_{c5}|^2 \\
&\quad + 4l_{1,x}r_{c5,x}\cos(\theta_3) + (r_{c5,x}^2 - r_{c5,z}^2)\cos(2\theta_2) \\
&\quad + 4l_{1,x}r_{c5,z}\sin(\theta_2) + 2r_{c5,x}r_{c5,z}\sin(2\theta_3)\Big) \\
d_{22} &= I_2 + I_4 + m_2|r_{c2}|^2 + m_3 l_{2,z}^2 + m_4 r_{c4,z}^2 \\
d_{23} &= d_{32} = m_3 l_{2,z} r_{c3,z}\cos(\theta_2 - \theta_3) \\
&\quad + (m_3 l_{2,z} r_{c3,x} + m_4 l_{5,x} r_{c4,z})\sin(\theta_2 - \theta_3) \\
d_{33} &= I_3 + I_5 + m_3|r_{c3}|^2 + m_4 l_{5,x}^2 + m_5|r_{c5}|^2 \\
d_{12} &= d_{13} = d_{21} = d_{31} = 0
\end{aligned}$$

Step 2: In step 2 the vectors describing the center of mass, derived in step 1, are used and the potential energy is easily calculated as

$$V = g^T \sum_{i=2}^{5} m_i r_0^{ci} \tag{3.56}$$

where the gravity vector equals

$$g = \begin{bmatrix} 0 & 0 & g_z \end{bmatrix}^T \tag{3.57}$$

Step 3: From the matrix $D(\theta)$ it is now possible to calculate the Christoffel symbols. The calculations have been done using Mathematica and here only the resulting matrix $C(\theta, dot\theta)$ is given,

$$C(\theta, \dot{\theta}) = \begin{bmatrix} 0 & c_{12} & c_{13} \\ c_{21} & 0 & c_{23} \\ c_{31} & c_{32} & 0 \end{bmatrix} \tag{3.58}$$

where

$$
\begin{aligned}
c_{12} &= \dot{\theta}_1 \Big(m_2 r_{c2,x} r_{c2,z} \cos(\theta_2)^2 - m_2 r_{c2,x} \sin(\theta_2)(l_{1,x} + r_{c2,z} \sin(\theta_2)) + \\
&\quad \cos(\theta_2)(l_{1,x} l_{2,z} m_3 + l_{1,x} m_2 r_{c2,z} + l_{1,x} m_4 r_{c4,z} + (l_{2,z} m_3 r_{c3,x} + \\
&\quad l_{5,x} m_4 r_{c4,z}) \cos(\theta_3) + (l_{2,z}^2 m_3 - m_2 r_{c2,x}^2 + m_2 r_{c2,z}^2 + m_4 r_{c4,z}^2) \sin(\theta_2) + \\
&\quad l_{2,z} m_3 r_{c3,z} \sin(\theta_3)) \Big) \\
c_{13} &= \dot{\theta}_1 (-l_{5,x} m_4 (l_{1,x} + l_{5,x} \cos(\theta_3) + r_{c4,z} \sin(\theta_2)) \sin(\theta_3) + m_3 (r_{c3,z} \cos(\theta_3) - \\
&\quad r_{c3,x} \sin(\theta_3))(l_{1,x} + r_{c3,x} \cos(\theta_3) + l_{2,z} \sin(\theta_2) + r_{c3,z} \sin(\theta_3)) + \\
&\quad m_5 (r_{c5,z} \cos(\theta_3) - r_{c5,x} \sin(\theta_3))(l_{1,x} + r_{c5,x} \cos(\theta_3) + r_{c5,z} \sin(\theta_3))) \\
c_{21} &= -c_{12} \\
c_{23} &= c_{32} = 0 \\
c_{31} &= -c_{13}
\end{aligned}
$$

Step 4: Now the part of the dynamics that come from the potential energy can be derived by letting

$$
g = \begin{bmatrix} \frac{\partial V}{\partial \theta_1} & \frac{\partial V}{\partial \theta_2} & \frac{\partial V}{\partial \theta_3} \end{bmatrix}^T = g_z \begin{bmatrix} 0 \\ g_2 \\ g_3 \end{bmatrix} \tag{3.59}
$$

where

$$
\begin{aligned}
g_2 &= (-m_2 r_{c2,z} + m_3 l_{2,z} - m_4 r_{c4,z}) \sin(\theta_2) - m_2 r_{c2,x} \cos(\theta_2) \\
g_3 &= (-m_3 r_{c3,z} - m_5 r_{c5,z}) \sin(\theta_3) + (-m_3 r_{c3,x} - m_4 l_{5,x} - m_5 r_{c5,x}) \cos(\theta_3)
\end{aligned}
$$

Step 5: From (3.55), (3.58), and (3.59) the equations of motion for the IRB1400 can be written down as

$$
D(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + g(\theta) = \tau \tag{3.60}
$$

In practice there will always be viscous friction and perhaps other kinds of friction. This can be included in the torque as, for example

$$
\tau = \tau_{ext} - f\dot{\theta} \tag{3.61}
$$

3.2 Newton-Euler formulation

We will now, very briefly, go through the Newton-Euler formulation to find the equations of motion for a manipulator. In this formulation we will consider the links one after each other and solve a series of equations involving one link at a time. The result of the calculations will be that, given values of the generalized coordinates ϕ, $\dot{\phi}$, and $\ddot{\phi}$, we can calculate the torque needed to achieve these particular values of the generalized coordinates. This means that we can solve for the $\tau(t)$ that will give a particular trajectory $\phi(t)$ which is very important for control applications.

When using the Newton-Euler formulation it is three facts that are used:

- The law of action and reaction. If body 1 applies a force f_i and a torque τ_i to body 2, then body 2 will react and applies the force $-f_i$ and torque $-\tau_i$ to body 1.
- The fact that

$$\frac{d}{dt}\{mv\} = f \tag{3.62}$$

- and finally that

$$\frac{d}{dt}\{I_0\omega_0\} = \tau_0 \tag{3.63}$$

Since the mass of the manipulator is assumed to be constant (3.62) can be written in the familiar form

$$f = ma \tag{3.64}$$

where f is the force applied to the body and a is the linear acceleration of the center of mass. As has been discussed also in the previous section the moment of inertia I_0 depends on the frame that has been used in the calculations. Here I_0 is referred to the moment of inertia of the body about an inertial frame with the origin in the center of mass of the body. Note that, in general I_0 will depend on the orientation of the body so it will change as a function of time. If we instead let the frame be fixed to the body and express the moment of inertia in this frame the I that we get will be constant. It is still possible to express I_0 as a function of the I and an orientation matrix transforming coordinates in the frame attached to the body to the inertial frame,

$$I_0 = RIR^T \tag{3.65}$$

Now consider (3.63) expressed in the frame rigidly attached to the body. This means that

$$I\dot{\omega} + \omega \times (I\omega) = \tau \tag{3.66}$$

where we have done the differentiation and used the fact that I is not time dependent. The second term is called the *gyroscopic term*. For an explanation of why this term arises see for example [11].

For the manipulator whose equations of motion that we want to find we now assume that we have $n+1$ frames, where frame 0 is an inertial frame and frame $1, \ldots, n$ are attached to the links $1, \ldots, n$. Now a number of vectors have to be defined,

$a_{c,i}$ the acceleration of the center of mass of link i.

$a_{e,i}$ the acceleration of the end of link i (i.e. joint $i+1$).

ω_i the angular velocity of frame i w.r.t. frame 0.

α_i the angular acceleration of frame i w.r.t. frame 0.

g_i the acceleration due to gravity.

f_i the force exerted by link $i-1$ on link i.

τ_i the torque exerted by link $i-1$ on link i.

R_i^{i+1} the rotation matrix transforming coordinates in frame $i+1$ to coordinates in frame i.

Note that the vectors above are expressed in frame i if nothing else is stated.

The following parameters have in common the fact that they are all independent of ϕ, i.e. they are all properties of the manipulator.

m_i the mass of link i.

I_i the inertia matrix for link i calculated in a frame parallel to frame i but in the center of mass of link i.

r_{ici} the vector from joint i to the center of mass of link i.

$r_{i+1,ci}$ the vector from joint $i+1$ to the center of mass of link i.

$r_{i,i+1}$ the vector from joint i to joint $i+1$.

The idea of the Newton-Euler approach is now to write down the force balance and torque balance equations for each link and solve these equations for f_i and τ_i. We start by writing down the force balance equation for link i,

$$f_i - R_i^{i+1} f_{i+1} + m_i g_i = m_i a_{c,i} \tag{3.67}$$

and the torque balance equation,

$$\tau_i - R_i^{i+1}\tau_{i+1} + f \times r_{i,ci} - (R_i^{i+1} f_{i+1}) \times r_{i+1,ci} = I_i\alpha_i + \omega_i \times (I_i\omega_i) \tag{3.68}$$

Now assume that we can calculate $a_{c,i}$, ω_i, α_i, g_i, and R_i^{i+1}, from ϕ, $\dot{\phi}$, and $\ddot{\phi}$ using the kinematic relations discussed in Section 2. Consider (3.67), by using the fact that $f_{n+1} = 0$ it is possible to calculate f_i by solving (3.67) first with $i = n$, and next $i = n-1$, etc.., until $i = 1$.

Next we can use the fact that $\tau_{n+1} = 0$ to solve (3.68) from $i = n$ until $i = 1$.

There is actually also a way to calculate $a_{c,i}$, ω_i, and α_i recursively. This procedure will however not be explained here. The interested reader is referred to for example [11] for the explanation.

3.3 A model of a flexible manipulator

First it should be established that there are numerous ways of approaching the problem of modeling of a flexible manipulator. The flexibilities can be modeled as joint flexibilities or as link flexibilities or, obviously, as both. Common for

many of the approaches to modeling flexible manipulators is, however, that the result is an infinite dimensional system. This system can then be approximated as a lumped system having a finite number of states. The approach that will be discussed here is based on a rigid robot having actuator and gear dynamics that include flexibilities.

3.3.1 Actuator and gear dynamics

The rigid body robot model found in Section 3.1 does not take into account the fact that the robot is not directly actuated on the arm side. Instead, the motor is connected to a gear box which in turn is connected to the arm. This simplified view of the actuator and gear box dynamics is depicted in Figure 3.2.

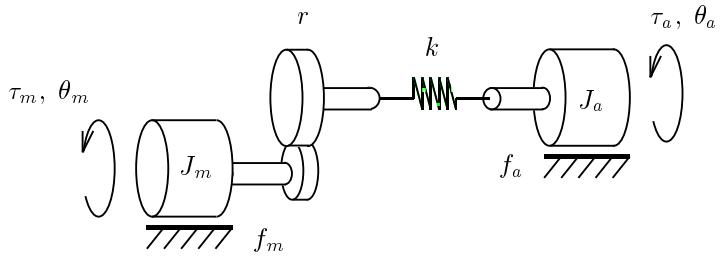


Figure 3.2: A simplified model of the drive train dynamics in an industrial robot.

Writing down the equations for the system in Figure 3.2 we get

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + k(r\theta_m - \theta_a) = \tau_m \tag{3.69}$$

and τ_{ext} in (3.61) becomes $\tau_{ext} = k(\theta_m - \theta_a)$. Obviously θ in (3.60) also changes so that the total dynamics can be modeled as

$$\begin{aligned} J_m\ddot{\theta}_m + f_m\dot{\theta}_m + k(r\theta_m - \theta_a) &= \tau_m \\ D(\theta_a)\ddot{\theta}_a + C(\theta_a, \dot{\theta}_a)\dot{\theta}_a + g(\theta_a) &= k(r\theta_m - \theta_a) - f_a\dot{\theta}_a \end{aligned} \tag{3.70}$$

where, now, the system has $2n$ states compared to the n states of a rigid body model.

3.3.2 Identification of a flexible joint

An experiment is performed on the ABB IRB1400 robot. Moving only joint 1 and keeping all the other joints still. A simplified model for this case is given by the linear model shown in Figure 3.2. The equations describing this model of the system are,

$$\begin{aligned} J_m\ddot{\theta}_m + f_m\dot{\theta}_m + k(r\theta_m - \theta_a) &= \tau_m \\ J_a\ddot{\theta}_a + f_a\dot{\theta}_a - k(r\theta_m - \theta_a) &= 0 \end{aligned} \tag{3.71}$$

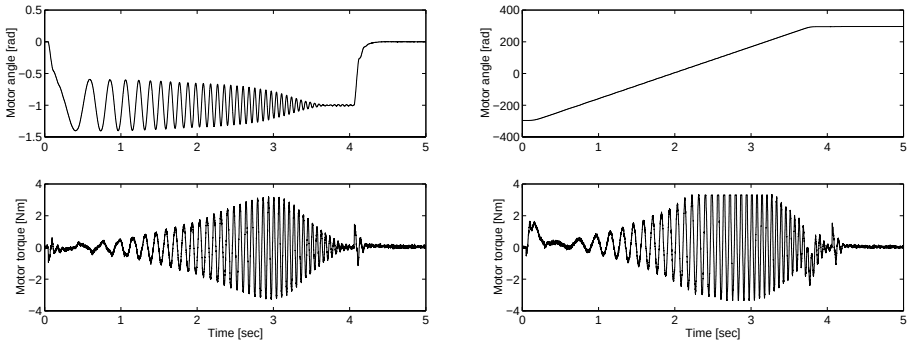
The system can now be written on state space form, having $\dot{x} = Ax + Bu$ and $y = Cx$, with the state variables defined as,

$$x_1 = \theta_m,\ x_2 = \dot{\theta}_m,\ x_3 = \theta_a,\ x_4 = \dot{\theta}_a \tag{3.72}$$

the A, B, and C matrices becomes

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -kr/J_m & -f_m/J_m & k/J_m & 0 \\ 0 & 0 & 0 & 1 \\ kr/J_a & 0 & -k/J_a & -f_a/J_a \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 1/J_m \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.73}$$

Data used for the identification are collected from the ABB IRB1400 robot in the research lab of Division of Automatic Control, Linköping University, and the raw data are shown in Figure 3.3. The data shown in Figure 3.3 are from



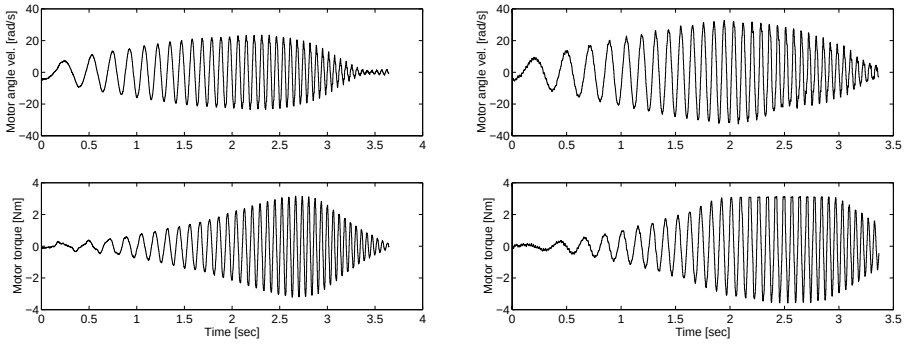
(a) First experiment. *(b)* Second experiment.

Figure 3.3: The raw-data used in the identification experiment.

two different experiments. In Figure 3(a) the robot is excited around one point and in Figure 3(b) the excitation is done while the robot is moving from one point to another.

Instead of directly using the data from the two experiments, the data have been processed. In Figure 3.4 the processed data are shown, estimation data to the left and validation data to the right. The validation and estimation data are part of the data set shown in Figure 3.3 where the beginning and the end of each experiment are removed. Also the mean value has been removed. Note that now the motor angle velocity is the output instead of the motor angle. This, in turn, means that the model from (3.73) changes and the resulting model will have three states only (in order not to loose observability). The new model is given on state space form according to,

$$\dot{x} = Ax + Bu,\ y = Cx$$



(a) Estimation data. *(b)* Validation data.

Figure 3.4: The processed data used in the identification experiment.

where

$$A = \begin{bmatrix} -\frac{f_m}{J_m} & 0 & -\frac{k}{J_m} \\ 0 & 0 & -\frac{k}{J_a} \\ r & -1 & 0 \end{bmatrix}, \; B = \begin{bmatrix} \frac{1}{J_m} \\ 0 \\ 0 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

The states in the model are $x_1 = \dot{\theta}_m$, $x_2 = \dot{\theta}_a$, and $x_3 = r\theta_m - \theta_a$. The physical parameters are estimated using the function `pem` in the system identification toolbox [7] for MATLAB™.

The identified parameters are given with their respective estimated standard deviations in Table 3.1. Compared to the model in (3.71)-(3.73) the number of parameters have been reduced because f_a is assumed to be zero. The gear ratio r is for the IRB1400, joint 1, equal to $\frac{1}{118.461539}$.

Parameter	Estimated value	Estimated standard deviation
J_m	$5.9 \cdot 10^{-4}$	$2.2 \cdot 10^{-5}$
J_a	$5.2 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$
f_m	$1.3 \cdot 10^{-2}$	$3.2 \cdot 10^{-4}$
k	677.2	29.0

Table 3.1: The estimated parameters in the model.

In Figure 3.5 a simulation is shown where the validation data set is used on the identified model. Obviously the model describes data well but, still, a more thorough analysis of the model has to be done. For example, how could a change of amplitude of the exciting torque signal change the behavior of the model. The true system has nonlinearities and these can give raise to very different behavior depending on the amplitude of the exciting signals.

In Figure 3.6 the result from a simulation using the original model, having

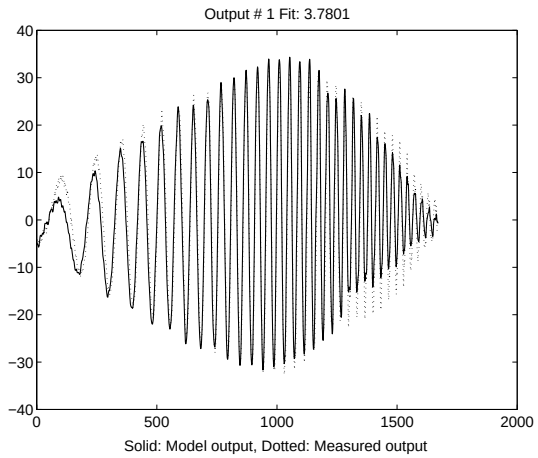


Figure 3.5: Comparison between simulated output and true output from the estimated model, using the validation data set.

the motor angle as output, is shown. The low frequency gain is not very well estimated and in order not to have too high deviation the input has been filtered through a high pass filter with cut-off frequency 0.01 of the Nyquist frequency, in this case this means 2.5Hz.

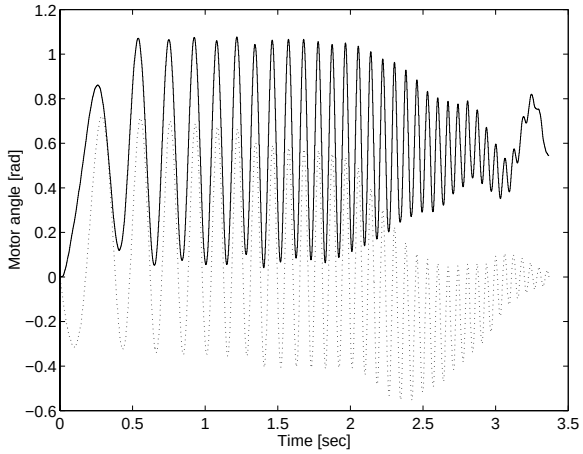


Figure 3.6: Comparison between simulated output and true output from the original model, having four states. The simulated output is solid while the measured output is given by the dotted line.

4 Summary and conclusions

The report gives an overview of the field of robot modeling with an example where the given tools and algorithms are applied. It is quite straight forward to build kinematic models using the Denavit-Hartenberg representation and these models are very easy to implement in MATLAB™ using the Robotics Toolbox [1]. The inverse kinematics is also quite straight forward to find and implement, at least for open kinematic chain manipulators. The notion of quaternions is covered because of the fact that this representation is used in the ABB control system.

The rigid body dynamics for a holonomic system, a rigid manipulator, is thoroughly discussed using the *Euler-Lagrange* formulation. The dynamical equations for the IRB1400 is found by implementing the Euler-Lagrange formulation in Mathematica. Another method, the *Newton-Euler* formulation, is shortly reviewed without applying it to the IRB1400.

A starting point for modeling of a flexible joint manipulator is given. The model presented is a linear two-mass model and this simple model is evaluated on real data from an experiment. Using data from the experiment it is shown that the model actually describes the system well, at least with the given amplitude of the excitation and over the particular frequence interval that has been considered.

What is left for further work?

- The modeling of a flexible joint serves as a good starting point for a project where also external measurements from the arm-side of the robot could be used. The ultimate goal is of course to find a model that describes exactly how the tool moves as a function of motor positions. The nonlinearities, friction, backlash, etc., has to be taken into account as well.

- Identification and estimation of all the parameters in the robot model given by the Euler-Lagrange formulation is not discussed at all in this report. This is a very interesting topic both from an academic as well as an industrial point of view.

References

[1] P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996.

[2] J. J. Craig. *Adaptive Control of Mechanical Manipulators.* Addison-Wesley Publishing Company, 1988.

[3] K. Dobrovodsky. Quaternion position representation in robot kinematic structures. In *Proceedings of the International Conference on Control 1994*, pages 561–564, 1994.

[4] J. Funda and R.P. Paul. A comparison of transforms and quaternions in robotics. In *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, volume 2, pages 886–891, 1988.

[5] J. Funda, R.H. Taylor, and R.P. Paul. On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics adn Automation*, 6(3):382–388, 1990.

[6] H. Goldstein. *Classical Mechanics.* Addison-Wesley Publishing Company, Inc., second edition, 1980.

[7] L. Ljung. *System Identification Toolbox - For Use with MATLAB™.* The MathWorks Inc., 1995.

[8] J.B. Marion and S.T. Thornton. *Classical Dynamics of Particles & Systems.* Harcourt Brace Jovanovich, Publishers, third edition, 1988.

[9] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the ACM SIGGRAPH'85*, pages 245–254, San Francisco, USA, 1985.

[10] M. W. Spong, F. L. Lewis, and C. T. Abdallah, editors. *Robot Control: Dynamics, Motion Planning, and Analysis.* IEEE Control Systems Society, IEEE Press, 1992.

[11] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* Wiley, 1989.