HAL
open science

Modeling and Control of Manipulators - Part II: Dynamics and Control

Wisama Khalil

To cite this version:

Wisama Khalil. Modeling and Control of Manipulators - Part II: Dynamics and Control. Doctoral. GdR Robotics Winter School: Robotica Principia, Centre de recherche Inria Sophia Antipolis – Méditérranée, France. 2019. cel-02130022

HAL Id: cel-02130022

https://inria.hal.science/cel-02130022v1

Submitted on 15 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Modeling and Control of Manipulators

Part II: Dynamics and Control

Wisama KHALIL

Ecole Centrale de Nantes

Master I EMARO

European Master on Advanced Robotics

Master I ARIA

Master Automatique, Robotique et Informatique Appliquée

Master in Control Engineering, Robotics and Applied Informatics

2013-2014

Education and Culture

Erasmus Mundus

Centrale Nantes

Dynamic modeling of serial robots 155
7.1. Introduction 155
7.2. Notations 156
7.3. Lagrange formulation 157
7.3.1. Introduction 157
7.3.2. General form of the dynamic equations 158
7.3.3. Computation of the elements of A, C and Q 159
7.3.3.1. Computation of the kinetic energy 159
7.3.3.2. Computation of the potential energy 162
7.3.3.3. Dynamic model properties 162
7.3.4. Considering friction 163
7.3.5. Considering the rotor inertia of actuators 165
7.3.6. Considering the forces and moments exerted by the end-effector on the environment 165
7.3.7. Relation between joint torques and actuator torques 166
7.3.8. Modeling of robots with elastic joints 166
7.4. Newton-Euler formulation 169
7.4.1. Introduction 169
7.4.2. Newton-Euler inverse dynamics linear in the inertial parameters ... 169
7.4.3. Practical form of the Newton-Euler algorithm 171
7.5. Real time computation of the inverse dynamic model 173
7.5.1. Introduction 173
7.5.2. Customization of the Newton-Euler formulation 177
7.5.3. Utilization of the base inertial parameters 179
7.6. Direct dynamic model 180
7.6.1. Using the inverse dynamic model to solve the direct dynamic problem 180
7.6.2. Recursive computation of the direct dynamic model 182
7.6.2.1 Algorithm of computation of the direct dynamic model 182
7.6.2.2 Demonstration of relations [7.67] and [7.68] 184
7.7. Conclusion 185

Trajectory generation.....187
8.1. Introduction 187
8.2. Trajectory generation and control loops 188
8.3. Point-to-point trajectory in the joint space..... 189
8.3.1. Polynomial interpolation..... 190
8.3.1.1. Linear interpolation..... 190
8.3.1.2. Cubic polynomial..... 190
8.3.1.3. Quintic polynomial 191
8.3.1.4. Computation of the minimum traveling time..... 193
8.3.2. Bang-bang acceleration profile 194
8.3.3. Trapezoidal velocity model..... 195
8.3.4. Continuous acceleration profile with constant velocity phase 201
8.4. Point-to-point trajectory in the task space 204
8.5. Conclusion..... 207

Motion control 209
9.1. Introduction 209
9.2. Equations of motion 209
9.3. PID control 210
9.3.1. PID control in the joint space 210
9.3.2. Stability analysis 212
9.3.3. PID control in the task space 214
9.4. Linearizing and decoupling control 215
9.4.1. Introduction 215
9.4.2. Computed torque control in the joint space 216
9.4.2.1. Principle of the control 216
9.4.2.2. Tracking control scheme 217
9.4.2.3. Position control scheme 218
9.4.2.4. Predictive dynamic control 219
9.4.2.5. Practical computation of the computed torque control laws ... 219
9.4.3. Computed torque control in the task space 220
9.7. Conclusion 222

Appendix: Dynamic model of Staubli 90

Modélisation, identification et commande des robots

Chapter 7

Dynamic modeling of serial robots

7.1. Introduction

The *inverse dynamic model* provides the joint torques and forces in terms of the joint positions, velocities and accelerations. It is described by:

$$\mathbf{\Gamma} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{F}_e) \qquad [7.1]$$

with:

- $\mathbf{\Gamma}$: vector of joint torques or forces, depending on whether the joint is revolute or prismatic respectively. In the sequel, we will only write *joint torques*;
- $\mathbf{q}$: vector of joint positions;
- $\dot{\mathbf{q}}$: vector of joint velocities;
- $\ddot{\mathbf{q}}$: vector of joint accelerations;
- $\mathbf{F}_e$: vector of forces and moments exerted by the robot on the environment.

Equation [7.1] is an inverse dynamic model because it defines the system input as a function of the output variables. It is often called the *dynamic model*. This form of model which is expressed in terms of Lagrangian variables (joint variables and their derivatives) is called "Lagrangian Model". The Euler model makes use of the Eulerian variables (linear and rotational Cartesian velocities and accelerations).

The *direct dynamic model* gives the joint accelerations in terms of the joint positions, velocities and torques. It is represented by the relation:

$$\ddot{\mathbf{q}} = \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{\Gamma}, \mathbf{F}_e) \qquad [7.2]$$

The dynamic model of robots plays an important role in their design and control. For robot design, the inverse dynamic model can be used to select the actuators [Chedmail 90b], [Potkonjak 86], while the direct dynamic model is employed to carry out simulations (§ 7.7) for the purpose of testing the performance of the robot and to study the relative merits of possible control schemes. Regarding robot control, the inverse dynamic model is used to compute the actuator torques, which are needed to achieve a desired motion . It is also used to identify the dynamic parameters that are necessary for both control and simulation applications .

Several approaches have been proposed to model the dynamics of robots [Renaud 75], [Coiffet 81], [Vukobratovic 82]. The most frequently employed in robotics are the Lagrange formulation [Uicker 69], [Khalil 76], [Renaud 80a], [Hollerbach 80], [Paul 81], [Megahed 84], [Renaud 85] and the Newton-Euler formulation [Hooker 65], [Armstrong 79], [Luh 80b], [Orin 79], [Khalil 85a], [Khosla 86], [Khalil 87b], [Renaud 87].

In this chapter, we present the dynamic modeling of serial robots using these two formulations. The problem of calculating a minimum set of inertial parameters (base parameters) will not be covered in this year. We will focus our study on the minimization of the number of operations of the dynamic model in view of its real time computation for control purposes. Lastly, the computation of the direct dynamic model will be addressed.

7.2. Notations

The main notations used in this chapter are compiled below:

- $\mathbf{a}_j$ unit vector along axis $\mathbf{z}_j$;
- $\mathbf{F}_{tj}$ total external wrench (forces and moments) on link j, composed of $\mathbf{f}_{tj}$ and $\mathbf{m}_{tj}$;
- $\mathbf{f}_j$ force exerted on link j by link j – 1;
- $\mathbf{f}_{ej}$ force exerted by link j on the environment;
- F_{cj} parameter of Coulomb friction acting at joint j;
- F_{vj} parameter of viscous friction acting at joint j;
- $\mathbf{g}$ gravitational acceleration;
- G_j center-of-mass of link j;
- $\mathbf{I}_{Gj}$ inertia tensor of link j about G_j and with respect to a frame parallel to frame R_j;
- Ia_j moment of inertia of the rotor and the transmission system of actuator j referred to the joint side;
- ${}^j\mathbf{I}_{Oj}$ inertia tensor of link j with respect to frame R_j. It is described by:

$$ {}^{j}\mathbf{I}_{O_j} = \begin{bmatrix} \int (y^2+z^2)\,dm & -\int xy\,dm & -\int xz\,dm \\ -\int xy\,dm & \int (x^2+z^2)\,dm & -\int yz\,dm \\ -\int xz\,dm & -\int yz\,dm & \int (x^2+y^2)\,dm \end{bmatrix} = \begin{bmatrix} XX_j & XY_j & XZ_j \\ XY_j & YY_j & YZ_j \\ XZ_j & YZ_j & ZZ_j \end{bmatrix} \quad [7.3] $$

$\mathbb{I}_j$ (6x6) spatial inertia matrix of link j (relation [7.21]);
$\mathbf{L}_j$ position vector between O_{j-1} and O_j;
M_j mass of link j;
$\mathbf{MS}_j$ first moments of link j with respect to frame R_j, equal to $M_j\ \mathbf{S}_j$. The components of ${}^{j}\mathbf{MS}_j$ are denoted by $[\ MX_j \quad MY_j \quad MZ_j\ ]^T$;
$\mathbf{m}_{Gj}$ moment of external forces on link j about G_j;
$\mathbf{m}_{tj}$ moment of external forces on link j about O_j;
$\mathbf{m}_j$ moment about O_j exerted on link j by link $j-1$;
$\mathbf{m}_{ej}$ moment about O_j exerted by link j on the environment;
$\mathbf{s}_j$ vector of the center-of-mass coordinates of link j. It is equal to $\mathbf{O}_j\mathbf{G}_j$;
$\mathbf{v}_j$ linear velocity of O_j;
$\mathbf{V}_j$ (6x1) kinematic screw vector of link j, formed by the components of $\mathbf{v}_j$ and $\boldsymbol{\omega}_j$;
$\dot{\mathbf{v}}_j$ linear acceleration of O_j;
$\mathbf{v}_{Gj}$ linear velocity of the center-of-mass of link j;
$\dot{\mathbf{v}}_{Gj}$ linear acceleration of the center-of-mass of link j;
$\boldsymbol{\omega}_j$ angular velocity of link j;
$\dot{\boldsymbol{\omega}}_j$ angular acceleration of link j.

7.3. Lagrange formulation

7.3.1. *Introduction*

The dynamic model of a robot with several degrees of freedom represents a complicated system. The Newton-Euler method developed in § 7.5 presents an efficient and systematic approach to solving this problem. In this section, we develop a simple Lagrange method to present the general form of the dynamic model of robots and to get an insight into its properties. Firstly, we consider an ideal system without friction or elasticity, exerting neither forces nor moments on the environment. These phenomena will be covered in § 7.3.4 through 7.3.8.

The Lagrange formulation describes the behavior of a dynamic system in terms of work and energy stored in the system. The Lagrange equations are written in the form:

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad \text{for } i = 1, ..., n \qquad [7.4a]$$

This relation can be written in matrix form as:

$$\Gamma = \frac{d}{dt}\left[\frac{\partial L}{\partial \dot{\mathbf{q}}}\right]^T - \left[\frac{\partial L}{\partial \mathbf{q}}\right]^T$$

where L is the Lagrangian of the robot defined as the difference between the kinetic energy E and the potential energy U of the system:

$$L = E - U \qquad [7.4b]$$

7.3.2. *General form of the dynamic equations*

The kinetic energy of the system is a quadratic function in the joint velocities such that:

$$E = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{A}\, \dot{\mathbf{q}} \qquad [7.5]$$

where **A** is the (nxn) symmetric and positive definite *inertia matrix* of the robot. Its elements are functions of the joint positions. The (i, j) element of **A** is denoted by A_{ij}.

Since the potential energy is a function of the joint positions, equations [7.4] and [7.5] lead to:

$$\mathbf{\Gamma} = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{q} + \mathbf{Q}(\mathbf{q}) \qquad [7.6]$$

where:

- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\, \dot{\mathbf{q}}$ is the (nx1) vector of Coriolis and centrifugal torques, such that:
  $\mathbf{C}\,\dot{\mathbf{q}} = \dot{\mathbf{A}}\,\dot{\mathbf{q}} - \frac{\partial E}{\partial \mathbf{q}}$

- $\mathbf{Q} = [Q_1 \ \dots \ Q_n]^T$ is the vector of gravity torques.

Consequently, the dynamic model of a robot is described by n coupled and nonlinear second order differential equations.

There exist several forms for the vector $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}}$. Using the *Christoffell symbols* $c_{i,jk}$, the (i, j) element of the matrix **C** can be written as:

$$\begin{cases} C_{ij} = \sum_{k=1}^{n} c_{i,jk}\dot{q}_k \\ c_{i,jk} = \frac{1}{2}\left[\frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} - \frac{\partial A_{jk}}{\partial q_i}\right] \end{cases} \qquad [7.7]$$

The Q_i element of the vector **Q** is calculated according to:

$$Q_i = \frac{\partial U}{\partial q_i} \qquad [7.8]$$

The elements of **A**, **C** and **Q** are functions of the geometric and inertial parameters of the robot.

7.3.3. *Computation of the elements of A, C and Q*

To compute the elements of **A**, **C** and **Q**, we begin by symbolically computing the expressions of the kinetic and potential energies of all the links of the robot. Then, we proceed as follows:

i – the element A_{ij} is equal to $\frac{\partial E}{\partial \dot{q}_i \dot{q}_j}$ this leads to:

- the element A_{ii} is equal to the coefficient of $(\dot{q}_i^2/2)$ in the expression of the kinetic energy, while A_{ij}, for $i \neq j$, is equal to the coefficient of $\dot{q}_i\,\dot{q}_j$;

ii –the elements of **C** are obtained from equation [7.7];

iii–the elements of **Q** are obtained from equation [7.8].

7.3.3.1. *Computation of the kinetic energy*

The kinetic energy of the robot is given as:

$$E = \sum_{j=1}^{n} E_j \qquad [7.9]$$

where E_j denotes the kinetic energy of link j, which can be computed by:

$$E_j = \frac{1}{2}(\omega_j^T \mathbf{I}_{Gj}\omega_j + M_j \mathbf{v}_{Gj}^T \mathbf{v}_{Gj}) \qquad [7.10]$$

Since the velocity of the center-of-mass can be expressed as (Figure 7.1):

$$\mathbf{V}_{Gj} = \mathbf{V}_j + \boldsymbol{\omega}_j \times \mathbf{S}_j \qquad [7.11]$$

and since:

$$\mathbf{I}_{Oj} = \mathbf{I}_{Gj} - M_j\, \hat{\mathbf{s}}_j\, \hat{\mathbf{s}}_j \qquad [7.12]$$

equation [7.10] becomes:

$$E_j = \frac{1}{2}(\boldsymbol{\omega}_j^T \mathbf{I}_{Oj}\boldsymbol{\omega}_j + M_j \mathbf{v}_j^T \mathbf{v}_j + 2\mathbf{MS}_j^T\ (\mathbf{v}_j \times \boldsymbol{\omega}_j)) \qquad [7.13]$$

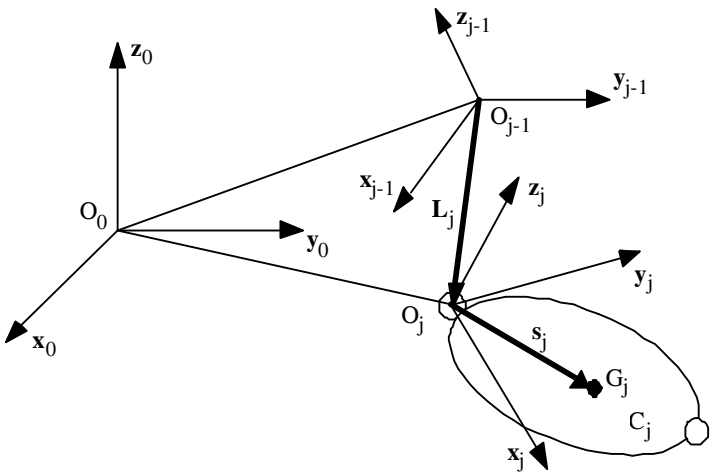

Figure 7.1. *Composition of velocities*

Equation [7.10] is not linear in the coordinates of the vector $\mathbf{S}_j$. On the contrary, equation [7.13] is linear in the elements of M_j, $\mathbf{MS}_j$ and $\mathbf{J}_j$, which we call the *standard inertial parameters*. The linear and angular velocities $\mathbf{V}_j$ and $\boldsymbol{\omega}_j$ are computed using the following recursive equations (Figure 7.1):

$$\boldsymbol{\omega}_j = \boldsymbol{\omega}_{j-1} + \bar{\sigma}_j\, \dot{q}_j\, \mathbf{a}_j \qquad [7.14]$$

$$\mathbf{v}_j = \mathbf{v}_{j-1} + \boldsymbol{\omega}_{j-1} \times \mathbf{L}_j + \sigma_j\, \dot{\mathbf{q}}_j\, \mathbf{a}_j \qquad [7.15]$$

If the base of the robot is fixed, the previous equations are initialized by $\mathbf{v}_0 = \mathbf{0}$ and $\boldsymbol{\omega}_0 = \mathbf{0}$.

All the elements appearing in equation [7.13] must be expressed in the same frame. The most efficient way is to express them relative to frame R_j which is fixed with the link j. Therefore, equations [7.13], [7.14] and [7.15] are rewritten as:

$$E_j = \frac{1}{2}\,[{}^j\boldsymbol{\omega}_j^T\,{}^j\mathbf{I}_{Oj}\,{}^j\boldsymbol{\omega}_j + M_j\,{}^j\mathbf{v}_j^T\,{}^j\mathbf{v}_j + 2\,{}^j\mathbf{MS}_j^T\,({}^j\mathbf{v}_j \times {}^j\boldsymbol{\omega}_j)] \qquad [7.16]$$

$${}^j\boldsymbol{\omega}_j = {}^j\mathbf{R}_{j-1}\,{}^{j-1}\boldsymbol{\omega}_{j-1} + \bar{\sigma}_j\,\dot{q}_j\,{}^j\mathbf{a}_j = {}^j\boldsymbol{\omega}_{j-1} + \bar{\sigma}_j\,\dot{q}_j\,{}^j\mathbf{a}_j \qquad [7.17]$$

$${}^j\mathbf{v}_j = {}^j\mathbf{R}_{j-1}\,({}^{j-1}\mathbf{v}_{j-1} + {}^{j-1}\boldsymbol{\omega}_{j-1} \times {}^{j-1}\mathbf{P}_j) + \sigma_j\,\dot{\mathbf{q}}_j\,{}^j\mathbf{a}_j \qquad [7.18]$$

The parameters ${}^j\mathbf{I}_{Oj}$ and ${}^j\mathbf{MS}_j$ are constants.

Using the spatial notation, the kinetic energy can be written in the following compact form by using screw notations:

$$E_j = \frac{1}{2}\,{}^j\mathbf{V}_j^T\,{}^j\mathbb{I}_j\,{}^j\mathbf{V}_j \qquad [7.19]$$

where:

$${}^j\mathbf{V}_j = \begin{bmatrix} {}^j\mathbf{v}_j \\ {}^j\boldsymbol{\omega}_j \end{bmatrix} \qquad [7.20]$$

$${}^j\mathbb{I}_j = \begin{bmatrix} M_j\mathbf{I_3} & {}^j\mathbf{M}\hat{\mathbf{S}}_j^T \\ {}^j\mathbf{M}\hat{\mathbf{S}}_j & {}^j\mathbf{I}_{Oj} \end{bmatrix} \qquad [7.21]$$

where ${}^j\mathbf{M}\hat{\mathbf{S}}_j^T = -{}^j\mathbf{M}\hat{\mathbf{S}}_j$

The recursive velocity relations [7.17] and [7.18] can be combined as follows:

$${}^j\mathbf{V}_j = {}^j\mathbf{S}_{j-1}\,{}^{j-1}\mathbf{V}_{j-1} + \dot{q}_j\,{}^j\mathbb{a}_j \qquad [7.22]$$

where ${}^j\mathbf{S}_{j-1}$ is the (6x6) screw transformation matrix defined in [2.46] as:

$${}^j\mathbf{S}_{j-1} = \begin{bmatrix} {}^j\mathbf{R}_{j-1} & -{}^j\mathbf{R}_{j-1}\,{}^{j-1}\hat{\mathbf{P}}_j \\ \mathbf{0}_3 & {}^j\mathbf{R}_{j-1} \end{bmatrix} = \begin{bmatrix} {}^j\mathbf{R}_{j-1} & {}^j\hat{\mathbf{P}}_{j-1}\,{}^j\mathbf{R}_{j-1} \\ \mathbf{0}_3 & {}^j\mathbf{R}_{j-1} \end{bmatrix} \qquad [7.23a]$$

and where ${}^j\mathbb{a}_j$ is the (6x1) column matrix:

$$ {}^{j}\mathbf{a}_j = \begin{bmatrix} \sigma_j \; {}^{j}\mathbf{a}_j \\ \bar{\sigma}_j \; {}^{j}\mathbf{a}_j \end{bmatrix} \qquad [7.23b] $$

7.3.3.2. *Computation of the potential energy*

The potential energy is given by:

$$ U = \sum_{j=1}^{n} U_j = -\sum_{j=1}^{n} M_j \, \mathbf{g}^T (\mathbf{L}_{0,j} + \mathbf{s}_j) \qquad [7.24] $$

where $\mathbf{L}_{0,j}$ is the position vector from the origin O_0 to O_j. Projecting the vectors appearing in [7.24] into frame R_0, we obtain:

$$ U_j = -M_j \, {}^{0}\mathbf{g}^T ({}^{0}\mathbf{P}_j + {}^{0}\mathbf{R}_j \, {}^{j}\mathbf{S}_j) \qquad [7.25a] $$

an expression that can be rewritten linearly in M_j and the elements of ${}^{j}\mathbf{MS}_j$ as:

$$ U_j = -{}^{0}\mathbf{g}^T (M_j \, {}^{0}\mathbf{P}_j + {}^{0}\mathbf{R}_j \, {}^{j}\mathbf{MS}_j) = -\begin{bmatrix} {}^{0}\mathbf{g}^T & 0 \end{bmatrix} {}^{0}\mathbf{T}_j \begin{bmatrix} {}^{j}\mathbf{MS}_j \\ M_j \end{bmatrix} \qquad [7.25b] $$

Since the kinetic and potential energies are linear in the elements of ${}^{j}\mathbf{I}_{Oj}$, ${}^{j}\mathbf{MS}_j$, M_j, we deduce that the dynamic model is also linear in these parameters.

7.3.3.3. *Dynamic model properties*

In this section, we summarize some important properties of the dynamic model of robots:

a) the inertia matrix **A** is symmetric and positive definite;

b) the energy of link j is a function of $(q_1, \ldots, q_j)$ and $(\dot{q}_1, \ldots, \dot{q}_j)$;

c) the element A_{ij} is a function of $q_{k+1}, \ldots, q_n$, with $k = \min(i, j)$, and of the inertial parameters of links r, ..., n, with $r = \max(i, j)$;

d) from property b and equation [7.4], we deduce that Γ_i is a function of the inertial parameters of links i, ..., n;

e) the matrix $[\frac{d}{dt}\mathbf{A}\text{-}2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]$ is skew-symmetric for the choice of the matrix $\mathbf{C}$ given by equation [7.7] [Koditschek 84], [Arimoto 84]. This property is used in Chapter 14 for the stability analysis of certain control schemes;

f) the inverse dynamic model is linear in the elements of the standard inertial parameters M_j, ${}^j\mathbf{MS}_j$ and ${}^j\mathbf{J}_j$. This property is exploited to identify the dynamic parameters (inertial and friction parameters), to reduce the computation burden of the dynamic model, and to develop adaptive control schemes;

g) there exist some positive real numbers $a_1, ..., a_7$ such that for any values of $\mathbf{q}$ and $\dot{\mathbf{q}}$ we have [Samson 87]:

$$\| \mathbf{A}(\mathbf{q}) \| \leq a_1 + a_2 \| \mathbf{q} \| + a_3 \| \mathbf{q} \|^2$$

$$\| \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \| \leq \| \dot{\mathbf{q}} \| (a_4 + a_5 \| \mathbf{q} \|)$$

$$\| \mathbf{Q} \| \leq a_6 + a_7 \| \mathbf{q} \|$$

where $\|*\|$ indicates a matrix or vector norm. If the robot has only revolute joints, these relations become:

$$\| \mathbf{A}(\mathbf{q}) \| \leq a_1$$

$$\| \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \| \leq a_4 \| \dot{\mathbf{q}} \|$$

$$\| \mathbf{Q} \| \leq a_6$$

h) a robot is a passive system which dissipates energy. This property is related to property e).

7.3.4. *Considering friction*

Friction plays a dominant role in limiting the quality of robot performance. Non-compensated friction produces static error, delay, and limit cycle behavior [Canudas de Wit 90]. Many works have been devoted to studying friction torque in the joint and transmission systems. Various friction models have been proposed in the literature [Dahl 77], [Canudas de Wit 89], [Armstrong 88], [Armstrong 91], [Armstrong 94]. In general, three kinds of frictions are noted: Coulomb friction, static friction, and viscous friction.

The model based on Coulomb friction assumes a constant friction component that is independent of the magnitude of the velocity. The static friction is the torque necessary to initiate motion from rest. It is often greater than the Coulomb friction (Figure 7.2a). The viscous friction is generally represented as being proportional to the velocity, but experimental studies [Armstrong 88] have pointed out the Stribeck phenomenon that arises from the use of fluid lubrication. It results in decreasing friction with increasing velocity at low velocity, then the friction becomes

proportional to velocity (Figure 7.2b). A general friction model describing these components is given by:

$$\Gamma_{fi} = F_{ci}\,\mathrm{sign}(\dot{q}_i) + F_{vi}\,\dot{q}_i + (F_{sti} - F_{ci})\,\mathrm{sign}(\dot{q}_i)\,e^{-|\dot{q}_i|B_i} \qquad [7.26]$$

In this expression, Γ_{fi} denotes the friction torque of joint i, F_{ci} and F_{vi} indicate the Coulomb and viscous friction parameters respectively. The static torque is equal to $F_{sti}\,\mathrm{sign}(\dot{q}_i)$.

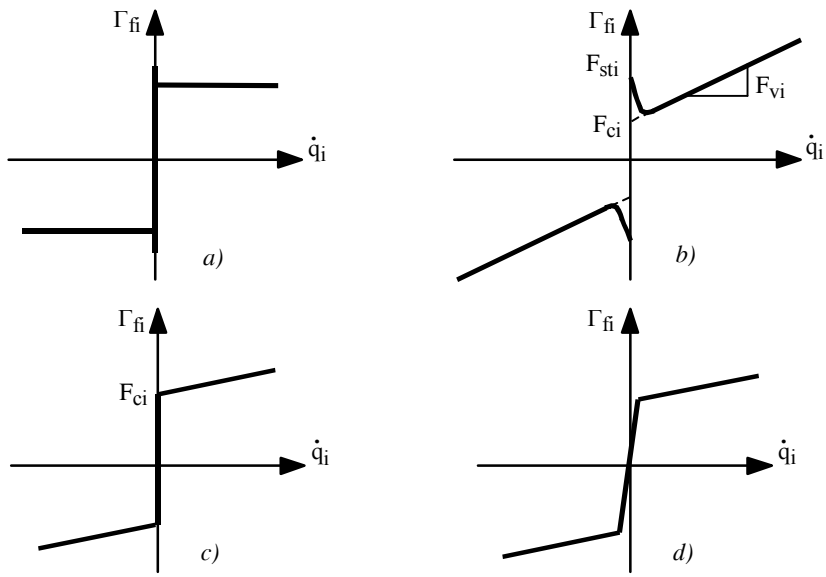

Figure 7.2. *Friction models*

The most often employed model is composed of Coulomb friction together with viscous friction (Figure 7.2c). Therefore, the friction torque at joint i is written as:

$$\Gamma_{fi} = F_{ci}\,\mathrm{sign}(\dot{q}_i) + F_{vi}\,\dot{q}_i \qquad [7.27]$$

To take into account the friction in the dynamic model of a robot, we add the vector $\mathbf{\Gamma}_f$ on the right side of equation [7.6] such that:

$$\mathbf{\Gamma}_f = \mathbf{diag}(\dot{\mathbf{q}})\mathbf{F}_v + \mathbf{diag}(\mathrm{sign}(\dot{\mathbf{q}}))\mathbf{F}_c \qquad [7.28]$$

where:

- $\mathbf{F}_v = [\ F_{v1}\ \ \dots\ \ F_{vn}\ ]^T$;
- $\mathbf{F}_c = [\ F_{c1}\ \ \dots\ \ F_{cn}\ ]^T$;
- **diag**($\dot{\mathbf{q}}$) is the diagonal matrix whose elements are the components of $\dot{\mathbf{q}}$.

This friction model can be approximated by a piecewise linear model as shown in Figure 7.2d.

7.3.5. *Considering the rotor inertia of actuators*

The kinetic energy of the rotor (and transmission system) of actuator j, is given by the expression $\frac{1}{2}\ Ia_j\,\dot{q}_j^2$. The inertial parameter Ia_j denotes the equivalent inertia referred to the joint velocity. It is given by:

$$Ia_j = Nj^2 Jm_j \qquad [7.29]$$

where J_{mj} is the moment of inertia of the rotor and transmissions of actuator j, N_j is the transmission ratio of joint j axis, equal to $\dot{q}_{mj}$ / $\dot{q}_j$ where $\dot{q}_{mj}$ denotes the rotor velocity of actuator j. In the case of a prismatic joint, Ia_j is an equivalent mass.

In order to consider the rotor inertia in the dynamic model of the robot, we add the inertia (or mass) Ia_j to the A_{jj} element of the matrix **A**, that is to say we add the term $Ia_j\,\ddot{q}_j$ on Γj.

Note that such modeling neglects the gyroscopic effects of the rotors, which take place when the actuator is fixed on a moving link. However, this approximation is justified for high gear transmission ratios. For more accurate modeling of the rotors the reader is referred to [Llibre 83], [Chedmail 86], [Murphy 93], [Sciavicco 94].

7.3.6. *Considering the forces and moments exerted by the end-effector on the environment*

We have seen in § 5.9 that the joint torque vector $\mathbf{\Gamma}_e$ necessary to exert a given wrench $\mathbb{f}_{en}$ on the environment is obtained using the basic static equation:

$$\mathbf{\Gamma}_e = \mathbf{J}_n^T\,\mathbf{F}_{en} \qquad [7.30]$$

Thus, we have to add the vector $\mathbf{\Gamma}_e$ on the right side of equation [7.6].

7.3.7. *Relation between joint torques and actuator torques*

In general, the joint variables are not equal to the motor variables because of the existence of transmission systems or because of couplings between the actuator variables. The relation between joint torques and actuator torques can be obtained using the principle of virtual work. Let the relationship between the infinitesimal joint displacement $\mathbf{dq}$ and the infinitesimal actuator variable $\mathbf{dq}_m$ be given by:

$$\mathbf{dq} = \mathbf{J}_{\mathbf{q}m}\, \mathbf{dq}_m \qquad [7.31]$$

where $\mathbf{J}_{\mathbf{q}m}$ is the Jacobian of $\mathbf{q}$ with respect to $\mathbf{q}_m$, equal to $\frac{\partial \mathbf{q}}{\partial \mathbf{q}_m}$.

The virtual work can be written as:

$$\mathbf{\Gamma}^T \mathbf{dq}^* = \boldsymbol{\tau}_m^T \mathbf{dq}_m^* \qquad [7.32]$$

where $\boldsymbol{\tau}_m$ is the actuator torque vector and the superscript (*) indicates virtual displacements.

Combining equations [7.31] and [7.32] yields:

$$\boldsymbol{\tau}_m = \mathbf{J}_{\mathbf{q}m}^T \mathbf{\Gamma} \qquad [7.33]$$

7.3.8. *Modeling of robots with elastic joints*

The presence of joint flexibility is a common feature of many current industrial robots. The joint elasticity may arise from several sources, such as elasticity in the gears, transmission belts, harmonic drives, etc. It follows that a time-varying displacement is introduced between the position of the driving actuator and the joint position. The joint elasticity is modeled as a linear torsional spring for revolute joints and a linear spring for prismatic joints [Khalil 78], [Spong 87]. Thus, the dynamic model requires twice the number of generalized coordinates to completely characterize the configuration of the links and the rotors of the actuators. Let $\mathbf{q}_M$ denotes the (nx1) vector of rotor positions as reflected through the gear ratios (Figure 7.3). Consequently, the vector of joint deformations is given by $(\mathbf{q} - \mathbf{q}_M)$. Note that the direct geometric model is only a function of the joint variables $\mathbf{q}$.

The potential energy of the springs is given by:

$$U_k = \frac{1}{2} (\mathbf{q} - \mathbf{q}_M)^T \mathbf{k} (\mathbf{q} - \mathbf{q}_M) \qquad [7.34]$$

where **k** is the (nxn) definite positive joint stiffness matrix.

The dynamic equations are obtained using the Lagrange equation, i.e.:

$$\mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}} + \mathbf{Q} + \mathbf{k}\,(\mathbf{q} - \mathbf{q}_M) + \mathbf{diag}(\dot{\mathbf{q}})\,\mathbf{F}_v + \mathbf{diag}(\mathrm{sign}(\dot{\mathbf{q}}))\,\mathbf{F}_c = \mathbf{0} \qquad [7.35a]$$

$$\mathbf{I}_a\,\ddot{\mathbf{q}}_M + \mathbf{diag}(\dot{\mathbf{q}}_M)\,\mathbf{F}_{vm} + \mathbf{diag}(\mathrm{sign}(\dot{\mathbf{q}}_M))\,\mathbf{F}_{cm} - \mathbf{k}\,(\mathbf{q} - \mathbf{q}_M) = \mathbf{\Gamma} \qquad [7.35b]$$

where $\mathbf{I}_a$ is the (nxn) diagonal matrix containing the inertia of the rotors, $\mathbf{F}_{vm}$ and $\mathbf{F}_{cm}$ are the (nx1) vectors containing the viscous and Coulomb parameters of the actuators and transmissions referred to the joint side. The joint friction terms can easily be included in equation [7.35a].

A general and systematic method to model systems with lumped elasticity is presented in [Khalil 00a].

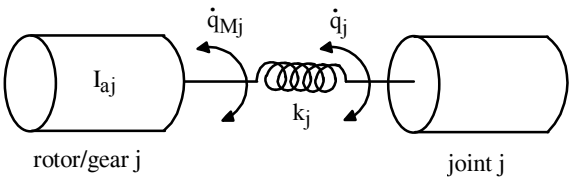

Figure 7.3. *Modeling of joint flexibility*

• **Example 7.1.** Computation of the elements of the matrices **A** and **Q** for the first three links of the Stäubli RX-90 robot whose geometric parameters are given in Example 3.1.

i) computation of the angular velocities (equation [7.17])

$$^0\boldsymbol{\omega}_0 = \mathbf{0}$$

$$^1\boldsymbol{\omega}_1 = [\,0 \quad 0 \quad \dot{q}_1]^T$$

$$^2\boldsymbol{\omega}_2 = {}^2\mathbf{R}_1\,{}^1\boldsymbol{\omega}_1 + \dot{q}_2\,{}^2\mathbf{a}_2$$

$$= \begin{bmatrix} C2 & 0 & S2 \\ -S2 & 0 & C2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{q}_2 \end{bmatrix} = [S2\,\dot{q}_1 \quad C2\,\dot{q}_1 \quad \dot{q}_2]^T$$

$$^3\boldsymbol{\omega}_3 = {}^3\mathbf{R}_2\,{}^2\boldsymbol{\omega}_2 + \dot{q}_3\,{}^3\mathbf{a}_3$$

$$= \begin{bmatrix} C3 & S3 & 0 \\ -S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S2\,\dot{q}_1 \\ C2\,\dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{q}_3 \end{bmatrix} = [S23\,\dot{q}_1 \quad C23\,\dot{q}_1 \quad \dot{q}_2 + \dot{q}_3]^T$$

ii) computation of the linear velocities (equation [7.18])

$$^0\mathbf{v}_0 = \mathbf{0}$$
$$^1\mathbf{v}_1 = \mathbf{0}$$
$$^1\mathbf{v}_2 = {}^1\mathbf{v}_1 + {}^1\boldsymbol{\omega}_1 \times {}^1\mathbf{P}_2 = \mathbf{0}$$
$$^2\mathbf{v}_2 = \mathbf{0}$$
$$^2\mathbf{v}_3 = {}^2\boldsymbol{\omega}_2 \times {}^2\mathbf{P}_3 = [0 \quad D3\,\dot{q}_2 \quad -C2D3\,\dot{q}_1]^T$$
$$^3\mathbf{v}_3 = {}^3\mathbf{R}_2\,{}^2\mathbf{v}_3 = [S3\,D3\,\dot{q}_2 \quad C3\,D3\,\dot{q}_2 \quad -C2D3\,\dot{q}_1]^T$$

iii) computation of the inertia matrix **A**. With the following general inertial parameters:

$$^j\mathbf{MS}_j = [MX_j \quad MY_j \quad MZ_j]^T$$

$$^j\mathbf{I}_{Oj} = \begin{bmatrix} XX_j & XY_j & XZ_j \\ XY_j & YY_j & YZ_j \\ XZ_j & YZ_j & ZZ_j \end{bmatrix}, \quad \mathbf{I}_a = \begin{bmatrix} I_{a1} & 0 & 0 \\ 0 & I_{a2} & 0 \\ 0 & 0 & I_{a3} \end{bmatrix}$$

we obtain the elements of the robot inertia matrix as:

$$A_{11} = Ia_1 + ZZ_1 + SS2\,XX_2 + 2CS2\,XY_2 + CC2\,YY_2 + SS23\,XX_3 + 2CS23\,XY_3 + CC23\,YY_3 + 2C2\,C23\,D3\,MX_3 - 2C2\,S23\,D3\,MY_3 + CC2\,D3^2\,M_3$$
$$A_{12} = S2\,XZ_2 + C2\,YZ_2 + S23\,XZ_3 + C23\,YZ_3 - S2\,D3\,MZ_3$$
$$A_{13} = S23\,XZ_3 + C23\,YZ_3$$
$$A_{22} = Ia_2 + ZZ_2 + ZZ_3 + 2C3\,D3\,MX_3 - 2S3\,D3\,MY_3 + D3^2\,M_3$$
$$A_{23} = ZZ_3 + C3\,D3\,MX_3 - S3\,D3\,MY_3$$
$$A_{33} = Ia_3 + ZZ_3$$

where $SSj = (\sin\theta_j)^2$, $CCj = (\cos\theta_j)^2$ and $CSj = \cos\theta_j \sin\theta_j$. The elements of the matrix **C** can be computed by equation [7.7];

iv) computation of the gravity forces. Assuming that gravitational acceleration is given as ${}^0\mathbf{g} = [0 \quad 0 \quad G3]^T$, and using equation [7.25], the potential energy is obtained as:

$$U = -G3\,(MZ_1 + S2MX_2 + C2MY_2 + S23MX_3 + C23MY_3 + D3S2M_3)$$

Using equation [7.8] gives:

$$Q_1 = 0$$
$$Q_2 = -G3\,(C2\,MX_2 - S2\,MY_2 + C23\,MX_3 - S23\,MY_3 + D3\,C2\,M_3)$$
$$Q_3 = -G3\,(C23\,MX_3 - S23\,MY_3)$$

7.4. Newton-Euler formulation

7.4.1. *Introduction*

The Newton-Euler equations describing the forces and moments (wrench) acting on the center-of-mass of link j are given as:

$$\mathbf{f}_j = M_j \dot{\mathbf{v}}_{Gj} \qquad [7.36]$$

$$\mathbf{m}_{Gj} = \mathbf{I}_{Gj}\,\dot{\boldsymbol{\omega}}_j + \boldsymbol{\omega}_j \times (\mathbf{I}_{Gj}\,\boldsymbol{\omega}_j) \qquad [7.37]$$

The Newton-Euler algorithm of Luh, Walker and Paul [Luh 80b], which is considered as one of the most efficient algorithms for real time computation of the inverse dynamic model, consists of two recursive computations: forward recursion and backward recursion. The forward recursion, from the base to the terminal link, computes the link velocities and accelerations and consequently the dynamic wrench on each link. The backward recursion, from the terminal link to the base, provides the reaction wrenches on the links and consequently the joint torques.

This method gives the joint torques as defined by equation [7.1] without explicitly computing the matrices **A**, **C** and **Q**. The model obtained is not linear in the inertial parameters because it makes use of M_j, $\mathbf{S}_j$ and $\mathbf{I}_{Gj}$.

7.4.2. *Newton-Euler inverse dynamics linear in the inertial parameters*

In this section, we develop a Newton-Euler algorithm based on the double recursive computations of Luh et al. [Luh 80b], but which uses as inertial parameters the elements of $\mathbf{J}_j$, $\mathbf{MS}_j$ and M_j [Khalil 87b], [Khosla 86]. The dynamic wrench on link j is calculated on O_j and not on the center of gravity G_j. Therefore, the resulting model is linear in the dynamic parameters. This reformulation allows

us to compute the dynamic model in terms of the base inertial parameters (minimum inertial parameters) and to use it for the identification of the dynamic parameters.

The Newton-Euler equations giving the forces and moments of link j at the origin of frame R_j are given as:

$$\mathbf{f}_{tj} = M_j \dot{\mathbf{v}}_j + \dot{\boldsymbol{\omega}}_j \times \mathbf{MS}_j + \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \mathbf{MS}_j) \quad [7.38]$$

$$\mathbf{m}_{tj} = \mathbf{I}_{Oj}\, \dot{\boldsymbol{\omega}}_j + \mathbf{MS}_j \times \dot{\mathbf{v}}_j + \boldsymbol{\omega}_j \times (\mathbf{I}_{Oj}\, \boldsymbol{\omega}_j) \quad [7.39]$$

Using the spatial notation, we can write these equations as:

$$\mathbf{F}_{tj} = \mathbb{I}_j \dot{\mathbf{V}}_j + \begin{bmatrix} \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \mathbf{MS}_j) \\ \boldsymbol{\omega}_j \times (\mathbf{I}_{Oj}\, \boldsymbol{\omega}_j) \end{bmatrix} \quad [7.40]$$

where $\mathbf{F}_{tj} = \begin{bmatrix} \mathbf{f}_{tj} \\ \mathbf{m}_{tj} \end{bmatrix}$, $\dot{\mathbf{V}}_j = \begin{bmatrix} \dot{\mathbf{v}}_j \\ \dot{\boldsymbol{\omega}}_j \end{bmatrix}$ $\mathbb{I}_j$ is the spatial inertia matrix (equation [7.21]).

i) forward recursive computation: to compute $\mathbf{f}_{tj}$ and $\mathbf{m}_{tj}$ for j = 1, …, n, using equations [7.38] and [7.39], we need $\boldsymbol{\omega}_j$, $\dot{\boldsymbol{\omega}}_j$ and $\dot{\mathbf{v}}_j$.. The velocities are given by the recursive equations [7.14] and [7.15] rewritten hereafter as:

$$\boldsymbol{\omega}_j = \boldsymbol{\omega}_{j-1} + \bar{\sigma}_j\, \dot{q}_j\, \mathbf{a}_j \quad [7.41]$$

$$\mathbf{v}_j = \mathbf{v}_{j-1} + \boldsymbol{\omega}_{j-1} \text{ x } \mathbf{L}_j + \sigma_j\, \dot{q}_j\, \mathbf{a}_j \quad [7.42]$$

Differentiating equations [7.41] and [7.42] with respect to time gives:

$$\dot{\boldsymbol{\omega}}_j = \dot{\boldsymbol{\omega}}_{j-1} + \bar{\sigma}_j\, (\ddot{q}_j\, \mathbf{a}_j + \boldsymbol{\omega}_{j-1} \text{ x } \dot{q}_j\, \mathbf{a}_j) \quad [7.43]$$

and

$$\dot{\mathbf{v}}_j = \dot{\mathbf{v}}_{j-1} + \dot{\boldsymbol{\omega}}_{j-1} \text{ x } \mathbf{L}_j + \boldsymbol{\omega}_{j-1} \text{ x } (\boldsymbol{\omega}_{j-1} \text{ x } \mathbf{L}_j + \sigma_j\, \dot{q}_j\, \mathbf{a}_j) + \sigma_j\, (\ddot{q}_j\, \mathbf{a}_j + \boldsymbol{\omega}_{j-1} \text{ x } \dot{q}_j\, \mathbf{a}_j)$$

giving

$$\dot{\mathbf{V}}_j = \dot{\mathbf{V}}_{j-1} + \dot{\boldsymbol{\omega}}_{j-1} \text{ x } \mathbf{L}_j + \boldsymbol{\omega}_{j-1} \text{ x } (\boldsymbol{\omega}_{j-1} \text{ x } \mathbf{L}_j) + \sigma_j\, (\ddot{q}_j\, \mathbf{a}_j + 2\, \boldsymbol{\omega}_{j-1} \text{ x } \dot{q}_j\, \mathbf{a}_j) \quad [7.44]$$

The initial conditions for a robot with a fixed base are $\boldsymbol{\omega}_0 = \mathbf{0}$, $\dot{\boldsymbol{\omega}}_0 = \mathbf{0}$ and and $\dot{\mathbf{v}}_0 = \mathbf{0}$.

ii) backward recursive computation: this is based on writing for each link j, for j = n,…,1, the Newton-Euler equations at the origin O_j, as follows (Figure 7.5):

$$\mathbf{f}_{tj} = \mathbf{f}_j - \mathbf{f}_{j+1} + M_j\, \mathbf{g} - \mathbf{f}_{ej} \quad [7.45]$$

$$\mathbf{m}_{tj} = \mathbf{m}_j - \mathbf{m}_{j+1} - \mathbf{L}_{j+1} \text{ x } \mathbf{f}_{j+1} + \mathbf{MS}_j \text{ x } \mathbf{g} - \mathbf{m}_{ej} \quad [7.46]$$

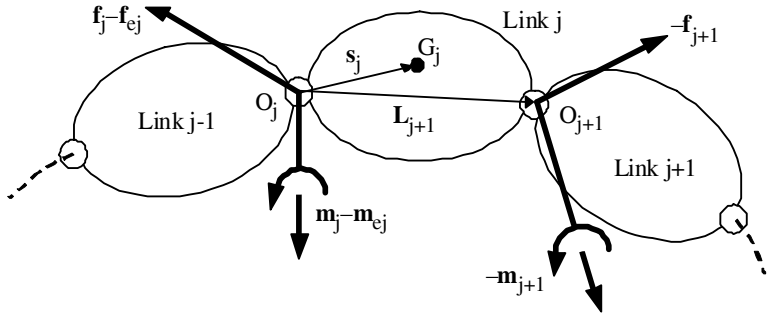

Figure 7.5. *Forces and moments on link j*

We note that $\mathbf{f}_{ej}$ and $\mathbf{m}_{ej}$, which represent the force and moment exerted by link j on the environment, may include contributions from springs, dampers, contact with the environment, etc. Their values are assumed to be known, or at least to be calculated from known quantities.

We can cancel the gravity terms from equations [7.45] and [7.46] and take into account their effects by setting up the initial linear acceleration such that:

$$\dot{\mathbf{v}}_0 = -\mathbf{g} \qquad [7.47]$$

Thus, using equations [7.45] and [7.46], we obtain:

$$\mathbf{f}_j = \mathbf{f}_{tj} + \mathbf{f}_{j+1} + \mathbf{f}_{ej} \qquad [7.48]$$
$$\mathbf{m}_j = \mathbf{m}_{tj} + \mathbf{m}_{j+1} + \mathbf{L}_{j+1} \times \mathbf{f}_{j+1} + \mathbf{m}_{ej} \qquad [7.49]$$

This backward recursive algorithm is initialized by $\mathbf{f}_{n+1} = \mathbf{0}$ and $\mathbf{m}_{n+1} = \mathbf{0}$.

Finally, the joint torque Γ_j can be obtained by projecting $\mathbf{f}_j$ or $\mathbf{m}_j$ on the joint axis, depending whether the joint is prismatic or revolute respectively. We can also consider the friction forces and the rotor inertia as shown in the Lagrange method:

$$\mathbf{\Gamma}_j = (\sigma_j \mathbf{f}_j + \bar{\sigma}_j \mathbf{m}_j)^T \mathbf{a}_j + F_{cj}\, \mathrm{sign}(\dot{q}_j) + F_{vj}\, \dot{q}_j + Ia_j\, \ddot{q}_j \qquad [7.50]$$

From equations [7.48], [7.49] and [7.50], we deduce that $\mathbf{\Gamma}_j$ is a function of the inertial parameters of links j, j+1, …, n. This property has been outlined in § 7.3.3.3.

7.4.3. *Practical form of the Newton-Euler algorithm*

Since $\mathbf{I}_{Oj}$ and $\mathbf{MS}_j$ are constants when referred to their own link coordinates, the Newton-Euler algorithm can be efficiently computed by referring the velocities, accelerations, forces and moments to the local link coordinate system [Luh 80b]. The forward recursive equations become, for j = 1, ..., n:

$$^{j}\boldsymbol{\omega}_{j-1} = {}^{j}\mathbf{R}_{j-1}\,{}^{j-1}\boldsymbol{\omega}_{j-1} \qquad [7.51]$$

$$^{j}\boldsymbol{\omega}_{j} = {}^{j}\boldsymbol{\omega}_{j-1} + \bar{\sigma}_j\,\dot{q}_j\,{}^{j}\mathbf{a}_j \qquad [7.52]$$

$$^{j}\dot{\boldsymbol{\omega}}_{j} = {}^{j}\mathbf{R}_{j-1}\,{}^{j-1}\dot{\boldsymbol{\omega}}_{j-1} + \bar{\sigma}_j\,(\ddot{q}_j\,{}^{j}\mathbf{a}_j + {}^{j}\boldsymbol{\omega}_{j-1} \times \dot{q}_j\,{}^{j}\mathbf{a}_j) \qquad [7.53]$$

$$^{j}\dot{\mathbf{v}}_{j} = {}^{j}\mathbf{R}_{j-1}\,({}^{j-1}\dot{\mathbf{v}}_{j-1} + {}^{j-1}\mathbf{U}_{j-1}\,{}^{j-1}\mathbf{P}_j) + \sigma_j\,(\ddot{q}_j\,{}^{j}\mathbf{a}_j + 2\,{}^{j}\boldsymbol{\omega}_{j-1} \times \dot{q}_j\,{}^{j}\mathbf{a}_j) \qquad [7.54]$$

$$^{j}\mathbf{f}_{tj} = M_j\,{}^{j}\dot{\mathbf{v}}_j + {}^{j}\mathbf{U}_j\,{}^{j}\mathbf{MS}_j \qquad [7.55]$$

$$^{j}\mathbf{m}_{tj} = {}^{j}\mathbf{I}_{Oj}\,{}^{j}\dot{\boldsymbol{\omega}}_j + {}^{j}\mathbf{MS}_j \times {}^{j}\dot{\mathbf{v}}_j + {}^{j}\boldsymbol{\omega}_j \times ({}^{j}\mathbf{I}_{Oj}\,{}^{j}\boldsymbol{\omega}_j) \qquad [7.56]$$

$$^{j}\mathbf{U}_j = {}^{j}\hat{\dot{\boldsymbol{\omega}}}_j + {}^{j}\hat{\boldsymbol{\omega}}_j\,{}^{j}\hat{\boldsymbol{\omega}}_j \qquad [7.57]$$

For a stationary base, the initial conditions are $\boldsymbol{\omega}_0 = \mathbf{0}$, $\dot{\boldsymbol{\omega}}_0 = \mathbf{0}$ and $\dot{\mathbf{v}}_0 = -\mathbf{g}$.

The use of $^{j}\mathbf{U}_j$ saves 21n multiplications and 6n additions in the computation of the inverse dynamic model of a general robot [Kleinfinger 86a].

The backward recursive equations, for j = n, ..., 1, are:

$$^{j}\mathbf{f}_j = {}^{j}\mathbf{f}_{tj} + {}^{j}\mathbf{f}_{j+1} + {}^{j}\mathbf{f}_{ej} \qquad [7.58]$$

$$^{j-1}\mathbf{f}_j = {}^{j-1}\mathbf{R}_j\,{}^{j}\mathbf{f}_j \qquad [7.59]$$

$$^{j}\mathbf{m}_j = {}^{j}\mathbf{m}_{tj} + {}^{j}\mathbf{R}_{j+1}\,{}^{j+1}\mathbf{m}_{j+1} + {}^{j}\mathbf{P}_{j+1} \times {}^{j}\mathbf{f}_{j+1} + {}^{j}\mathbf{m}_{ej} \qquad [7.60]$$

$$\boldsymbol{\Gamma}_j = (\sigma_j\,{}^{j}\mathbf{f}_j + \bar{\sigma}_j\,{}^{j}\mathbf{m}_j)^T\,{}^{j}\mathbf{a}_j + F_{sj}\,\mathrm{sign}(\dot{q}_j) + F_{vj}\,\dot{q}_j + Ia_j\,\ddot{q}_j \qquad [7.61]$$

The previous algorithm can be numerically programmed for a general serial robot. Its computational complexity is $O(n)$, which means that the number of operations is linear in the number of degrees of freedom. However, as we will see in the next section, the use of the base inertial parameters in a customized symbolic algorithm considerably reduces the number of operations of the dynamic model.

NOTES:-

- the symbol $^{j}\dot{\mathbf{v}}_j$ means $^{j}\mathbf{R}_0\,{}^{0}\dot{\mathbf{v}}_j$, and not the time derivative of $^{j}\mathbf{v}_j$. Using the fact that:

  $^{j}\mathbf{v}_j = {}^{j}\mathbf{R}_0\,{}^{0}\mathbf{v}_j$ and $^{0}\dot{\mathbf{R}}_j = {}^{0}\hat{\boldsymbol{\omega}}_j\,{}^{0}\mathbf{R}_j$

  we can deduce that

$$^{j}\dot{\mathbf{v}}_j = \frac{d}{dt}\,^{j}\mathbf{v}_j + {}^{j}\hat{\boldsymbol{\omega}}_j\,^{j}\mathbf{v}_j$$

since .

$$\frac{d}{dt}\,^{j}\mathbf{v}_j = -{}^{j}\mathbf{R}_0\,^{0}\hat{\boldsymbol{\omega}}_j\,^{0}\mathbf{v}_j + {}^{j}\mathbf{R}_0\,^{0}\dot{\mathbf{v}}_j = -{}^{j}\mathbf{R}_0\,^{0}\hat{\boldsymbol{\omega}}_j\,^{0}\mathbf{R}_j\,^{j}\mathbf{R}_0\,^{0}\mathbf{v}_j + {}^{j}\mathbf{R}_0\,^{0}\dot{\mathbf{v}}_j = -{}^{j}\hat{\boldsymbol{\omega}}_j\,^{j}\mathbf{v}_j + {}^{j}\dot{\mathbf{v}}_j$$

On the contrary, $^{j}\dot{\boldsymbol{\omega}}_j = \frac{d}{dt}\,^{j}\boldsymbol{\omega}_j = {}^{j}\mathbf{R}_0\,^{0}\dot{\boldsymbol{\omega}}_j$.

- We use σ_j =2 to define a fixed frame R_j with resoect to j-1. In this case $\dot{q}_j$ and $\ddot{q}_j$ are equal to zero, the torque equation [9.94] has no physical meaning and should not be calculated, whereas the velocity and acceleration equations can be used after eliminating the terms multiplied by σ_j or $\bar{\sigma}_j$.
- If link 0, the base, is mobile (not fixed) with r dof (car, walking robot, or mobile manipulator), we can represent its motion using r virtual joints (Lagrangian variables). The inertial parameters of the first r-1 virtual links are taken equal to zero whereas the rth link parameters are equal to those of the base. This case is efficient if the motion is in a plane. The mobile motion can be represented by two prismatic joints followed by a rotational joint.

- If link 0, the base is mobile, it is in general more efficient to represent its motion using Eulerian variables (linear and rotational Cartesian velocities and accelerations). The equations of the base will be represented in Eulerian form.

7.5. Real time computation of the inverse dynamic model

7.5.1. *Introduction*

Much work has been focused on the problem of computational efficiency of the inverse dynamic model of robots in order to realize real time dynamic control. This objective is now recognized as being attained (Table 7.5).

Before describing our method, which is based on a customized symbolic Newton-Euler formulation linear in the inertial parameters [Khalil 87b], [Kleinfinger 86a], we review the main approaches presented in the literature.

The Lagrangian formulation of Uicker and Kahn [Uicker 69], [Kahn 69] served as a standard robot dynamics formulation for over a decade. In this form, the complexity of the equations precluded the real time computation of the inverse dynamic model. For example, for a six degree-of-freedom robot, this formulation requires 66271 multiplications and 51548 additions [Hollerbach 80]. This led researchers to investigate either simplification or tabulation approaches to achieve real time implementation.

The first approach towards simplification is to neglect the Coriolis and centrifugal terms with the assumption that they are not significant except at high speeds [Paul 72], [Bejczy 74]. Unfortunately, this belief is not valid: firstly, Luh et al. [Luh 80b] have shown that such simplifications leads to large errors not only in the value of the computed joint torques but also in its sign; later, Hollerbach [Hollerbach 84a] has shown that the velocity terms have the same significance relative to the acceleration terms whatever the velocity.

An alternative simplification approach has been proposed by Bejczy [Bejczy 79]. This approach is based on an exhaustive term-by-term analysis of the elements A_{ij}, $C_{i,jk}$ and Q_i so that the most significant terms are only retained. A similar procedure has been utilized by Armstrong et al. [Armstrong 86] who also proposed computing the elements A_{ij}, $C_{i,jk}$ and Q_i with a low frequency rate with respect to that of the servo rate. Using such an analysis for a six degree-of-freedom robot becomes very laborious and tedious.

In the tabulation approach, some terms of the dynamic equations are precomputed and tabulated. The combination of a look-up table with reduced analytical computations renders them feasible in real time. Two methods based on a trade-off between memory space and computation time have been investigated by Raibert [Raibert 77]. In the first method *SSM (State Space Model)*, the table was carried out as a function of the joint positions and velocities ($\mathbf{q}$ and $\dot{\mathbf{q}}$), but the required memory was too big to consider in real applications at that time. In the *Configuration Space Method (CSM)*, the table is computed as a function of the joint positions. Another technique, proposed by Aldon [Aldon 82] consists of tabulating the elements A_{ij} and Q_i and of calculating the elements $C_{i,jk}$ on-line in terms of the A_{ij} elements. This method considerably reduces the required memory but increases the number of on-line operations, which becomes proportional to n^3. We note that the tabulated elements are functions of the load inertial parameters, which means making a table for each load.

Luh et al. [Luh 80a] proposed to determine the inverse dynamic model using a Newton-Euler formulation (§ 7.5). The complexity of this method is $O(n)$. This method has proved the importance of the recursive computations and the organization of the different steps of the dynamic algorithm. Therefore, other researchers tried to improve the existing Lagrange formulations by introducing recursive computations. For example, Hollerbach [Hollerbach 80] proposed a new recursive Lagrange formulation with complexity $O(n)$, whereas Megahed [Megahed 84] developed a new recursive computational procedure for the Lagrange method of Uicker and Kahn. However, these methods are less efficient than the Newton-Euler formulation of Luh et al. [Luh 80a].

More recently, researchers investigated alternative formulations [Kane 83], [Vukobratovic 85], [Renaud 85], [Kazerounian 86], but the recursive Newton-Euler proved to be computationally more efficient.

The most efficient models proposed until now are based on a customized symbolic Newton-Euler formulation that takes into account the particularity of the

geometric and inertial parameters of each robot [Kanade 84], [Khalil 85a], [Khalil 87b], [Renaud 87]. Moreover, the use of the base inertial parameters in this algorithm reduces the computational cost by about 25%. We note that the number of operations for this method is even less than that of the tabulated CSM method.

Table 7.5. *Computational cost of the inverse dynamic modeling methods*

Method	Robot	General case		2RP(3R) Stanford		R(2P)(3R) TH8		6R Stäubli RX-90	
	Operations	**n ddl**	**n=6**	**General**	**Simplified***	**General**	**Simplified***	**General**	**Simplified***
Raibert 78**	Mult.	n^3+2n^2	288	288	288	288	288	288	288
	Add.	3^n+n^2	252	252	252	252	252	252	252
Luh 80b	Mult.	137n-22	800	800	800	800	800	800	800
	Add.	101n-11	595	595	595	595	595	595	595
Hollerbach 80**	Mult.	412n-277	2195	2195	2195	2195	2195	2195	2195
	Add.	320n-201	1719	1719	1719	1719	1719	1719	1719
Kane 83**	Mult.	?	?	646	?	?	?	?	?
	Add.	?	?	394	?	?	?	?	?
Vukobratovic 85**	Mult.	?	?	?	>372	?	>193	?	?
	Add.	?	?	?	>167	?	>80	?	?
Renaud 85**	Mult.	?	?	?	?	?	?	?	368
	Add.	?	?	?	?	?	?	?	271
Presented method Khalil [87b]	Mult.	92n-127	425	240	142	175	104	253	159
	Add.	81n-117	369	227	99	162	72	238	113

? the number of operations is not given.
* the matrix IOj is diagonal and two components of the first moments are zero.
** forces and moments exerted by the end-effector on the environment are not considered.
> the given number of operations corresponds to the computation of the elements of **A**, $C_{i,jk}$ and **Q**.

Before closing this section, it is worth noting the formidable technological progress in the field of computer processors, to the point that the dynamic model can be calculated at control rate with standard personal computers (Chapter 14).

7.5.2. *Customization of the Newton-Euler formulation*

The recursive Newton-Euler formulation of robot dynamics has become a standard algorithm for real time control and simulation (§ 7.5.3). To increase the efficiency of the Newton-Euler algorithm, we generate a customized symbolic model for each specific robot and utilize the base dynamic parameters. To obtain this model, we expand the recursive equations to transform them into scalar equations without incorporating loop computations. The elements of a vector or a matrix containing at least one mathematical operation are replaced by an intermediate variable. This variable is written in the output file, which contains the customized model [Kanade 84], [Khalil 85a]. The elements that do not contain any operation are not modified. We propagate the obtained vectors and matrices in the subsequent equations. Consequently, customizing eliminates multiplications by one (and minus one) and zero, and additions with zero. A good choice of the intermediate variables allows us to avoid redundant computations. In the end, the dynamic model is obtained as a set of intermediate variables. Those that have no effect on the desired output, the joint torques in the case of inverse dynamics, can be eliminated by scanning the intermediate variables from the end to the beginning.

The customization technique allows us to reduce the computational load for a general robot, but this reduction is larger when carried out for a specific robot [Kleinfinger 86a]. The computational efficiency in customization is obtained at the cost of a software symbolic iterative structure [Khalil 97] and a relatively increased program memory requirement.

• **Example 7.4.** To illustrate how to generate a customized symbolic model, we develop in this example the computation of the link angular velocities ${}^{j}\boldsymbol{\omega}_j$ for the Stäubli RX-90 robot. The computation of the orientation matrices ${}^{j-1}\mathbf{A}_j$ (Example 3.3) generates the 12 sinus and cosinus intermediate variables:

$$Sj = \sin(q_j)$$
$$Cj = \cos(q_j) \qquad \text{for } j = 1, \ldots, 6$$

The computation of the angular velocities for j = 1, …, 6 is given as:

$${}^{1}\boldsymbol{\omega}_1 = \begin{bmatrix} 0 \\ 0 \\ QP1 \end{bmatrix}$$

Computation of ${}^{1}\boldsymbol{\omega}_1$ does not generate any intermediate variable.

$$ {}^{2}\boldsymbol{\omega}_1 = \begin{bmatrix} \text{S2*QP1} \\ \text{C2*QP1} \\ 0 \end{bmatrix} = \begin{bmatrix} \text{WI12} \\ \text{WI22} \\ 0 \end{bmatrix} $$

Computation of ${}^{2}\boldsymbol{\omega}_2$ generates the following intermediate variables:

WI12 = S2*QP1
WI22 = C2*QP1

In the following, the vector ${}^{2}\boldsymbol{\omega}_2$ is set as:

$$ {}^{2}\boldsymbol{\omega}_2 = \begin{bmatrix} \text{WI12} \\ \text{WI22} \\ \text{QP2} \end{bmatrix} $$

Continuing the recursive computation leads to:

$$ {}^{3}\boldsymbol{\omega}_2 = \begin{bmatrix} \text{C3*WI12 + S3*WI22} \\ \text{–S3*WI12 + C3*WI22} \\ \text{QP2} \end{bmatrix} = \begin{bmatrix} \text{WI13} \\ \text{WI23} \\ \text{QP2} \end{bmatrix} $$

$$ {}^{3}\boldsymbol{\omega}_3 = \begin{bmatrix} \text{WI13} \\ \text{WI23} \\ \text{QP2 + QP3} \end{bmatrix} = \begin{bmatrix} \text{WI13} \\ \text{WI23} \\ \text{W33} \end{bmatrix} $$

$$ {}^{4}\boldsymbol{\omega}_3 = \begin{bmatrix} \text{C4*WI13 – S4*W33} \\ \text{–S4*WI12 – C4*W33} \\ \text{WI23} \end{bmatrix} = \begin{bmatrix} \text{WI14} \\ \text{WI24} \\ \text{WI23} \end{bmatrix} $$

$$ {}^{4}\boldsymbol{\omega}_4 = \begin{bmatrix} \text{WI14} \\ \text{WI24} \\ \text{WI23 + QP4} \end{bmatrix} = \begin{bmatrix} \text{WI14} \\ \text{WI24} \\ \text{W34} \end{bmatrix} $$

$$ {}^{5}\boldsymbol{\omega}_4 = \begin{bmatrix} \text{C5*WI14 + S5*W34} \\ \text{–S5*WI14 + C5*W34} \\ \text{–WI24} \end{bmatrix} = \begin{bmatrix} \text{WI15} \\ \text{WI25} \\ \text{–WI24} \end{bmatrix} $$

$$ {}^{5}\boldsymbol{\omega}_5 = \begin{bmatrix} \text{WI15} \\ \text{WI25} \\ \text{–WI24 + QP5} \end{bmatrix} = \begin{bmatrix} \text{WI15} \\ \text{WI25} \\ \text{W35} \end{bmatrix} $$

$$^{6}\omega_5 = \begin{bmatrix} C6*WI15 - S6*W35 \\ -S6*WI15 - C6*W35 \\ WI25 \end{bmatrix} = \begin{bmatrix} WI16 \\ WI26 \\ WI25 \end{bmatrix}$$

$$^{6}\omega_6 = \begin{bmatrix} WI16 \\ WI26 \\ WI25 + QP6 \end{bmatrix} = \begin{bmatrix} WI16 \\ WI26 \\ W36 \end{bmatrix}$$

Finally, the computation of $^{j}\omega_j$, for j = 1, …, 6, requires the following intermediate variables: WI12, WI22, WI13, WI23, W33, WI14, WI24, W34, WI15, WI25, W35, WI16, WI26 and W36, in addition to the variables Sj and Cj for j = 2, …, 6. The variables S1 and C1 can be eliminated because they have no effect on the angular velocities.

7.5.3. *Utilization of the base inertial parameters*

The dynamic model can be obtained in a reduced number of inertial parameters called the base inertial parameters. These parameters are obtained from the standard parameters after eliminating those who have no effect on the dynamic model and by grouping some parameters together. It is obvious that the use of the base inertial parameters in a customized Newton-Euler formulation that is linear in the inertial parameters will reduce the number of operations because the parameters that have no effect on the model or have been grouped are set equal to zero. Practically, the number of operations of the inverse dynamic model when using the base inertial parameters for a general n revolute degree-of-freedom robot is 92n – 127 multiplications and 81n – 117 additions (n > 2), which gives 425 multiplications and 369 additions for n = 6. By general robot, we mean:

- the geometric parameters r_1, d_1, α_1 and r_n are zero (this assumption holds for any robot);
- the other geometric parameters, all the inertial parameters, and the forces and moments exerted by the terminal link on the environment can have an arbitrary real value;
- the friction forces are assumed to be negligible, otherwise, with a Coulomb and viscous friction model, we add n multiplications, 2n additions, and n sign functions.

Table 7.6. shows the computational complexity of the inverse dynamic model for the Stäubli RX-90 robot using the customized Newton-Euler formulation. In Appendix 7, we give the dynamic model of the Stäubli RX-90 robot when using the base inertial parameters of Table 7.4, which takes into account the symmetry of the

links. The corresponding computational cost is 160 multiplications and 113 additions.

Table 7.6. *Computational complexity of the inverse dynamic model for the Stäubli RX-90 robot*

	Inertial parameters	**Multiplications**	**Additions**
General inertial parameters	Standard parameters	294	283
	Base parameters	253	238
Simplified inertial parameters	Standard parameters	202	153
	Base parameters	160	113

7.6. Direct dynamic model

The computation of the direct dynamic model is employed to carry out simulations for the purpose of testing the robot performances and studying the synthesis of the control laws. During simulation, the dynamic equations are solved for the joint accelerations given the input torques and the current state of the robot (joint positions and velocities). Through integration of the joint accelerations, the robot trajectory is then determined. Although the simulation may be carried out off-line, it is interesting to have an efficient direct dynamic model to reduce the simulation time. In this section, we consider two methods: the first is based on using a specialized Newton-Euler inverse dynamic model and needs to compute the inverse of the inertia matrix **A** of the robot; the second method is based on a recursive Newton-Euler algorithm that does not explicitly use the matrix **A** and has a computational cost that varies linearly with the number of degrees of freedom of the robot.

7.6.1. *Using the inverse dynamic model to solve the direct dynamic problem*

From equation [7.6], we can express the direct dynamic problem as the solution of the joint accelerations from the following equation:

$$\mathbf{A}\,\ddot{\mathbf{q}} = [\boldsymbol{\Gamma} - \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})] \qquad [7.62]$$

where **H** contains the Coriolis, centrifugal, gravity, friction and external torques:

$$\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{Q} + \mathbf{diag}(\dot{\mathbf{q}})\mathbf{F}_v + \mathbf{diag}(\mathrm{sign}(\dot{\mathbf{q}}))\mathbf{F}_c + \mathbf{J}^T\,\mathbb{f}_{en}$$

Although in practice we do not explicitly calculate the inverse of the matrix **A**, the solution of equation [7.95] is generally denoted by:

$$\ddot{\mathbf{q}} = \mathbf{A}^{-1}\,[\boldsymbol{\Gamma} - \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})] \qquad [7.63]$$

The computation of the direct dynamics can be broken down into three steps: the calculation of $\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})$, the calculation of **A**, and the solution of the linear equation [7.62] for $\ddot{\mathbf{q}}$.

The computational complexity of the first step is minimized by the use of a specialized version of the inverse dynamics algorithm in which the desired joint accelerations are zero [Walker 82]. By comparing equations [7.1] and [7.63], we deduce that $\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})$ is equal to $\boldsymbol{\Gamma}$ if $\ddot{\mathbf{q}} = \mathbf{0}$.

The inertia matrix can also be calculated one column at a time, using Newton-Euler inverse dynamic model [Walker 82]. From relation [7.62], we deduce that the ith column of **A** is equal to $\boldsymbol{\Gamma}$ if:

$$\ddot{\mathbf{q}} = \mathbf{u}_i,\ \dot{\mathbf{q}} = \mathbf{0},\ \mathbf{g} = \mathbf{0},\ \mathbf{F}_c = \mathbf{0}\ (\mathbf{f}_{ej} = \mathbf{0},\ \mathbf{m}_{ej} = \mathbf{0} \quad \text{for } j = 1,\ldots, n) \qquad [7.64]$$

where $\mathbf{u}_i$ is an (nx1) unit vector with 1 in the ith row and zeros elsewhere. Iterating the procedure for i = 1,…, n leads to the construction of the entire inertia matrix.

To reduce the computational complexity of this algorithm, we can make use of the base inertial parameters and the customized symbolic techniques. Moreover, we can take advantage of the fact that the inertia matrix **A** is symmetric. A more efficient procedure for computing the inertia matrix using the concept of composite links is given in Appendix 8. Alternative efficient approaches for computing the inertia matrix based on the Lagrange formulation are proposed in [Megahed 82], [Renaud 85].

NOTE.– The nonlinear state equation of a robot follows from relation [7.62] as:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{A}^{-1}\,[\boldsymbol{\Gamma} - \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})] \end{bmatrix} \qquad [7.65]$$

and the output equation is written as:

$$\mathbf{y} = \mathbf{q} \ \text{ or } \ \mathbf{y} = \mathbf{X}(\mathbf{q}) \qquad [7.66]$$

In this formulation, the state variables are given by $[\mathbf{q}^T \ \ \dot{\mathbf{q}}^T]^T$, the equation $\mathbf{y} = \mathbf{q}$ gives the output vector in the joint space, and $\mathbf{y} = \mathbf{X}(\mathbf{q})$ denotes the coordinates of the end-effector frame in the task space.

7.6.2. *Recursive computation of the direct dynamic model*

This method is based on the recursive Newton-Euler equations and does not use explicitly the inertia matrix of the robot [Armstrong 79], [Featherstone 83b], [Brandl 86]. The joint accelerations are obtained as a result of three recursive computations:

7.6.2.1 Algorithm of computation of the direct dynamic model

i) first forward recursive computations for j = 1, …, n: in this step, we compute the screw transformation matrices ${}^{j}\mathbf{S}_{j-1}$, the link angular velocities ${}^{j}\boldsymbol{\omega}_j$ as well as ${}^{j}\boldsymbol{\gamma}_j$ and ${}^{j}\boldsymbol{\beta}_j$ vectors, which represent the link accelerations and the link wrenches respectively when $\ddot{\mathbf{q}} = \mathbf{0}$;

Using the *screw notation* and by combining equations [7.54] and [7.53], which give ${}^{j}\dot{\mathbf{V}}_j$ and ${}^{j}\dot{\boldsymbol{\omega}}_j$, we obtain:

$$ {}^{j}\dot{\mathbf{V}}_j = {}^{j}\mathbf{S}_{j-1}\,{}^{j-1}\dot{\mathbf{V}}_{j-1} + \ddot{q}_j\,{}^{j}\mathbb{a}_j + {}^{j}\boldsymbol{\gamma}_j \qquad [7.67] $$

where ${}^{j}\mathbb{a}_j$ is defined by equation [7.23b], and:

$$ {}^{j}\boldsymbol{\gamma}_j = \begin{bmatrix} {}^{j}\mathbf{R}_{j-1}[{}^{j-1}\boldsymbol{\omega}_{j-1} \times ({}^{j-1}\boldsymbol{\omega}_{j-1} \times {}^{j-1}\mathbf{P}_j)] + 2\sigma_j\,({}^{j}\boldsymbol{\omega}_{j-1} \times \dot{q}_j\,{}^{j}\mathbf{a}_j) \\ \bar{\sigma}_j\,{}^{j}\boldsymbol{\omega}_{j-1} \times \dot{q}_j\,{}^{j}\mathbf{a}_j \end{bmatrix} \qquad [7.68] $$

Equations [7.88], [7.89], [7.91] and [7.93], which represent the equilibrium equations of link j, can be combined as:

$$ {}^{j}\mathbb{I}_j\,{}^{j}\dot{\mathbf{V}}_j = {}^{j}\mathbf{F}_j - {}^{j+1}\mathbf{S}_j^{T}\,{}^{j+1}\mathbf{F}_{j+1} + {}^{j}\boldsymbol{\beta}_j \qquad [7.69] $$

where:

$$ {}^{j}\boldsymbol{\beta}_j = -{}^{j}\mathbf{F}_{ej} - \begin{bmatrix} {}^{j}\boldsymbol{\omega}_j \times ({}^{j}\boldsymbol{\omega}_j \times {}^{j}\mathbf{M}\mathbf{S}_j) \\ {}^{j}\boldsymbol{\omega}_j \times ({}^{j}\mathbf{I}_{Oj}\,{}^{j}\boldsymbol{\omega}_j) \end{bmatrix} \qquad [7.70] $$

In equation [7.69], we use equation [2.34] to transform the dynamic wrench from frame R_{j+1} to frame R_j.

ii) backward recursive computations for j = n, ..., 1: in this step we calculate the elements H_j, ${}^{j}\mathbb{J}_j^*$, ${}^{j}\boldsymbol{\beta}_j^*$, ${}^{j}\mathbb{K}_j$, ${}^{j}\boldsymbol{\alpha}_j$ which express $\ddot{q}_j$ and $\mathbf{F}_j$ in terms of ${}^{j-1}\dot{\mathbf{V}}_{j-1}$ in the third recursive equations. These equations will be demonstrated in the next section.

For j = n, ..., 1, compute:

$$H_j = ({}^{j}\mathbb{a}_j^T\, {}^{j}\mathbb{I}_j^*\, {}^{j}\mathbb{a}_j + Ia_j) \qquad [7.71]$$

$${}^{j}\mathbf{K}_j = {}^{j}\mathbb{I}_j^* - {}^{j}\mathbb{I}_j^*\, {}^{j}\mathbb{a}_j\, H_j^{-1}\, {}^{j}\mathbb{a}_j^T\, {}^{j}\mathbb{I}_j^* \qquad [7.72]$$

$${}^{j}\boldsymbol{\alpha}_j = {}^{j}\mathbf{K}_j\, {}^{j}\boldsymbol{\gamma}_j + {}^{j}\mathbb{I}_j^*\, {}^{j}\mathbb{a}_j\, H_j^{-1}\, (\tau_j + {}^{j}\mathbb{a}_j^T\, {}^{j}\boldsymbol{\beta}_j^*) - {}^{j}\boldsymbol{\beta}_j^* \qquad [7.73]$$

where (see equation 7.82)

$$\tau_j = \Gamma_j - F_{sj}\, \mathrm{sign}(\dot{q}_j) - F_{vj}\, \dot{q}_j$$

If j $\neq$ 1, calculate also:

$${}^{j-1}\boldsymbol{\beta}_{j-1}^* = {}^{j-1}\boldsymbol{\beta}_{j-1}^* - {}^{j}\mathbf{S}_{j-1}^T\, {}^{j}\boldsymbol{\alpha}_j \qquad [7.74]$$

$${}^{j-1}\mathbb{I}_{j-1}^* = {}^{j-1}\mathbb{I}_{j-1}^* + {}^{j}\mathbf{S}_{j-1}^T\, {}^{j}\mathbf{K}_j\, {}^{j}\mathbf{S}_{j-1} \qquad [7.75]$$

Note that these equations are initialized by ${}^{j}\mathbb{I}_j^* = {}^{j}\mathbb{I}_j$ and ${}^{j}\boldsymbol{\beta}_j^* = {}^{j}\boldsymbol{\beta}_j$;

iii) second forward recursive computations. Since ${}^{0}\dot{\mathbf{V}}_0$ is composed of the linear and angular accelerations of the base that are assumed to be known ($\dot{\mathbf{v}}_0 = -\mathbf{g}$, $\dot{\boldsymbol{\omega}}_0 = \mathbf{0}$ for fixed base), the third recursive computation allows us to compute $\ddot{q}_j$, ${}^{j}\dot{\mathbf{v}}_j$ and ${}^{j}\mathbf{F}_j$ (if needed) for j = 1… n. as follows :

$${}^{j}\dot{\mathbf{V}}_{j-1,j} = {}^{j}\mathbf{S}_{j-1}\, {}^{j-1}\dot{\mathbf{V}}_{j-1} \qquad [7.76]$$

$$\ddot{q}_j = H_j^{-1}\, [-{}^{j}\mathbb{a}_j^T\, {}^{j}\mathbb{I}_j^*\, ({}^{j}\dot{\mathbf{V}}_{j-1,j} + {}^{j}\boldsymbol{\gamma}_j) + \tau_j + {}^{j}\mathbb{a}_j^T\, {}^{j}\boldsymbol{\beta}_j^*] \qquad [7.77]$$

$${}^{j}\dot{\mathbf{V}}_j = {}^{j}\dot{\mathbf{V}}_{j-1,j} + {}^{j}\mathbb{a}_j\, \ddot{q}_j + {}^{j}\boldsymbol{\gamma}_j \qquad [7.78]$$

$${}^{j}\mathbf{F}_j = \begin{bmatrix} {}^{j}\mathbf{f}_j \\ {}^{j}\mathbf{m}_j \end{bmatrix} = {}^{j}\mathbb{I}_j^*\, {}^{j}\dot{\mathbf{V}}_{j-} - {}^{j}\boldsymbol{\beta}_j^* \qquad [7.79]$$

where ${}^{j}\dot{\mathbf{V}}_{j-1,j}$ means that the origin is transferred to Oj.

NOTES.–

- to reduce the number of operations of this algorithm, we can make use of the base inertial parameters and the customized symbolic technique. Thereby, the number of operations of the direct dynamic model for the Stäubli RX-90 robot is 889 multiplications and 653 additions [Khalil 97]. In the case of the use of simplified inertial parameters (Table 7.4), the computational cost becomes 637 multiplications and 423 additions;
- the computational complexity of this method is $O(n)$, while the method requiring the inverse of the robot inertia matrix is of complexity $O(n^3)$;
- from the numerical point of view, this method is more stable than the method requiring the inverse of the robot inertia matrix [Cloutier 95].

7.6.2.2 Demonstration of relations [7.67] and [7.68]

To illustrate the equations required, we detail the case when $j = n$ and $j = n - 1$. By combining equations [7.67] and [7.69] for $j = n$, and since ${}^{n+1}\mathbf{F}_{n+1} = \mathbf{0}$, we obtain:

$$ {}^{n}\mathbb{I}_n \left({}^{n}\mathbf{S}_{n-1}\, {}^{n-1}\dot{\mathbf{V}}_{n-1} + \ddot{q}_n\, {}^{n}\mathbb{a}_n + {}^{n}\boldsymbol{\gamma}_n \right) = {}^{n}\mathbf{F}_n + {}^{n}\boldsymbol{\beta}_n \qquad [7.80] $$

Since:

$$ {}^{j}\mathbb{a}_j^{T}\, {}^{j}\mathbf{F}_j = \tau_j - Ia_j\, \ddot{q}_j \qquad [7.81] $$

$$ \tau_j = \Gamma_j - F_{sj}\, \mathrm{sign}(\dot{q}_j) - F_{vj}\, \dot{q}_j \qquad [7.82] $$

multiplying equation [7.71] by ${}^{n}\mathbb{a}_n^{T}$ and using equation [7.72], we deduce the joint acceleration of joint n:

$$ \ddot{q}_n = H_n^{-1} \left(- {}^{n}\mathbb{a}_n^{T}\, {}^{n}\mathbb{I}_n \left({}^{n}\mathbf{S}_{n-1}\, {}^{n-1}\dot{\mathbf{V}}_{n-1} + {}^{n}\boldsymbol{\gamma}_n \right) + \tau_n + {}^{n}\mathbb{a}_n^{T}\, {}^{n}\boldsymbol{\beta}_n \right) \qquad [7.83] $$

where H_n is a scalar given as:

$$ H_n = \left({}^{n}\mathbb{a}_n^{T}\, {}^{n}\mathbb{I}_n\, {}^{n}\mathbb{a}_n + Ia_n \right) \qquad [7.84] $$

Substituting for $\ddot{q}_n$ from equation [7.83] into equation [7.82], we obtain the dynamic wrench ${}^{n}\mathbf{F}_n$ as:

$$ {}^{n}\mathbf{F}_n = \begin{bmatrix} {}^{n}\mathbf{f}_n \\ {}^{n}\mathbf{m}_n \end{bmatrix} = {}^{n}\mathbf{K}_n \, {}^{n}\mathbf{S}_{n-1} \, {}^{n-1}\dot{\mathbf{V}}_{n-1} + {}^{n}\boldsymbol{\alpha}_n \qquad [7.85] $$

where:

$$ {}^{n}\mathbf{K}_n = {}^{n}\mathbb{I}_n - {}^{n}\mathbb{I}_n \, {}^{n}\mathbb{a}_n \, H_n^{-1} \, {}^{n}\mathbb{a}_n^{T} \, {}^{n}\mathbb{I}_n \qquad [7.86] $$

$$ {}^{n}\boldsymbol{\alpha}_n = {}^{n}\mathbf{K}_n \, {}^{n}\boldsymbol{\gamma}_n + {}^{n}\mathbb{I}_n \, {}^{n}\mathbb{a}_n \, H_n^{-1} \, (\tau_n + {}^{n}\mathbb{a}_n^{T} \, {}^{n}\boldsymbol{\beta}_n) - {}^{n}\boldsymbol{\beta}_n \qquad [7.87] $$

We now have $\ddot{q}_n$ and ${}^{n}\mathbf{F}_n$ in terms of ${}^{n-1}\dot{\mathbf{V}}_{n-1}$. Iterating the procedure for $j = n-1$, we obtain, from equation [7.79]:

$$ {}^{n-1}\mathbb{I}_{n-1} \, {}^{n-1}\dot{\mathbf{V}}_{n-1} = {}^{n-1}\mathbf{F}_{n-1} + {}^{n}\mathbf{S}_{n-1}^{T} \, {}^{n}\mathbf{F}_n + {}^{n-1}\boldsymbol{\beta}_{n-1} \qquad [7.88] $$

which can be rewritten using equation [7.67] as:

$$ {}^{n-1}\mathbb{I}_{n-1}^{*} \, ({}^{n-1}\mathbf{S}_{n-2} \, {}^{n-2}\dot{\mathbf{V}}_{n-2} + \ddot{q}_{n-1} \, {}^{n-1}\mathbb{a}_{n-1} + {}^{n-1}\boldsymbol{\gamma}_{n-1}) = {}^{n-1}\mathbf{F}_{n-1} + {}^{n-1}\boldsymbol{\beta}_{n-1}^{*} \qquad [7.89] $$

where:

$$ {}^{n-1}\mathbb{I}_{n-1}^{*} = {}^{n-1}\mathbb{I}_{n-1} + {}^{n}\mathbf{S}_{n-1}^{T} \, {}^{n}\mathbf{K}_n \, {}^{n}\mathbf{S}_{n-1} \qquad [7.90] $$

$$ {}^{n-1}\boldsymbol{\beta}_{n-1}^{*} = {}^{n-1}\boldsymbol{\beta}_{n-1} - {}^{n}\mathbf{S}_{n-1}^{T} \, {}^{n}\boldsymbol{\alpha}_n \qquad [7.91] $$

Equation [7.90] has the same form as equation [7.83]. Consequently, we can express $\ddot{q}_{n-1}$ and ${}^{n-1}\mathbf{F}_{n-1}$ in terms of ${}^{n-2}\dot{\mathbf{V}}_{n-2}$. Iterating this procedure for $j = n-2, \ldots, 1$, we obtain $\ddot{q}_j$ and ${}^{j}\mathbf{F}_j$ in terms of ${}^{n-1}\dot{\mathbf{V}}_{n-1}$ for for $j = n-1, \ldots, 1$ using equations [2.77] and [7.78] which are similar to [7.83] and [7.85].

7.7. Conclusion

In this chapter, we have presented the dynamics of serial robots using Lagrange and Newton-Euler formulations that are linear in the inertial parameters. The Lagrange formulation allowed us to study the characteristics and properties of the dynamic model of robots, while the Newton-Euler was shown to be the most efficient for real time implementation. In order to increase the efficiency of the Newton-Euler algorithms, we have proposed the use of the base inertial parameters in a customized symbolic programming algorithm. The problem of computing the direct dynamic model for simulating the dynamics of robots has been treated using two methods; the first is based on the Newton-Euler inverse dynamic algorithm,

while the second is based on another Newton-Euler algorithm that does not require the computation of the robot inertia matrix.

We note that the inverse and direct dynamic algorithms using recursive Newton-Euler equations have been generalized to flexible robots [Boyer 98] and to systems with lumped elasticities [Khalil 00a].

Chapter 8

Trajectory generation

8.1. Introduction

A robotic motion task is specified by defining a path along which the robot must move. A *path* is in general defined by a geometric curve or by a sequence of *points* defined either in task coordinates (end-effector coordinates) or in joint coordinates. The issue of trajectory generation is to compute for the control system the desired reference joint or end-effector variables as functions of time such that the robot tracks the desired path. Thus, a *trajectory* refers to a path and a *time history* along the path.

The trajectories of a robot can be classified as follows:

- trajectory between two points with free path between them;
- trajectory between two points via a sequence of desired intermediate points, also called *via points*, with free paths between via points;
- trajectory between two points with constrained path between the points (straight line segment for instance);
- trajectory between two points via intermediate points with constrained paths between the via points.

In the first two classes, the trajectory is generally generated in the joint space. In the last two classes, it is better to generate the trajectory in the task space.

In the next sections, we present trajectory generation techniques related to this classification, but we first analyze the reasons that motivate the choice of either the joint space or the task space for the generation.

8.2. Trajectory generation and control loops

The two approaches to trajectory generation – in the joint space and in the task space – are illustrated in Figures 8.1 and 8.2 (superscripts i and f designate the initial and final values respectively).

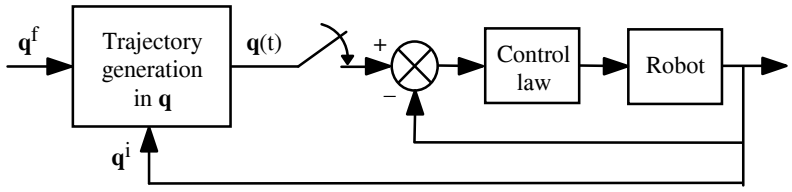

Figure 8.1. *Trajectory generation in the joint space*

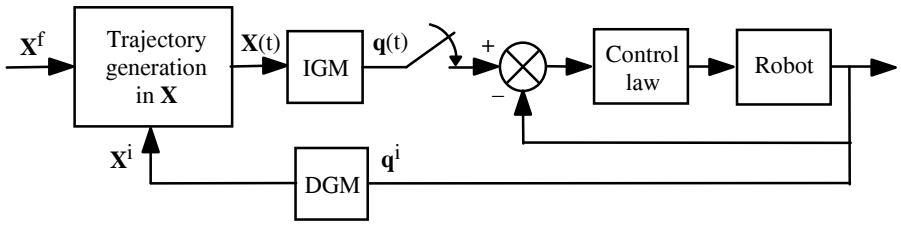

Figure 8.2. *Trajectory generation in the task space*

Trajectory generation in the joint space presents several advantages:

- it requires fewer on-line computations, since there is no need to compute the inverse geometric or kinematic models;
- the trajectory is not affected by crossing singular configurations;
- maximum velocities and torques are determined from actuator data sheets.

The drawback is that the corresponding end-effector path in the task space is not predictable, although it is repetitive, which increases risk of undesirable collisions when the robot works in a cluttered environment. In conclusion, the joint space scheme is appropriate to achieve fast motions in a free space.

Trajectory generation in the task space permits prediction of the geometry of the path. It has, however, a number of disadvantages:

- it may fail when the computed trajectory crosses a singular configuration;
- it fails when the generated points are out of the joint limits or when the robot is forced to change its current aspect (§ 5.7.4);

– velocities and accelerations of the task coordinates depend on the robot configuration. Lower bounds are generally used such that joint velocity and torque limits are satisfied. Consequently, the robot may work under its nominal performance.

The choice of a trajectory generation scheme depends on the application at hand. Each approach has its own limits, due to the fact that constraints are specified either in the joint space (velocities, torques, joint limits), or in the task space (accuracy, geometry of obstacles). Accounting for these remarks, the first two sections cover the problem of trajectory generation between two points in the joint space and the task space respectively. The last section extends the results to trajectory generation between several points.

8.3. Point-to-point trajectory in the joint space

We consider a robot with n degrees of freedom. Let $\mathbf{q}^i$ and $\mathbf{q}^f$ be the joint coordinate vectors corresponding to the initial and final configurations. Let $\mathbf{k}_v$ and $\mathbf{k}_a$ be the vectors of maximum joint velocities and maximum joint accelerations respectively. The value of k_{vj} can be exactly computed from the actuator specifications and transmission ratios, while the value of k_{aj} is approximated by the ratio of the maximum actuator torque to the maximum joint inertia (upper bound of the diagonal element A_{jj} of the inertia matrix defined in Chapter 9). Once the trajectory has been computed with these kinematic constraints, we can proceed to a time scaling in order to better match the maximum joint torques using the dynamic model [Hollerbach 84a].

The trajectory between $\mathbf{q}^i$ and $\mathbf{q}^f$ is determined by the following equation:

$$\mathbf{q}(t) = \mathbf{q}^i + r(t)\,\mathbf{D} \quad \text{for } 0 \le t \le t_f \qquad [8.1]$$

$$\dot{\mathbf{q}}(t) = \dot{r}(t)\,\mathbf{D} \qquad [8.2]$$

with $\mathbf{D} = \mathbf{q}^f - \mathbf{q}^i$.

The boundary conditions of the *interpolation function* r(t) are given by:

$$\begin{cases} r(0) = 0 \\ r(t_f) = 1 \end{cases}$$

Equation [8.1] can also be written as:

$$\mathbf{q}(t) = \mathbf{q}^f(t) - [1 - r(t)]\,\mathbf{D} \qquad [8.3]$$

which is more appropriate for tracking moving objects where $\mathbf{q}^f$ is time-varying [Taylor 79]. In this case, $\mathbf{D} = \mathbf{q}^f(0) - \mathbf{q}^i$.

Several interpolation functions can provide a trajectory such that $\mathbf{q}(0) = \mathbf{q}^i$ and $\mathbf{q}(t_f) = \mathbf{q}^f$. We will study successively the polynomial interpolation, the so-called *bang-bang acceleration profile*, and the bang-bang profile with a constant velocity phase termed *trapeze velocity profile*.

8.3.1. *Polynomial interpolation*

The most commonly used polynomials are the linear interpolation, the third degree polynomials (cubic) and the fifth degree polynomials (quintic).

8.3.1.1. *Linear interpolation*

The trajectory of each joint is described by a linear equation in time. The equation of the joint position is written as:

$$\mathbf{q}(t) = \mathbf{q}^i + \frac{t}{t_f}\mathbf{D} \qquad [8.4]$$

With this trajectory, the position is continuous but not the velocity. This induces undesirable vibrations on the robot and may cause early wear and tear of the mechanical parts.

8.3.1.2. *Cubic polynomial*

If the initial and final velocities are also set to zero, the minimum degree of the polynomial satisfying the constraints is at least three, and has the form:

$$\mathbf{q}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3 \qquad [8.5]$$

The coefficients $\mathbf{a}_i$ are determined from the following boundary conditions:

$$\begin{cases} \mathbf{a}_0 = \mathbf{q}^i \\ \mathbf{a}_1 = \mathbf{0} \\ \mathbf{a}_2 = \dfrac{3}{t_f^2}\mathbf{D} \\ \mathbf{a}_3 = -\dfrac{2}{t_f^3}\mathbf{D} \end{cases} \qquad [8.6]$$

The expression [8.5] can also be written under the form [8.1] or [8.3] with the following interpolation function:

$$r(t) = 3\left(\frac{t}{t_f}\right)^2 - 2\left(\frac{t}{t_f}\right)^3 \qquad [8.7]$$

The cubic polynomial ensures the continuity of velocity but not of acceleration. Practically, the industrial robots are sufficiently rigid so that this discontinuity is filtered by the mechanical structure. Therefore, such a trajectory is generally satisfactory for most applications.

Figure 8.3 shows the position, velocity and acceleration profiles for joint j. The velocity is maximum at $t = t_f / 2$ and its magnitude is given by:

$$|\dot{q}_{jmax}| = \frac{3|D_j|}{2t_f} \quad \text{with } |D_j| = |q_j^f - q_j^i| \qquad [8.8]$$

The maximum acceleration occurs at $t = 0$ and $t = t_f$ with the magnitude:

$$|\ddot{q}_{jmax}| = \frac{6|D_j|}{t_f^2} \qquad [8.9]$$

8.3.1.3. *Quintic polynomial*

For high speed robots or when a robot is handling heavy or delicate loads, it is worth ensuring the continuity of accelerations as well, in order to avoid exciting resonances in the mechanics. The trajectory is said to be of class C^2. Since six constraints have to be satisfied, the interpolation requires a polynomial of at least fifth degree [Binford 77]. The additional two constraints are written as:

$$\begin{cases} \ddot{\mathbf{q}}(0) = \mathbf{0} \\ \ddot{\mathbf{q}}(t_f) = \mathbf{0} \end{cases} \qquad [8.10]$$

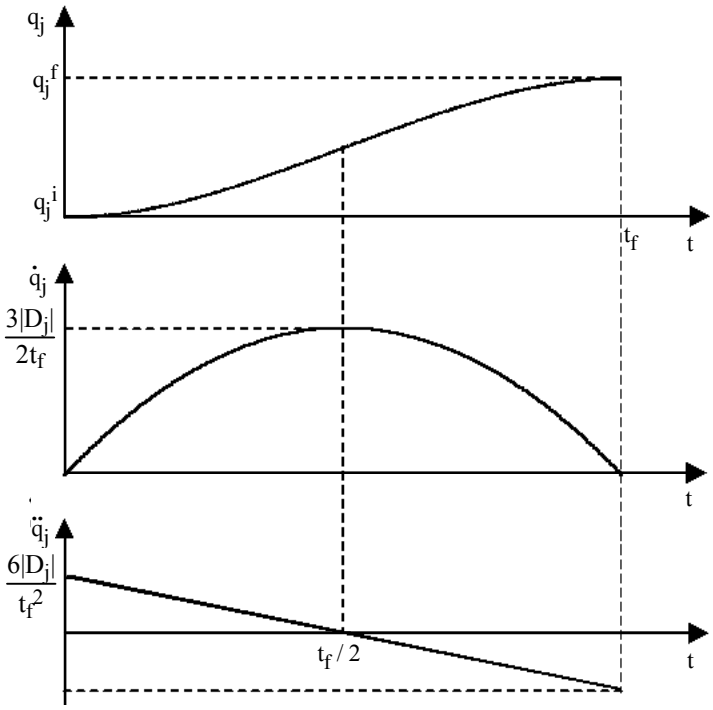

Figure 8.3. *Position, velocity and acceleration profiles for a cubic polynomial*

Solving for the six constraints yields the following interpolation function:

$$r(t) = 10 \left(\frac{t}{t_f}\right)^3 - 15 \left(\frac{t}{t_f}\right)^4 + 6 \left(\frac{t}{t_f}\right)^5 \qquad [8.11]$$

The position, velocity and acceleration with respect to time for joint j are plotted in Figure 8.4. Maximum velocity and acceleration are given by:

$$|\dot{q}_{jmax}| = \frac{15\,|D_j|}{8\,t_f} \qquad [8.12]$$

$$|\ddot{q}_{jmax}| = \frac{10\,|D_j|}{\sqrt{3}\,t_f^2} \qquad [8.13]$$

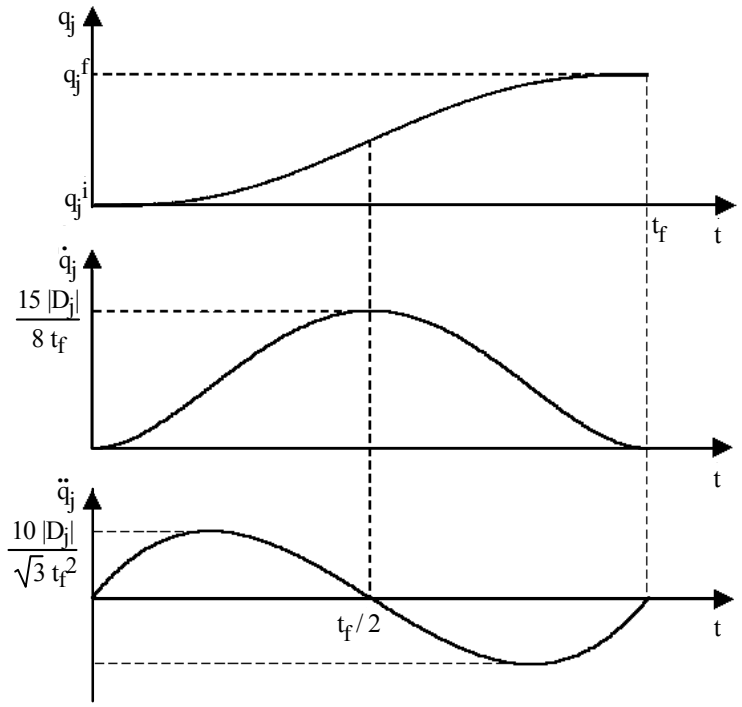

Figure 8.4. *Position, velocity and acceleration profiles for a quintic polynomial*

8.3.1.4. *Computation of the minimum traveling time*

Generally, the duration t_f of a trajectory is not specified. The goal is to minimize the time to travel from the initial configuration $\mathbf{q}^i$ to the final one $\mathbf{q}^f$ while satisfying velocity and acceleration constraints. The approach is to compute the minimum time separately for each joint, and then to synchronize all the joints at a common time.

The minimum traveling time t_{fj} for joint j occurs if either the velocity or the acceleration is saturated during the trajectory. This minimum time is computed from the maximum magnitudes of velocity and acceleration of the different polynomial interpolation functions (Table 8.1). The global minimum traveling time t_f is equal to the largest minimum time:

$$t_f = \max(t_{f1}, \ldots, t_{fn}) \qquad [8.14]$$

Table 8.1. *Minimum traveling time for joint j*

Interpolation function	Minimum time
Linear interpolation	$t_{fj} = \frac{\lvert D_j \rvert}{k_{vj}}$
Cubic polynomial	$t_{fj} = \max\left[\frac{3\,\lvert D_j \rvert}{2\,k_{vj}}, \sqrt{\frac{6\,\lvert D_j \rvert}{k_{aj}}}\right]$
Quintic polynomial	$t_{fj} = \max\left[\frac{15\,\lvert D_j \rvert}{8\,k_{vj}}, \sqrt{\frac{10\,\lvert D_j \rvert}{\sqrt{3}\,k_{aj}}}\right]$
Bang-bang profile (§ 8.3.2)	$t_{fj} = \max\left[\frac{2\,\lvert D_j \rvert}{k_{vj}}, 2\sqrt{\frac{\lvert D_j \rvert}{k_{aj}}}\right]$

8.3.2. *Bang-bang acceleration profile*

A bang-bang acceleration profile consists of a constant acceleration phase until $t_f/2$ followed by a constant deceleration phase (Figure 8.5). The initial and final velocities are zero. Thus, the trajectory is continuous in position and velocity, but discontinuous in acceleration.

The position is given by:

$$\begin{cases} \mathbf{q}(t) = \mathbf{q}^i + 2\left(\frac{t}{t_f}\right)^2 \mathbf{D} & \text{for } 0 \le t \le \frac{t_f}{2} \\ \mathbf{q}(t) = \mathbf{q}^i + \left[-1 + 4\left(\frac{t}{t_f}\right) - 2\left(\frac{t}{t_f}\right)^2\right] \mathbf{D} & \text{for } \frac{t_f}{2} \le t \le t_f \end{cases} \qquad [8.15]$$

For joint j, the maximum velocity and acceleration are given by:

$$\lvert \dot{q}_{jmax} \rvert = \frac{2\,\lvert D_j \rvert}{t_f} \qquad [8.16]$$

$$\lvert \ddot{q}_{jmax} \rvert = \frac{4\,\lvert D_j \rvert}{t_f^2} \qquad [8.17]$$

The minimum traveling time is obtained as for a polynomial trajectory (Table 8.1).

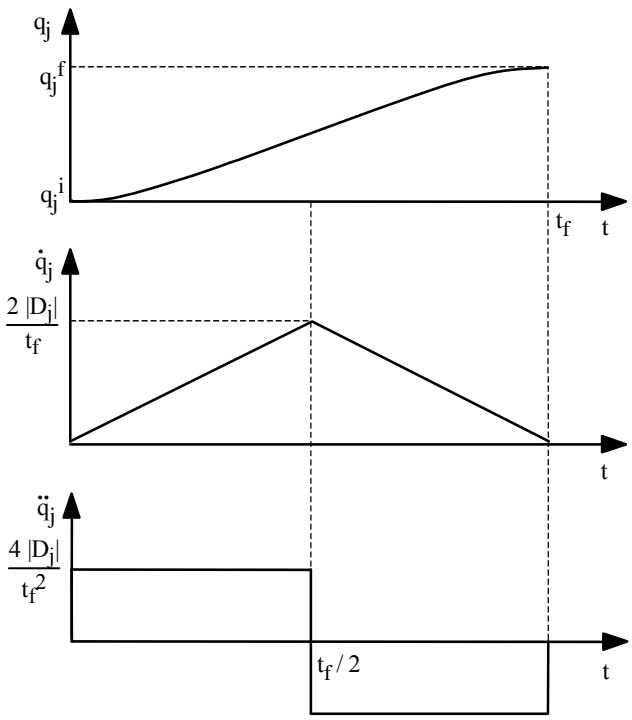

Figure 8.5. *Bang-bang acceleration model*

8.3.3. *Trapezoidal velocity model*

With a bang-bang profile, when the velocity reaches its maximum, adding a constant velocity phase would allow us to saturate also the acceleration and to minimize the traveling time (Figure 8.6). According to equations [8.16] and [8.17], the condition to have a constant velocity phase on joint j is such that:

$$|D_j| > \frac{k_{vj}^2}{k_{aj}} \qquad [8.18]$$

The trapeze velocity trajectory results in the minimum traveling time among those providing the continuity of velocity. The joint j trajectory (Figure 8.7) is represented by the following equations:

$$\begin{cases} q_j(t) = q_j^i + \frac{1}{2} t^2 k_{aj} \text{ sign } (D_j) \text{ for } 0 \le t \le \tau_j \\ q_j(t) = q_j^i + (t - \frac{\tau_j}{2}) k_{vj} \text{ sign } (D_j) \text{ for } \tau_j \le t \le t_{fj} - \tau_j \\ q_j(t) = q_j^f - \frac{1}{2} (t_{fj} - t)^2 k_{aj} \text{ sign } (D_j) \text{ for } t_{fj} - \tau_j \le t \le t_{fj} \end{cases} \quad [8.19]$$

with:

$$\tau_j = \frac{k_{vj}}{k_{aj}} \quad [8.20]$$

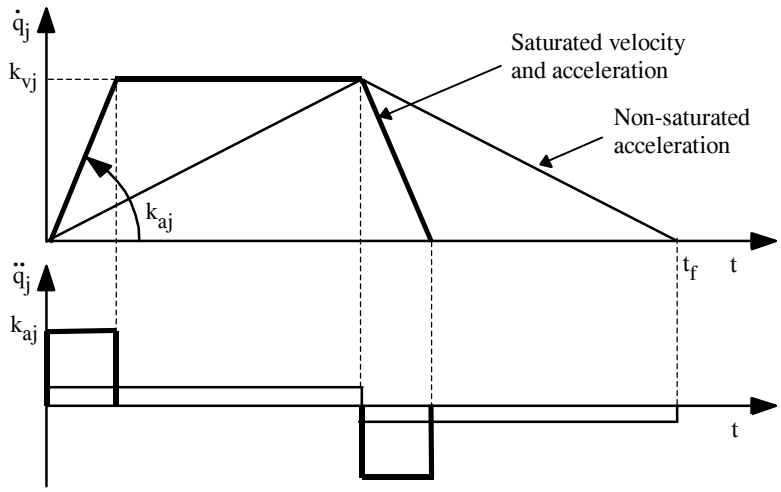

Figure 8.6. *Trapeze velocity profile versus bang-bang acceleration profile*

The area of the trapeze under the velocity curve is equal to the distance traveled in the interval [0, t_{fj}], which can be written as:

$$|D_j| = |q_j^f - q_j^i| = 2\int_0^{\tau_j} k_{aj}\, t\, dt + \int_{\tau_j}^{t_{fj}-\tau_j} k_{vj} dt = k_{vj}\, t_{fj} - \frac{k_{vj}^2}{k_{aj}} \quad [8.21]$$

Hence, the minimum time for joint j is given by:

$$t_{fj} = \frac{k_{vj}}{k_{aj}} + \frac{|D_j|}{k_{vj}} = \tau_j + \frac{|D_j|}{k_{vj}} \quad [8.22]$$

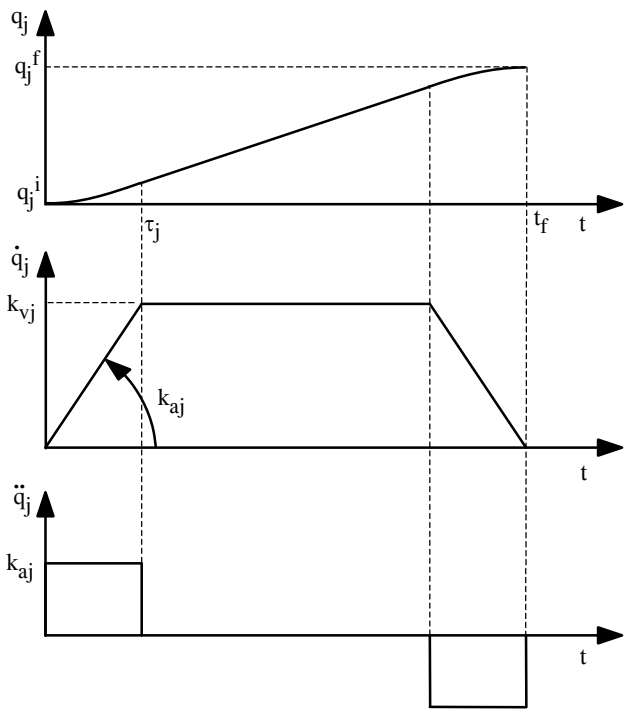

Figure 8.7. *Position, velocity and acceleration profiles for a trapeze trajectory*

In order to synchronize the joint trajectories, we present in the following a method giving homothetical trajectories with the same acceleration and deceleration duration for all joints. Such a method is the most common in use in industrial robot controllers. Let us designate by α_j the ratio between the velocity profile of joint j and an arbitrary joint k. We can write that (Figure 8.8):

$$\dot{q}_j(t) = \alpha_j \dot{q}_k(t) \quad \text{for } j = 1, \ldots, n \qquad [8.23]$$

Doing this, the duration τ of the acceleration phase of the synchronized trajectories is *a priori* not equal to any optimal τ_j computed for each joint separately (equation [8.20]).

Let $\lambda_j k_{vj}$ be the maximum velocity of the synchronized trajectory for joint j and let $\upsilon_j k_{aj}$ be the corresponding maximum acceleration. To calculate the optimal τ, resulting in a minimum time t_f, let us first solve the problem for two joints. According to equation [8.22], the minimum traveling time for each joint, if calculated separately, should be:

$$\begin{cases} t_{f1} = \tau_1 + \dfrac{|D_1|}{k_{v1}} \\ t_{f2} = \tau_2 + \dfrac{|D_2|}{k_{v2}} \end{cases} \qquad [8.24]$$

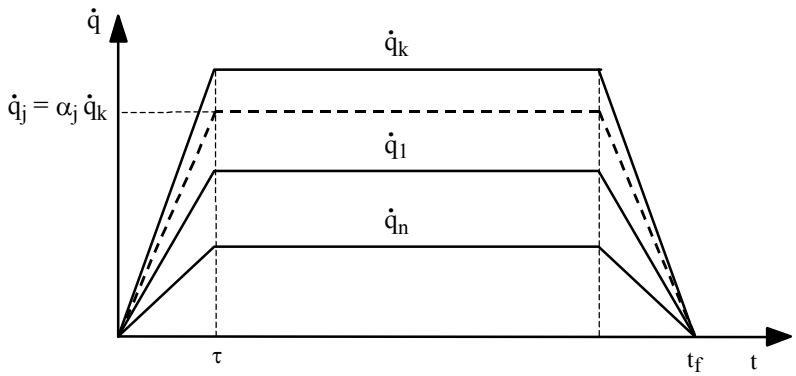

Figure 8.8. *Homothetical velocity profiles*

The synchronized trajectories should be such that:

$$t_f = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} + \frac{|D_1|}{\lambda_1 k_{v1}} = \frac{\lambda_2 k_{v2}}{\upsilon_2 k_{a2}} + \frac{|D_2|}{\lambda_2 k_{v2}} \qquad [8.25]$$

with $t_f \geq \max(t_{f1}, t_{f2})$.

From equation [8.25], it is straightforward to obtain:

$$\tau = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} = \frac{\lambda_2 k_{v2}}{\upsilon_2 k_{a2}} \qquad [8.26]$$

$$\lambda_2 = \lambda_1 \frac{k_{v1}|D_2|}{k_{v2}|D_1|} \qquad [8.27]$$

$$\upsilon_2 = \upsilon_1 \frac{k_{a1}|D_2|}{k_{a2}|D_1|} \qquad [8.28]$$

The velocity constraints imply that:

$$\begin{cases} 0 \le \lambda_1 \le 1 \\ 0 \le \lambda_2 \le 1 \end{cases} \qquad [8.29]$$

Combining the last inequality with equation [8.27] yields:

$$0 \le \lambda_1 \le \frac{k_{v2}|D_1|}{k_{v1}|D_2|} \qquad [8.30]$$

Likewise, from the acceleration constraints, we get:

$$\begin{cases} 0 \le \upsilon_1 \le 1 \\ 0 \le \upsilon_1 \le \dfrac{k_{a2}|D_1|}{k_{a1}|D_2|} \end{cases} \qquad [8.31]$$

The minimum time t_f is obtained when the parameters λ_1 and υ_1 are the largest and satisfy simultaneously the above constraints, which results in:

$$\begin{cases} \lambda_1 = \min\left[1, \dfrac{k_{v2}|D_1|}{k_{v1}|D_2|}\right] \\ \upsilon_1 = \min\left[1, \dfrac{k_{a2}|D_1|}{k_{a1}|D_2|}\right] \end{cases} \qquad [8.32]$$

and the corresponding duration of the acceleration phase is:

$$\tau = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} \qquad [8.33]$$

These equations are easily generalized for n joints:

$$\begin{cases} \lambda_1 = \min\left[1, \dfrac{k_{vj}|D_1|}{k_{v1}|D_j|}\right] \\ \upsilon_1 = \min\left[1, \dfrac{k_{aj}|D_1|}{k_{a1}|D_j|}\right] \end{cases} \quad \text{for } j = 2, \ldots, n \qquad [8.34]$$

assuming that $D_1 \neq 0$ and $D_j \neq 0$.

We note that this trajectory is represented by three polynomials F1 (of second degree), F2 (of first degree) and F3 (of second degree). They are represented by 8 parameters : 3 for F1, 2 for F2 and 3 for F3. These parameters can be calculated

using the following 8 conditions: $F1(0) = q^i$, $\dot{F}1(0) = 0$, $F1(\tau) = F2(\tau)$, $\dot{F}1(\tau) = \dot{F}2(\tau)$, $F2(t_f - \tau) = F3(t_f - \tau)$, $\dot{F}2(t_f - \tau) = \dot{F}3(t_f - \tau)$, $F3(t_f) = q^f$, $\dot{F}3(t_f) = 0$. For given t_f and τ we can obtain different coefficients for these functions. The foregoing relations give the minimum time with kinematic constraints. La relation [8.19] can be rewritten for all the joints in terms of τ and t_f, under the general form [8.1] par la relation suivante as follows:

$$\mathbf{q}(t) = \begin{cases} \mathbf{q}^i + \mathbf{D}\dfrac{t^2}{2\tau(t_f - \tau)} & \text{for} \quad 0 \le t \le \tau \\ \mathbf{q}^i + \mathbf{D}\dfrac{(2t - \tau)}{2(t_f - \tau)} & \text{for} \quad \tau \le t \le t_f - \tau \\ \mathbf{q}^i + \mathbf{D}\left(1 - \dfrac{(t_f - t)^2}{2\tau(t_f - \tau)}\right) & \text{for} \quad t_f - \tau \le t \le t_f \end{cases}$$

NOTE.– If, for a given joint j, the distance $|D_j|$ is such that the maximum velocity k_{vj} cannot be attained, we replace in the above formulas the term k_{vj} by the maximum achievable velocity. According to equation [8.18], this occurs when:

$$|D_j| < \frac{k_{vj}^2}{k_{aj}}$$

which implies that the maximum achievable velocity is:

$$k'_{vj} = \sqrt{|D_j|\, k_{aj}} \qquad [8.35]$$

8.3.4. *Continuous acceleration profile with constant velocity phase*

We can modify the previous method to have a continuous trajectory in acceleration by replacing the acceleration and deceleration phases either by a second degree polynomial (Figure 8.9a) or by a trapeze acceleration profile (Figure 8.9b) [Castain 84]. In the following, we detail the first approach, which is simpler to implement. Let ' be the new duration of the acceleration and let $\lambda_j\, k_{vj}$ be the maximum velocity of the trapeze profile. The boundary conditions for joint j are defined as:

$$q_j(0) = q_j^i,\ \dot{q}_j(0) = 0,\ \ddot{q}_j(0) = 0,\ \dot{q}(\tau) = \lambda_j\, k_{vj}\, \mathrm{sign}(D_j),\ \ddot{q}_j(\tau) = 0 \qquad [8.36]$$

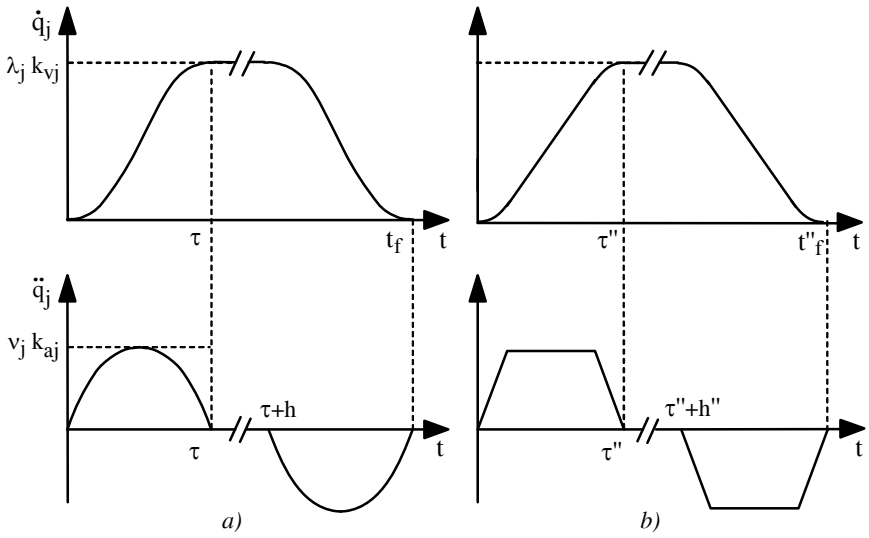

Figure 8.9. *Modification of the acceleration of the trapeze profile to ensure a continuous acceleration*

From these constraints, we derive the equations of position, velocity and acceleration of joint j for $1 \leq j \leq n$ as $0 \leq t \leq \tau$ as follows:

$$q_j(t) = q_j^i - \frac{1}{\tau'^3} \lambda_j k_{vj} \operatorname{sign}(D_j) \left(\frac{1}{2} t - \tau\right) t^3 \qquad [8.37]$$

$$\dot{q}_j(t) = -\frac{1}{\tau'^3} \lambda_j k_{vj} \operatorname{sign}(D_j) (2t - 3\tau) t^2 \qquad [8.38]$$

$$\ddot{q}_j(t) = -\frac{6}{\tau'^3} \lambda_j k_{vj} \operatorname{sign}(D_j) (t - \tau) t \qquad [8.39]$$

The acceleration is maximum at $t = \tau/2$ and its magnitude is:

$$|\ddot{q}_{jmax}| = \frac{3}{2} \frac{\lambda_j k_{vj}}{\tau} \qquad [8.40]$$

If we take for $|\ddot{q}_{jmax}|$ the value $\upsilon_j k_{aj}$ of the velocity trapeze profile, all the joints have the same synchronized duration of acceleration such that:

$$\tau = \frac{3}{2} \frac{\lambda_j k_{vj}}{\upsilon_j k_{aj}} \qquad [8.41]$$

Hence, the duration of the acceleration phase is 1.5 times that with a constant acceleration. The joint position equation corresponding to the constant velocity phase, given a duration h, is as follows:

$$q_j(t) = q_j(\tau) + (t - \tau) \lambda_j k_{vj} \operatorname{sign}(D_j) \quad \text{for } \tau \leq t \leq \tau + h \qquad [8.42]$$

Assuming that the acceleration and deceleration phases are symmetrical ($t_f = 2\tau + h$), the trajectory corresponding to the deceleration phase is defined in the interval $\tau + h \leq t \leq t_f$ as:

$$\begin{cases} q_j(t) = q_j^f + \frac{1}{2} \left[\frac{1}{\tau^3}(t - 3\tau - h)(t - \tau - h)^3 + (2t - 3\tau - 2h)\right] \lambda_j k_{vj} \operatorname{sign}(D_j) \\ \dot{q}_j(t) = \left[\frac{1}{\tau^3} (2t - 5\tau - 2h)(t - \tau - h)^2 + 1\right] \lambda_j k_{vj} \operatorname{sign}(D_j) \\ \ddot{q}_j(t) = \frac{6}{\tau^3} (t - 2\tau - h)(t - \tau - h) \lambda_j k_{vj} \operatorname{sign}(D_j) \end{cases} \qquad [13.43]$$

According to equations [8.37] and [8.41], it should be noted that the distance traveled during the acceleration phase is equal to:

$$|q_j^i - q_j(\tau)| = \frac{3}{4}\frac{(\lambda_j\, k_{vj})^2}{\upsilon_j\, k_{aj}} \qquad [8.44]$$

By computing the area under the velocity curve, we verify that:

$$t_f = \tau + \frac{|D_j|}{\lambda_j\, k_{vj}} \qquad [8.45]$$

This expression is similar to equation [8.22] giving the traveling time for the trapeze profile, which suggests that the computation of λ_j and υ_j can be achieved with equations [8.32]. We note as well that to saturate the velocity and the acceleration of a joint trajectory, the distance to travel must be such that:

$$|D_j| > \frac{3}{2}\frac{k_{vj}^2}{k_{aj}} \qquad [8.46]$$

If this condition is not verified, we replace k_{vj} in equations [8.34] and [8.36] by the maximum achievable velocity:

$$k_{vj} = \sqrt{\frac{2}{3}|D_j|\, k_{aj}} \qquad [8.47]$$

We can show that the number of parameters needed for the three functions representing this motion is equal to 12 and the number of constraint relations is also equal to 12:

$F1(0) = q^i$, $\dot{F}1(0) = 0$, $\ddot{F}1(0) = 0$, $F1(\tau) = F2(\tau)$, $\dot{F}1(\tau) = \dot{F}2(\tau)$, $\ddot{F}1(\tau) = \ddot{F}2(\tau)$, $F2(t_f-\tau) = F3(t_f-\tau)$, $\dot{F}2(tf-\tau) = \dot{F}3(t_f-\tau)$, $\ddot{F}2(t_f-\tau) = \ddot{F}3(t_f-\tau)$, $F3(t_f) = q^f$, $\dot{F}3(t_f) = 0$, $\ddot{F}3(t_f) = 0$.

Once τ and t_f are calculated, the trajectory is defined for all the joints using the following relation:

$$\mathbf{q}(t) = \begin{cases} \mathbf{q}^i + \mathbf{D}\dfrac{1}{2(t_f-\tau)}\left(\dfrac{2t^3}{\tau^2} - \dfrac{t^4}{\tau^3}\right) & 0 \le t \le \tau \\ \mathbf{q}^i + \mathbf{D}\dfrac{(2t-\tau)}{2(t_f-\tau)} & \tau \le t \le t_f - \tau \\ \mathbf{q}^i + \mathbf{D}\left(1 - \dfrac{(t_f-t)^3}{2(t_f-\tau)}\left(\dfrac{2\tau - t_f + t}{\tau^3}\right)\right) & t_f - \tau \le t \le t_f \end{cases}$$

8.4. Point-to-point trajectory in the task space

Let ${}^0\mathbf{T}_E^i$ and ${}^0\mathbf{T}_E^f$ be the homogeneous transformations describing the initial and final desired locations respectively. For convenience, let us note:

$${}^0\mathbf{T}_E^i = \begin{bmatrix} \mathbf{R}^i & \mathbf{P}^i \\ 0\ 0\ 0 & 1 \end{bmatrix} \text{ and } {}^0\mathbf{T}_E^f = \begin{bmatrix} \mathbf{R}^f & \mathbf{P}^f \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

The most common way to move from one location to the other is to split the motion into a linear translation between the origins of frames ${}^0\mathbf{T}_E^i$ and ${}^0\mathbf{T}_E^f$, and a rotation α around an axis of the end-effector ${}^E\mathbf{u}$ to align $\mathbf{A}^i$ and $\mathbf{A}^f$. The translation and the rotation should be synchronized.

The distance to travel for the translation is obtained as:

$$D = \|\mathbf{P}^f - \mathbf{P}^i\| = \sqrt{(P_x^f - P_x^i)^2 + (P_y^f - P_y^i)^2 + (P_z^f - P_z^i)^2} \qquad [8.48]$$

The terms $\mathbf{u}$ and α are computed from the equation:

$$\mathbf{R}^i\,\mathbf{rot}(\mathbf{u}, \alpha) = \mathbf{R}^f \qquad [8.49]$$

where we recall that **rot**(**u**, α) is a (3x3) rotation matrix corresponding to a rotation of an angle α about a vector **u**. Hence, we get:

$$\mathbf{rot}(\mathbf{u}, \alpha) = [\mathbf{R}^i]^T \mathbf{R}^f = \begin{bmatrix} \mathbf{s}^{iT} \\ \mathbf{n}^{iT} \\ \mathbf{a}^{iT} \end{bmatrix} \begin{bmatrix} \mathbf{s}^f & \mathbf{n}^f & \mathbf{a}^f \end{bmatrix} = \begin{bmatrix} \mathbf{s}^i.\mathbf{s}^f & \mathbf{s}^i.\mathbf{n}^f & \mathbf{s}^i.\mathbf{a}^f \\ \mathbf{n}^i.\mathbf{s}^f & \mathbf{n}^i.\mathbf{n}^f & \mathbf{n}^i.\mathbf{a}^f \\ \mathbf{a}^i.\mathbf{s}^f & \mathbf{a}^i.\mathbf{n}^f & \mathbf{a}^i.\mathbf{a}^f \end{bmatrix} \qquad [8.50]$$

the symbol "**.**" designating the dot product. Using equations [2.34] through [2.37] yields:

$$\begin{cases} C\alpha = \frac{1}{2}[\mathbf{s}^i.\mathbf{s}^f + \mathbf{n}^i.\mathbf{n}^f + \mathbf{a}^i.\mathbf{a}^f - 1] \\ S\alpha = \frac{1}{2}\sqrt{(\mathbf{a}^i.\mathbf{n}^f - \mathbf{n}^i.\mathbf{a}^f)^2 + (\mathbf{s}^i.\mathbf{a}^f - \mathbf{a}^i.\mathbf{s}^f)^2 + (\mathbf{n}^i.\mathbf{s}^f - \mathbf{s}^i.\mathbf{n}^f)^2} \\ \alpha = \text{atan2}(S\alpha, C\alpha) \\ \mathbf{u} = \frac{1}{2S\alpha}\begin{bmatrix} \mathbf{a}^i.\mathbf{n}^f - \mathbf{n}^i.\mathbf{a}^f \\ \mathbf{s}^i.\mathbf{a}^f - \mathbf{a}^i.\mathbf{s}^f \\ \mathbf{n}^i.\mathbf{s}^f - \mathbf{s}^i.\mathbf{n}^f \end{bmatrix} \end{cases} \qquad [8.51]$$

When Sα is small, the vector **u** is computed as indicated in § 2.3.8.

Let k_{v1} and k_{a1} be the maximum velocity and acceleration for the translation motion, and let k_{v2} and k_{a2} be the maximum velocity and acceleration for the rotation motion. The methods described in § 8.3 can be used to generate a synchronized trajectory for the two variables D and α, resulting in the minimum time t_f. The trajectory of the end-effector frame is given by:

$${}^0\mathbf{T}_E(t) = \begin{bmatrix} & \mathbf{A}(t) & & \mathbf{P}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [8.52]$$

with:

$$\mathbf{P}(t) = \mathbf{P}^i + \frac{s(t)}{D}(\mathbf{P}^f - \mathbf{P}^i) = \mathbf{P}^i + r(t)\,(\mathbf{P}^f - \mathbf{P}^i) \qquad [8.53]$$

$$\mathbf{R}(t) = \mathbf{R}^i\,\mathbf{rot}(\mathbf{u}, r(t)\,\alpha) \qquad [8.54]$$

where s(t) = D r(t) is the curvilinear distance traveled at time t and r(t) is the interpolation function.

NOTES.–

– we can specify the rotation from $\mathbf{A}^i$ to $\mathbf{A}^f$ with the three Euler angles ϕ, θ and ψ. Let $(\phi^i, \theta^i, \psi^i)$ and $(\phi^f, \theta^f, \psi^f)$ designate the Euler angles corresponding to $\mathbf{A}^i$ and $\mathbf{A}^f$ respectively. Thus, equation [8.54] is replaced by:

$$\mathbf{R}(t) = \mathbf{R}^i\, \mathbf{rot}(\mathbf{z}, \phi^i + r(t)\,\phi)\, \mathbf{rot}(\mathbf{x}, \theta^i + r(t)\,\theta)\, \mathbf{rot}(\mathbf{z}, \psi^i + r(t)\,\psi) \qquad [8.55]$$

with $\phi = \phi^f - \phi^i$, $\theta = \theta^f - \theta^i$, $\psi = \psi^f - \psi^i$. The computation of ϕ, θ and ψ can be carried out as described in § 3.6.1. Thus, we have to deal with four variables: D, ϕ, θ and ψ;

– we can also choose to specify the rotation around an axis that is fixed with respect to frame R_0. In this case, $\mathbf{u}$ and α are calculated by solving:

$$\mathbf{rot}(\mathbf{u}, \alpha)\, \mathbf{R}^i = \mathbf{R}^f \qquad [8.56]$$

– the angular velocity $\boldsymbol{\omega}$ of the end-effector, with respect to the frame where $\mathbf{u}$ is defined, is such that:

$$\boldsymbol{\omega} = \mathbf{u}\, \dot{r}(t)\, \alpha = w\, \mathbf{u} \qquad [8.57]$$

8.5. Conclusion

In this chapter, we have presented several methods of trajectory generation that are commonly used in robotics. We dealt particularly with point-to-point trajectories: different interpolation functions have been studied, namely the trapeze velocity profile, which is implemented in most of the industrial controllers. For each function, we computed the minimum traveling time, from which it is possible to synchronize the joints so that they reach the final point simultaneously. These methods apply for both joint space and task space. The choice of a space depends on the trajectory specification and on the task description.

The interested reader will find in [Shin 85], [Fourquet 90], [Shiller 94], other techniques using the dynamic model which allows replacement of the constraints of acceleration by those more realistic of actuator torques. Likewise, in [Pledel 96], an approach using an exact model of actuators is considered. However, instead of implementing these techniques, an *a posteriori* verification of the constraint validity and scaling the traveling time may be satisfactory [Hollerbach 84a].

Modélisation, identification et commande des robots

Chapter 9

Motion control

9.1. Introduction

The problem of controlling robots has been extensively addressed in the literature. A great variety of control approaches have been proposed. The most common in use with present industrial robots is a decentralized "proportional, integral, derivative" (PID) control for each degree of freedom. More sophisticated nonlinear control schemes have been developed, such as so-called *computed torque control*, termed *inverse dynamic control*, which linearizes and decouples the equation of motion of the robot. Owing to the modeling uncertainties, nonlinear adaptive techniques have been considered in order to identify on-line the dynamic parameters. More recently, properties of the dynamic model have led Lyapunov-based and passivity-based controls to be proposed.

In this chapter, we first study the classical PID control, then the nonlinear linearizing and decoupling control, which is considered to be the best theoretical solution for the control of robots. Detailed surveys on robot control can be found in [Spong 89], [Samson 91], [Lewis 93], [Zodiac 96].

9.2. Equations of motion

In order to understand the basic problem of robot control, it is useful to recall the dynamic model whose general form for a robot with n degrees of freedom is the following:

$$\mathbf{\Gamma} = \mathbf{A}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{Q}(\mathbf{q}) + \mathbf{diag}(\dot{\mathbf{q}})\,\mathbf{F}_v + \mathbf{diag}(\mathrm{sign}(\dot{\mathbf{q}}))\,\mathbf{F}_c \qquad [9.1]$$

or, in a more compact form:

$$\mathbf{\Gamma} = \mathbf{A}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) \qquad [9.2]$$

and, since the model is linear in the dynamic parameters (equation [12.18]), we can write:

$$\mathbf{\Gamma} = \mathbf{\Phi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\chi \qquad [9.3]$$

where $\mathbf{\Gamma}$ is the (nx1) vector of joint torques; $\mathbf{A}(\mathbf{q})$ is the (nxn) inertia matrix of the robot; $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}}$ is the (nx1) vector of Coriolis and centrifugal torques; $\mathbf{Q}(\mathbf{q})$ is the vector of gravity torques; $\mathbf{F}_v$ and $\mathbf{F}_c$ are the vectors of the viscous friction and Coulomb friction parameters respectively; χ is the vector of the dynamic parameters (inertial parameters and friction parameters).

The torque transmitted to joint j by a current-driven electrical actuator (continuous or synchronous), assuming that the transmissions introduce neither backlash nor flexibility, is expressed by:

$$\Gamma_j = N_j\,K_{aj}\,K_{Tj}\,u_j \qquad [9.4]$$

where N_j is the gear transmission ratio, K_{aj} is the current amplifier gain, K_{Tj} is the torque constant of actuator j, and u_j is the control input of the amplifier.

The design of the control consists of computing the joint actuator torques (Γ_j, then u_j) in order to track a desired trajectory or to reach a given position.

9.3. PID control

9.3.1. *PID control in the joint space*

The dynamic model is described by a system of n coupled nonlinear second order differential equations, n being the number of joints. However, for most of today's industrial robots, a local decentralized "proportional, integral, derivative" (PID) control with constant gains is implemented for each joint. The advantages of such a technique are the simplicity of implementation and the low computational cost. The drawbacks are that the dynamic performance of the robot varies according to its configuration, and poor dynamic accuracy when tracking a high velocity trajectory. In many applications, these drawbacks are not of much significance.

Practically, the block diagrams of such a control scheme in the joint space is shown in Figure 9.1. The control law is given by:

$$\mathbf{\Gamma} = \mathbf{K}_p (\mathbf{q}^d - \mathbf{q}) + \mathbf{K}_d (\dot{\mathbf{q}}^d - \dot{\mathbf{q}}) + \mathbf{K}_I \int_{t0}^{t} (\mathbf{q}^d - \mathbf{q}) d\tau \qquad [9.5]$$

where $\mathbf{q}^d(t)$ and $\dot{\mathbf{q}}^d(t)$ denote the desired joint positions and velocities, and where $\mathbf{K}_p$, $\mathbf{K}_d$ and $\mathbf{K}_I$ are (nxn) positive definite diagonal matrices whose generic elements are the proportional K_{pj}, derivative K_{dj} and integral K_{Ij} gains respectively.

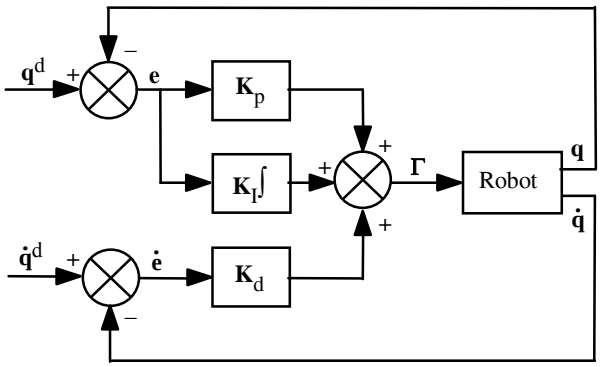

Figure 9.1. *Block diagram of a PID control scheme in the joint space*

The computation of K_{pj}, K_{dj} and K_{Ij} is carried out by considering that joint j is modeled by a linear second order differential equation such that:

$$\Gamma_j = a_j \ddot{q}_j + F_{vj} \dot{q}_j + \gamma_j \qquad [9.6]$$

where $a_j = A_{jj_{max}}$ is the maximum magnitude of the A_{jj} element of the inertia matrix of the robot and γ_j represents a disturbance torque.

Hence, assuming $\gamma_j = 0$, the closed-loop transfer function is given by:

$$\frac{q_j(s)}{q_j^d(s)} = \frac{K_{dj}\, s^2 + K_{pj}\, s + K_{Ij}}{a_j\, s^3 + (K_{dj} + F_{vj})\, s^2 + K_{pj}\, s + K_{Ij}} \qquad [9.7]$$

and the characteristic equation is:

$$\Delta(s) = a_j s^3 + (K_{dj} + F_{vj}) s^2 + K_{pj} s + K_{Ij} \qquad [9.8]$$

The most common solution in robotics consists of adjusting the gains in order to obtain a negative real triple pole. This yields the fastest possible response without overshoot. Thus, the characteristic equation is written as:

$$\Delta(s) = a_j (s + \omega_j)^3 \qquad [9.9]$$

with $\omega_j > 0$, and after solution, we obtain:

$$\begin{cases} K_{pj} = 3\, a_j\, \omega_j^2 \\ K_{dj} + F_{vj} = 3\, a_j\, \omega_j \\ K_{Ij} = a_j\, \omega_j^3 \end{cases} \qquad [9.10]$$

NOTES.–

- high gains $\mathbf{K}_p$ and $\mathbf{K}_d$ decrease the tracking error but bring the system to the neighborhood of the instability domain. Thus, the frequency ω_j should not be greater than the structural resonance frequency ω_{rj}. A reasonable trade-off is that $\omega_j = \omega_{rj} / 2$;
- in the absence of integral action, a static error due to gravity may affect the final position. Practically, the integral action can be deactivated when the position error is very large, since the proportional action is sufficient. It should also be deactivated if the position error becomes too small in order to avoid oscillations that could be caused by Coulomb frictions;
- the predictive action $\mathbf{K}_d\, \dot{\mathbf{q}}^d$ of equation [9.5] reduces significantly the tracking errors. In classical control engineering, this action is not often used;
- the gain $\mathbf{K}_d$ is generally integrated within the servo amplifier, whereas gain $\mathbf{K}_p$ is numerically implemented;
- the performance of a robot controlled in this way is acceptable if high-gear transmission ratios are used (scaling down the time-varying inertias and the coupling torques), if the robot is moving at low velocity, and if high position gains are assigned [Samson 83].

9.3.2. *Stability analysis*

If gravity effects are compensated by an appropriate mechanical design as for the SCARA robot, or by the control software, it can be shown that a PD control law is asymptotically stable for the regulation control problem [Arimoto 84]. The

demonstration is based on the definition of the following Lyapunov function candidate (Appendix 9):

$$V = \frac{1}{2}\,\dot{\mathbf{q}}^T\,\mathbf{A}(\mathbf{q})\,\dot{\mathbf{q}} + \frac{1}{2}\,\mathbf{e}^T\,\mathbf{K}_p\,\mathbf{e} \qquad [9.11]$$

where $\mathbf{e} = \mathbf{q}^d - \mathbf{q}$ is the position error, and where $\mathbf{q}^d$ is the desired joint position.

Since $\mathbf{q}^d$ is constant, the PD control law is given by:

$$\boldsymbol{\Gamma} = \mathbf{K}_p\,\mathbf{e} - \mathbf{K}_d\,\dot{\mathbf{q}} + \mathbf{Q}(\mathbf{q}) \qquad [9.12]$$

From equations [9.1] and [9.12], and in the absence of friction, we obtain the following closed-loop equation:

$$\mathbf{K}_p\,\mathbf{e} - \mathbf{K}_d\,\dot{\mathbf{q}} = \mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}} \qquad [9.13]$$

Differentiating the Lyapunov function [9.11] with respect to time yields:

$$\dot{V} = \frac{1}{2}\,\dot{\mathbf{q}}^T\,\dot{\mathbf{A}}\,\dot{\mathbf{q}} + \dot{\mathbf{q}}^T\,\mathbf{A}\,\ddot{\mathbf{q}} - \mathbf{e}^T\,\mathbf{K}_p\,\dot{\mathbf{q}} \qquad [9.14]$$

and substituting $\mathbf{A}\,\ddot{\mathbf{q}}$ from equation [9.13] yields:

$$\dot{V} = \frac{1}{2}\,\dot{\mathbf{q}}^T\,(\dot{\mathbf{A}} - 2\,\mathbf{C})\,\dot{\mathbf{q}} - \dot{\mathbf{q}}^T\,\mathbf{K}_d\,\dot{\mathbf{q}} \qquad [9.15]$$

Since the matrix $[\dot{\mathbf{A}} - 2\,\mathbf{C}]$ is skew-symmetric [Koditschek 84], [Arimoto 84] (§ 9.3.3.3), the term $\dot{\mathbf{q}}^T\,[\dot{\mathbf{A}} - 2\,\mathbf{C}]\,\dot{\mathbf{q}}$ is zero, giving:

$$\dot{V} = -\,\dot{\mathbf{q}}^T\,\mathbf{K}_d\,\dot{\mathbf{q}} \leq 0 \qquad [9.16]$$

This result shows that $\dot{V}$ is negative semi-definite, which is not sufficient to demonstrate that the equilibrium point ($\mathbf{e} = \mathbf{0}$, $\dot{\mathbf{q}} = \mathbf{0}$) is asymptotically stable (Appendix 9). We have now to prove that as $\dot{\mathbf{q}} = \mathbf{0}$, the robot does not reach a configuration $\mathbf{q} \neq \mathbf{q}^d$. This can be done, thanks to the La Salle invariant set theorem [Hahn 67] (Appendix 9). The set $\mathcal{R}$ of points in the neighborhood of the equilibrium that satisfies $\dot{V} = 0$ is such that $\dot{\mathbf{q}} = \mathbf{0}$ and thus $\ddot{\mathbf{q}} = \mathbf{0}$. From equation [9.13], we conclude that necessarily $\mathbf{e} = \mathbf{0}$. Consequently, the equilibrium point ($\mathbf{e} = \mathbf{0}$, $\dot{\mathbf{q}} = \mathbf{0}$) is the only possible equilibrium for the system and is also the largest invariant set in $\mathcal{R}$. Therefore, the equilibrium point is asymptotically stable.

Furthermore, it has been demonstrated that the system is asymptotically stable if in equation [9.12] we replace $\mathbf{Q}(\mathbf{q})$ by the constant term $\mathbf{Q}(\mathbf{q}^d)$, corresponding to gravity torque at the desired position $\mathbf{q}^d$. The stability is also proven if one takes $K_{pj} > \|\partial\mathbf{Q}(\mathbf{q})/\partial\mathbf{q}\|$, which represents the 2-norm of the Jacobian matrix of gravity torques with respect to the joint variables [Korrami 88], [Tomei 91]. For more details on the computation of the gains when considering the robot dynamics, interested readers should refer to [Qu 91], [Kelly 95], [Rocco 96], [Freidovich 97].

9.3.3. *PID control in the task space*

When the motion is specified in the task space, one of the following schemes can be used to control the system:

- the control law is designed in the task space;
- the specified trajectory in the task space is transformed into a trajectory in the joint space, then a control in the joint space is performed.

For PID control in the task space, the control law is obtained by replacing $\mathbf{q}$ by $\mathbf{X}$ in equation [9.5] and by transforming the task space error signal into the joint space by multiplying it by $\mathbf{J}^T$(Figure 9.2):

$$\mathbf{\Gamma} = \mathbf{J}^T [\mathbf{K}_p (\mathbf{X}^d - \mathbf{X}) + \mathbf{K}_d (\dot{\mathbf{X}}^d - \dot{\mathbf{X}}) + \mathbf{K}_I \int_{t0}^{t} (\mathbf{X}^d - \mathbf{X})\, d\tau] \qquad [9.17]$$

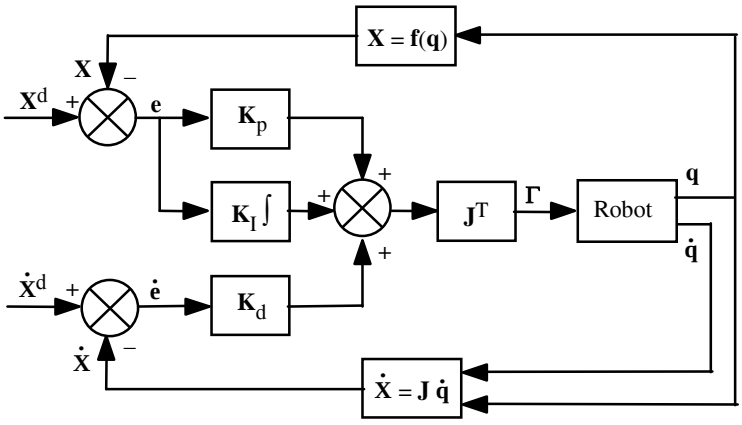

Figure 9.2. *Block diagram of a PID control scheme in the task space*

Two solutions are possible to transform the desired task space trajectory into the joint space: either we use the IGM to compute the joint positions then we compute the velocities and accelerations by differentiating the positions; or we compute the joint positions, velocities and accelerations as indicated below:

i) using the IGM (Chapter 4) to compute the joint positions:

$$\mathbf{q}^d = \mathbf{g}(\mathbf{X}^d) \qquad [9.18]$$

ii) using the IKM (Chapter 6) to compute the joint velocities. In the regular positions:

$$\dot{\mathbf{q}}^d = \mathbf{J}(\mathbf{q}^d)^{-1}\,\dot{\mathbf{X}}^d \qquad [9.19]$$

In singular positions or for redundant robots, the matrix $\mathbf{J}^{-1}$ should be replaced by a generalized inverse as described in Chapter 6;

iii) using the second order IKM (§ 5.10) to compute the joint accelerations (if desired):

$$\ddot{\mathbf{q}}^d = \mathbf{J}^{-1}\,(\ddot{\mathbf{X}}^d - \dot{\mathbf{J}}\,\dot{\mathbf{q}}^d) \qquad [9.20]$$

with:

$$\dot{\mathbf{J}}(\mathbf{q}^d, \dot{\mathbf{q}}^d) = \frac{d}{dt}\mathbf{J}(\mathbf{q}^d) \qquad [9.21]$$

9.4. Linearizing and decoupling control

9.4.1. *Introduction*

When the task requires fast motion of the robot and high dynamic accuracy, it is necessary to improve the performance of the control by taking into account, partially or totally, the dynamic interaction torques. Linearizing and decoupling control is based on canceling the nonlinearities in the robot dynamics [Khalil 78], [Zabala 78], [Raibert 78], [Khatib 80], [Luh 80a], [Freund 82], [Bejczy 85]... Such a control is known as *computed torque control* or *inverse dynamic control* since it is based on the utilization of the dynamic model. Theoretically, it ensures the linearization and the decoupling of the equations of the model, providing a uniform behavior whatever the configuration of the robot.

Implementing this method requires on-line computation of the inverse dynamic model, as well as knowledge of the numerical values of the inertial parameters and friction parameters. Efficient modeling approaches to minimize the computational burden have been presented in § 9.6. With current computer performance, the computation can be handled on-line at a sufficiently high rate and is not anymore a limiting problem. The inertial parameters can be determined off-line with good accuracy by identification techniques as described in Chapter 12.

The linearizing and decoupling techniques consist of transforming a nonlinear control problem into a linear one by using an appropriate feedback law. In the case of rigid robot manipulators, the design of a linearizing and decoupling control law is facilitated by the fact that the number of actuators is equal to the number of joint variables, and that the inverse dynamic model giving the control input Γ of the system as a function of the state vector $(\mathbf{q}, \dot{\mathbf{q}})$ and of $\ddot{\mathbf{q}}$ is naturally obtained. These features ensure that the equations of the robot define a so-called *flat system* whose *flat outputs* are the joint variables $\mathbf{q}$ [Fliess 95]. Since the control law only involves the state variables $\mathbf{q}$ and $\dot{\mathbf{q}}$, it is termed a *static decoupling control law*. In the following, we describe this method both in the joint space and in the task space.

9.4.2. *Computed torque control in the joint space*

9.4.2.1. *Principle of the control*

Let us assume that joint positions and velocities are measurable and that measurements are noiseless. Let $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$ be the estimates of $\mathbf{A}$ and $\mathbf{H}$ respectively. Hence, from equation [9.2], if we define a control law $\boldsymbol{\Gamma}$ such that [Khalil 79]:

$$\boldsymbol{\Gamma} = \hat{\mathbf{A}}(\mathbf{q})\mathbf{w}(t) + \hat{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) \qquad [9.22]$$

then, after substituting [9.22] into [9.2], we deduce that in the ideal case of perfect modeling and in the absence of disturbances, the problem reduces to that of the linear control of n decoupled double-integrators:

$$\ddot{\mathbf{q}} = \mathbf{w}(t) \qquad [9.23]$$

$\mathbf{w}(t)$ is the new input control vector. In order to define $\mathbf{w}(t)$, we study in the following sections two schemes: the first one is suited for *tracking control* when the trajectory is fully specified, the second one is suited for *position (regulation) control* when just the final point is specified.

9.4.2.2. *Tracking control scheme*

Let $\ddot{\mathbf{q}}^d(t)$, $\dot{\mathbf{q}}^d(t)$ and $\mathbf{q}^d(t)$ be the desired acceleration, velocity and position in the joint space. If we define $\mathbf{w}(t)$ according to the following equation[1]:

$$\mathbf{w}(t) = \ddot{\mathbf{q}}^d + \mathbf{K}_d (\dot{\mathbf{q}}^d - \dot{\mathbf{q}}) + \mathbf{K}_p (\mathbf{q}^d - \mathbf{q}) \qquad [9.24]$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ are (nxn) positive definite diagonal matrices; hence, referring to equation [9.23], the closed-loop system response is determined by the following decoupled linear error equation:

$$\ddot{\mathbf{e}} + \mathbf{K}_d \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} = \mathbf{0} \qquad [9.25]$$

where $\mathbf{e} = \mathbf{q}^d - \mathbf{q}$.

The solution $\mathbf{e}(t)$ of the error equation is globally exponentially stable. The gains K_{pj} and K_{dj} are adjusted to provide the axis j, over the whole set of configurations of the robot, the desired dynamics with a given damping coefficient ξ_j, and a given control bandwidth fixed by a frequency ω_j:

$$\begin{cases} K_{pj} = \omega_j^2 \\ K_{dj} = 2\, \xi_j\, \omega_j \end{cases} \qquad [9.26]$$

Generally, one seeks a critically damped system ($\xi_j = 1$) to obtain the fastest response without overshoot. The block diagram of this control scheme is represented in Figure 9.3. The control input torque to the actuators includes three components: the first compensates for Coriolis, centrifugal, gravity, and friction effects; the second is a proportional and derivative control with variable gains $\hat{\mathbf{A}}\, \mathbf{K}_p$ and $\hat{\mathbf{A}}\, \mathbf{K}_d$ respectively; and the third provides a predictive action of the desired acceleration torques $\hat{\mathbf{A}}\, \ddot{\mathbf{q}}^d$.

In the presence of modeling errors, the closed loop equation corresponding to Figure 9.3 is obtained by combining equations [9.22] and [9.2]:

$$\hat{\mathbf{A}} (\ddot{\mathbf{q}}^d + \mathbf{K}_d \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e}) + \hat{\mathbf{H}} = \mathbf{A}\, \ddot{\mathbf{q}} + \mathbf{H} \qquad [9.27]$$

yielding:

$$\ddot{\mathbf{e}} + \mathbf{K}_d \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} = \hat{\mathbf{A}}^{-1} [(\mathbf{A} - \hat{\mathbf{A}})\, \ddot{\mathbf{q}} + \mathbf{H} - \hat{\mathbf{H}}] \qquad [9.28]$$

[1] An integral action on $\mathbf{w}(t)$ can also be added.

In this equation, the modeling errors constitute an excitation for the error equation. When these errors are too large, it is necessary to increase the proportional and derivative gains, but their magnitudes are limited by the stability of the system. The robustness and the stability of this control are addressed by Samson et al. [Samson 87]. It is shown namely that the matrix $\hat{\mathbf{A}}$ must be positive definite. It is shown as well that the errors $\mathbf{e}$ and $\dot{\mathbf{e}}$ decrease while the gains increase.

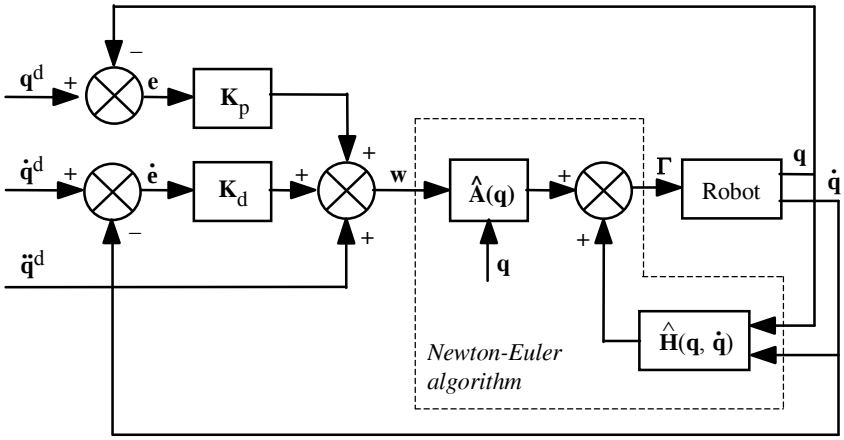

Figure 9.3. *Computed torque: block diagram of the tracking control scheme in the joint space*

9.4.2.3. *Position control scheme*

Let $\mathbf{q}^d$ be the desired position. A possible choice for $\mathbf{w}(t)$ is as follows (Figure 9.4):

$$\mathbf{w}(t) = \mathbf{K}_p (\mathbf{q}^d - \mathbf{q}) - \mathbf{K}_d \dot{\mathbf{q}} \qquad [9.29]$$

From equations [9.23] and [9.29], we obtain the closed-loop equation of the system:

$$\ddot{\mathbf{q}} + \mathbf{K}_d \dot{\mathbf{q}} + \mathbf{K}_p \mathbf{q} = \mathbf{K}_p \mathbf{q}^d \qquad [9.30]$$

which describes a decoupled linear system of second order differential equations. The solution $\mathbf{q}(t)$ is globally exponentially stable by properly choosing $\mathbf{K}_p$ and $\mathbf{K}_d$.

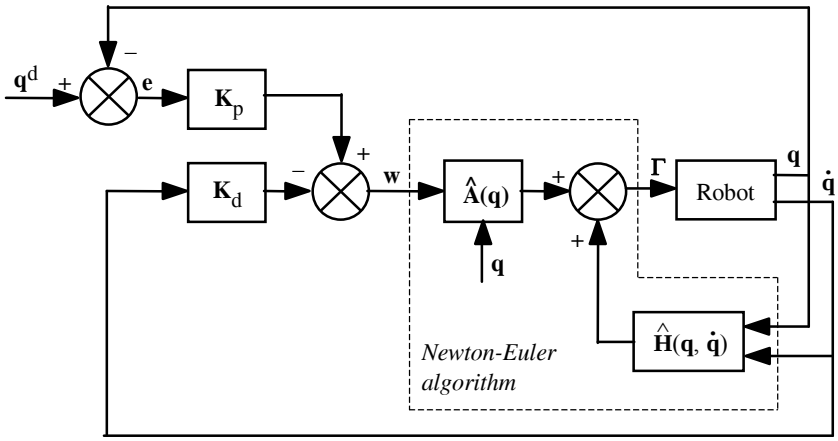

Figure 9.4. *Computed torque: block diagram of the position control scheme in the joint space*

9.4.2.4. *Predictive dynamic control*

Another scheme has been proposed by [Khalil 78] based on a predictive dynamic control: the estimates $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$ are no longer computed with the current values of $\mathbf{q}$ and $\dot{\mathbf{q}}$, but rather with the desired values $\mathbf{q}^d$ and $\dot{\mathbf{q}}^d$. Thus, the control law is written as:

$$\boldsymbol{\Gamma} = \hat{\mathbf{A}}\,(\mathbf{q}^d)\,\mathbf{w}(t) + \hat{\mathbf{H}}\,(\mathbf{q}^d, \dot{\mathbf{q}}^d) \qquad [9.31]$$

where $\mathbf{w}(t)$ is given by equation [9.24] or [9.29] according to the desired scheme.

In the case of exact modeling, we can assume that $\hat{\mathbf{A}}(\mathbf{q}) = \hat{\mathbf{A}}(\mathbf{q}^d)$ and $\hat{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) = \hat{\mathbf{H}}(\mathbf{q}^d, \dot{\mathbf{q}}^d)$. The control law [9.31] linearizes and decouples the equations of the system as in the previous case. The main advantage of this scheme is that the computation of $\hat{\mathbf{A}}(\mathbf{q}^d)$ and $\hat{\mathbf{H}}(\mathbf{q}^d, \dot{\mathbf{q}}^d)$ is not corrupted by noisy variables.

9.4.2.5. *Practical computation of the computed torque control laws*

The control laws [9.22] and [9.31] can be computed by the inverse dynamic Newton-Euler algorithm (§ 9.5) without requiring explicit knowledge of $\mathbf{A}$ and $\mathbf{H}$. The algorithm provides the joint torques as a function of three arguments, namely the vectors of joint positions, velocities and accelerations. By comparing equations [9.2] and [9.22], we can conclude that:

– to compute the control law [9.22] (Figures 9.3 and 9.4), the input arguments of the Newton-Euler algorithm should be:
  - the joint position is equal to the current joint position $\mathbf{q}$;
  - the joint velocity is equal to the current joint velocity $\dot{\mathbf{q}}$;
  - the joint acceleration is equal to $\mathbf{w}(t)$;
– to compute the control law [9.31], the input arguments of the Newton-Euler algorithm should be:
  - the joint position is equal to the desired joint position $\mathbf{q}^d$;
  - the joint velocity is equal to the desired joint velocity $\dot{\mathbf{q}}^d$;
  - the joint acceleration is equal to $\mathbf{w}(t)$.

The computational cost of the computed torque control in the joint space is therefore more or less equal to the number of operations of the inverse dynamic model. As we stated in Chapter 9, the problem of on-line computation of this model at a sufficient rate is now considered solved (Chapter 9). Some industrial robot controllers offer a partial implementation of the computed torque control algorithm.

9.4.3. *Computed torque control in the task space*

The dynamic control in the task space is also known as *resolved acceleration control* [Luh 80a]. The dynamic behavior of the robot in the task space is described by the following equation, obtained after substituting $\ddot{\mathbf{q}}$ from equation [5.44] into equation [9.2]:

$$\mathbf{\Gamma} = \mathbf{A}\,\mathbf{J}^{-1}(\ddot{\mathbf{X}} - \dot{\mathbf{J}}\,\dot{\mathbf{q}}) + \mathbf{H} \qquad [9.32]$$

As in the case of the joint space decoupling control, a control law that linearizes and decouples the equations in the task space is formulated as:

$$\mathbf{\Gamma} = \hat{\mathbf{A}}\,\mathbf{J}^{-1}(\mathbf{w}(t) - \dot{\mathbf{J}}\,\dot{\mathbf{q}}) + \hat{\mathbf{H}} \qquad [9.33]$$

Assuming an exact model, the system is governed by the following equation of a double integrator in the task space:

$$\ddot{\mathbf{X}} = \mathbf{w}(t) \qquad [9.34]$$

Several schemes may be considered for defining $\mathbf{w}$ [Chevallereau 88]. For a tracking control scheme with a PD controller, the control law has the form:

$$\mathbf{w}(t) = \ddot{\mathbf{X}}^d + \mathbf{K}_d(\dot{\mathbf{X}}^d - \dot{\mathbf{X}}) + \mathbf{K}_p(\mathbf{X}^d - \mathbf{X}) \qquad [9.35]$$

The closed-loop behavior of the robot is described by the following error equation:

$$\ddot{\mathbf{e}}_x + \mathbf{K}_d\,\dot{\mathbf{e}}_x + \mathbf{K}_p\,\mathbf{e}_x = \mathbf{0} \qquad [9.36]$$

with:

$$\mathbf{e}_x = \mathbf{X}^d - \mathbf{X} \qquad [9.37]$$

The corresponding block diagram is represented in Figure 9.5. The control input $\boldsymbol{\Gamma}$ can be computed by the inverse dynamic algorithm of Newton-Euler with the following arguments:

- the joint position is equal to the current joint position $\mathbf{q}$;
- the joint velocity is equal to the current joint velocity $\dot{\mathbf{q}}$;
- the joint acceleration is equal to $\mathbf{J}^{-1}(\mathbf{w}(t) - \dot{\mathbf{J}}\,\dot{\mathbf{q}})$.

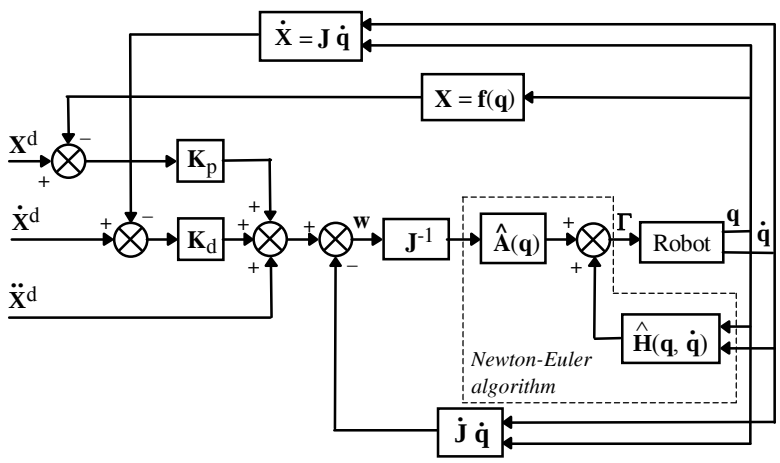

Figure 9.5. *Computed torque control in the task space*

NOTE.– If the robot is redundant, we replace the matrix $\mathbf{J}^{-1}$ in equation [14.33] by a generalized inverse. It can be shown that the robot is also governed by equation [14.36] in non-singular configurations. The homogeneous term of the generalized inverse must be chosen appropriately in order to avoid self joint motions in the null space of $\mathbf{J}$ [Hsu 88], [de Luca 91a], [Ait Mohamed 95].

9.7. Conclusion

Although the controllers of most present day industrial robots are merely designed from linear control theory, more advanced methods must be developed to cope with the nonlinear nature of the articulated structures, namely for applications requiring high dynamic performances (cycle time, dynamic accuracy...).

We presented in this chapter the computed-torque control that the methods achieve this objective. The implementation of such controls requires on-line computation of the inverse dynamic model, which can be carried out according to the algorithms proposed in Chapter 9. In order to estimate the dynamic parameters, we make use of the dynamic identification techniques.

We assumed that the system and the controller are continuous. In practice, the control is achieved by a computer, which introduces time delays due to data acquisition and control law computation. The effect of these delays on the process performance is an issue of the sampling control theory and is out of the scope of this book. However, from an implementation viewpoint, the sampling period should be small enough with respect to the bandwidth of the mechanical system. Typically, a frequency close to 1000 Hz has been used for the controller of the Acma SR400 robot [Restrepo 96]. Note that the use of a high frequency allows us to increase the value of the feedback gains and results in a more robust control [Samson 87].

All the control laws presented in this chapter rely on the availability of joint positions and velocities. All the robots are equipped with high precision sensors for joint position measurements. On the other hand, the tachometers used for joint velocity measurements provide noisy signals. Therefore, it is better to generate the velocity signal by numerical differentiation of the position measurements. Other sophisticated techniques consist of designing a velocity observer from the input torque and the joint position data [Nicosia 90], [Canudas de Wit 92], [Berguis 93], [Khelfi 95], [Cherki 96].

In this chapter, we only considered rigid robots. For further reading about the control of robots with flexible joints, refer for example to [Benallegue 91], [Brogliato 91], [Zodiac 96].

Appendix MOCOM II

Dynamic model of the Stäubli RX-90 robot

In this appendix, we present the simplified Newton-Euler inverse dynamic model of the Stäubli RX-90 robot. This model is obtained automatically using the software package SYMORO+ [Khalil 97]. The inertial parameters correspond to the case of symmetric links, which are given in Table 9.4. The components of the force and moments exerted by the end-effector on the environment are denoted by FX6, FY6, FZ6, CX6, CY6, and CZ6. The joint friction forces are neglected. The velocity, acceleration and torque of joint j are denoted by QPj, QDPj and GAMj respectively. The acceleration of gravity is denoted by G3. As already mentioned, Sj and Cj denote $\sin(\theta_j)$ and $\cos(\theta_j)$ respectively.

Noting that the equations with an asterisk (*) on their left are constants and can be evaluated off-line, the computational cost of this model is 160 multiplications and 113 additions.

```
No31=QDP1*ZZ1R
WI12=QP1*S2
WI22=C2*QP1
WP12=QDP1*S2 + QP2*WI22
WP22=C2*QDP1 – QP2*WI12
DV222=–WI22**2
DV332=–QP2**2
DV122=WI12*WI22
DV132=QP2*WI12
DV232=QP2*WI22
U112=DV222 + DV332
U212=DV122 + QDP2
U312=DV132 – WP22
VP12=– G3*S2
VP22=– C2*G3
PIS22=XXR2 – ZZR2
```

```
No12=WP12*XXR2 + DV232*ZZR2
No22=DV132*PIS22
No32=–DV122*XXR2 + QDP2*ZZR2
WI13=C3*WI12 + S3*WI22
WI23=–S3*WI12 + C3*WI22
W33=QP2 + QP3
WP13=QP3*WI23 + C3*WP12 + S3*WP22
WP23=–QP3*WI13 – S3*WP12 + C3*WP22
WP33=QDP2 + QDP3
DV113=–WI13**2
DV333=–W33**2
DV123=WI13*WI23
DV133=W33*WI13
DV233=W33*WI23
U123=DV123 – WP33
U223=DV113 + DV333
U323=DV233 + WP13
VSP13=d3*U112 + VP12
VSP23=d3*U212 + VP22
VSP33=d3*U312
VP13=C3*VSP13 + S3*VSP23
VP23=–S3*VSP13 + C3*VSP23
F13=MYR3*U123
F23=MYR3*U223
F33=MYR3*U323
*PIS23=XXR3 – ZZR3
No13=WP13*XXR3 + DV233*ZZR3
No23=DV133*PIS23
No33=–DV123*XXR3 + WP33*ZZR3
WI14=–S4*W33 + C4*WI13
WI24=–C4*W33 – S4*WI13
W34=QP4 + WI23
WP14=QP4*WI24 + C4*WP13 – S4*WP33
WP24=–QP4*WI14 – S4*WP13 – C4*WP33
WP34=QDP4 + WP23
DV124=WI14*WI24
DV134=W34*WI14
DV234=W34*WI24
VSP14=RL4*U123 + VP13
VSP24=RL4*U223 + VP23
VSP34=RL4*U323 + VSP33
VP14=C4*VSP14 – S4*VSP34
VP24=–S4*VSP14 – C4*VSP34
```

```
*PIS24=XXR4 – ZZR4
No14=WP14*XXR4 + DV234*ZZR4
No24=DV134*PIS24
No34=–DV124*XXR4 + WP34*ZZR4
WI15=S5*W34 + C5*WI14
WI25=C5*W34 – S5*WI14
W35=QP5 – WI24
WP15=QP5*WI25 + C5*WP14 + S5*WP34
WP25=–QP5*WI15 – S5*WP14 + C5*WP34
WP35=QDP5 – WP24
DV115=–WI15**2
DV335=–W35**2
DV125=WI15*WI25
DV135=W35*WI15
DV235=W35*WI25
U125=DV125 – WP35
U225=DV115 + DV335
U325=DV235 + WP15
VP15=C5*VP14 + S5*VSP24
F15=MYR5*U125
F25=MYR5*U225
F35=MYR5*U325
*PIS25=XXR5 – ZZR5
No15=WP15*XXR5 + DV235*ZZR5
No25=DV135*PIS25
No35=–DV125*XXR5 + WP35*ZZR5
WI16=–S6*W35 + C6*WI15
WI26=–C6*W35 – S6*WI15
W36=QP6 + WI25
WP16=QP6*WI26 + C6*WP15 – S6*WP35
WP36=QDP6 + WP25
DV126=WI16*WI26
DV136=W36*WI16
DV236=W36*WI26
*PIS26=XXR6 – ZZ6
No16=WP16*XXR6 + DV236*ZZ6
No26=DV136*PIS26
No36=–DV126*XXR6 + WP36*ZZ6
N16=CX6 + No16
N26=CY6 + No26
N36=CZ6 + No36
FDI16=C6*FX6 – FY6*S6
FDI36=–C6*FY6 – FX6*S6
```

E15=F15 + FDI16
E25=F25 + FZ6
E35=F35 + FDI36
N15=C6*N16 + No15 – N26*S6 – MYR5*VP24
N25=N36 + No25
N35=–(C6*N26) + No35 – N16*S6 – MYR5*VP15
FDI15=C5*E15 – E25*S5
FDI35=C5*E25 + E15*S5
N14=C5*N15 + No14 – N25*S5
N24=–N35 + No24
N34=C5*N25 + No34 + N15*S5
FDI14=C4*FDI15 + E35*S4
FDI34=C4*E35 – FDI15*S4
E13=F13 + FDI14
E23=F23 + FDI35
E33=F33 + FDI34
N13=C4*N14 + No13 + FDI34*RL4 – N24*S4 + MYR3*VSP33
N23=N34 + No23
N33=–(C4*N24) + No33 – FDI14*RL4 – N14*S4 – MYR3*VP13
FDI23=C3*E23 + E13*S3
N12=C3*N13 + No12 – N23*S3
N22=–(d3*E33) + C3*N23 + No22 + N13*S3
N32=d3*FDI23 + N33 + No32 – MY2*VP12 + MXR2*VP22
N31=C2*N22 + No31 + N12*S2
GAM1=N31
GAM2=N32
GAM3=N33 + IA3*QDP3
GAM4=N34 + IA4*QDP4
GAM5=N35 + IA5*QDP5
GAM6=N36 + IA6*QDP6