Chapter 7
Dynamics of Serial Robotic Manipulators

7.1 Introduction

The main objectives of this chapter are (a) to devise an algorithm for the real-time *computed-torque control* and (b) to derive the system of second-order ordinary differential equations (ODE) governing the motion of an n-axis manipulator. We will focus on serial manipulators, the dynamics of a much broader class of robotic mechanical systems, namely, parallel manipulators and mobile robots, being the subject of Chap. 12. Moreover, we will study mechanical systems with rigid links and rigid joints and will put aside systems with flexible elements, which pertain to a more specialized realm.

7.2 Inverse vs. Forward Dynamics

The two basic problems associated with the dynamics of robotic mechanical systems, namely, the *inverse* and the *forward* problems, are thoroughly discussed in this chapter. The relevance of these problems cannot be overstated: the former is essential for the computed-torque control of robotic manipulators, while the latter is required for the simulation and the real-time feedback control of the same systems. Because the inverse problem is purely algebraic, it is conceptually simpler to grasp than the forward problem, and hence, the inverse problem will be discussed first. Moreover, the inverse problem is also computationally simpler than the forward problem. In the inverse problem, a time-history of either the Cartesian or the joint coordinates is given, and from knowledge of these histories and the architecture and inertial parameters of the system at hand, the torque or force requirements at the different actuated joints are determined as time-histories as well. In the forward problem, current values of the joint coordinates and their first time-derivatives are known at a given instant, the time-histories of the applied torques or forces being also known, along with the architecture and the inertial parameters of the

J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, Mechanical Engineering Series 124, DOI 10.1007/978-3-319-01851-5_7,

© Springer International Publishing Switzerland 2014

manipulator at hand. With these data, the values of the joint coordinates and their time-derivatives are computed at a later sampling instant by integration of the underlying system of nonlinear ordinary differential equations.

The study of the dynamics of systems of multiple rigid bodies is classical, but up until the advent of the computer, it was limited only to theoretical results and a reduced number of bodies. First Uicker (1965) and then Kahn (1969) produced a method based on the Euler–Lagrange equations of mechanical systems of rigid bodies that they used to simulate the dynamical behavior of such systems. A breakthrough in the development of algorithms for dynamics computations was reported by Luh et al. (1980), who proposed a recursive formulation of multibody dynamics that is applicable to systems with serial kinematic chains. This formulation, based on the Newton–Euler equations of rigid bodies, allowed the calculation of the joint torques of a six-revolute manipulator with only 800 multiplications and 595 additions, a tremendous gain if we consider that the straightforward calculation of the Euler–Lagrange equations for the same type of manipulator involves 66,271 multiplications and 51,548 additions, as pointed out by Hollerbach (1980). In the foregoing reference, a recursive derivation of the Euler–Lagrange equations was proposed, whereby the computational complexity was reduced to only 2,195 multiplications and 1,719 additions.

The foregoing results provoked a discussion on the merits and demerits of each of the Euler–Lagrange and the Newton–Euler formulations. Silver (1982) pointed out that since both formulations are equivalent, they should lead to the same computational complexity. In fact, Silver showed how to derive the Euler–Lagrange equations from the Newton–Euler formulation by following an approach first introduced by Kane (1961) in connection with nonholonomic systems. Kane and Levinson (1983) then showed how Kane's equations can be applied to particular robotic manipulators and arrived at lower computational complexities. They applied the said equations to the Stanford Arm (Paul 1981) and computed its inverse dynamics with 646 multiplications and 394 additions. Thereafter, Khalil et al. (1986) proposed a condensed recursive Newton–Euler method that reduced the computational complexity to 538 multiplications and 478 additions, for *arbitrary architectures*. Further developments in this area were reported by Balafoutis and Patel (1991), who showed that the underlying computational complexity can be reduced to 489 multiplications and 420 additions for the most general case of a six-revolute manipulator, i.e., without exploiting particular features of the manipulator geometry. Balafoutis and Patel based their algorithm on tensor analysis, whereby tensor identities are exploited to their fullest extent in order to reduce the number of operations involved. Li and Sankar (1992), in turn, reported further savings that allowed them to bring down those numbers to 459 multiplications and 390 additions.

In this chapter, the inverse dynamics problem is solved with the well-known recursive Newton–Euler algorithm, while the forward dynamics problem is handled with a novel approach, based on the reciprocity relations between the *constraint wrenches* and the *feasible twists* of a manipulator. This technique is developed with the aid of a modeling tool known as the *natural orthogonal complement*, thoroughly discussed in Sect. 7.5.

Throughout the chapter, we will follow a multibody system approach, which requires a review of the underlying fundamentals.

7.3 Fundamentals of Multibody System Dynamics

7.3.1 On Nomenclature and Basic Definitions

We consider here a mechanical system composed of r rigid bodies and denote by $\mathbf{M}_i$ the 6×6 *inertia dyad*—see Sect. 3.8—of the ith body. Moreover, we let $\mathbf{W}_i$, already introduced in Eq. (3.140), be the 6×6 *angular-velocity dyad* of the same body. As pertaining to the case at hand, the said matrices are displayed below:

$$\mathbf{M}_i \equiv \begin{bmatrix} \mathbf{I}_i & \mathbf{O} \\ \mathbf{O} & m_i \mathbf{1} \end{bmatrix}, \quad \mathbf{W}_i \equiv \begin{bmatrix} \boldsymbol{\Omega}_i & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}, \quad i = 1, \ldots, r \tag{7.1}$$

where $\mathbf{1}$ and $\mathbf{O}$ denote the 3×3 identity and zero matrices, respectively, while $\boldsymbol{\Omega}_i$ and $\mathbf{I}_i$ are the angular-velocity and the inertia matrices of the ith body, this last being defined with respect to the center of mass C_i of this body. Moreover, the mass of this body is denoted by m_i, whereas $\mathbf{c}_i$ and $\dot{\mathbf{c}}_i$ denote the position and the velocity vectors of C_i in an inertial frame. Furthermore, let $\mathbf{t}_i$ denote the twist of the same body, the latter being defined in terms of the angular velocity vector $\boldsymbol{\omega}_i$, the vector of $\boldsymbol{\Omega}_i$, and the velocity of C_i. The six-dimensional *momentum screw* $\boldsymbol{\mu}_i$ is defined likewise. Furthermore, $\mathbf{w}_i^W$ and $\mathbf{w}_i^C$ are defined as the *working wrench* and the *nonworking constraint wrench* exerted on the ith body by its neighbors, in which forces are assumed to be applied at C_i. We thus have, for $i = 1, \ldots, r$,

$$\mathbf{t}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \dot{\mathbf{c}}_i \end{bmatrix}, \quad \boldsymbol{\mu}_i = \begin{bmatrix} \mathbf{I}_i \boldsymbol{\omega}_i \\ m_i \dot{\mathbf{c}}_i \end{bmatrix}, \quad \mathbf{w}_i^W = \begin{bmatrix} \mathbf{n}_i^W \\ \mathbf{f}_i^W \end{bmatrix}, \quad \mathbf{w}_i^C = \begin{bmatrix} \mathbf{n}_i^C \\ \mathbf{f}_i^C \end{bmatrix} \tag{7.2}$$

where superscripted $\mathbf{n}_i$ and $\mathbf{f}_i$ stand, respectively, for the moment and the force acting on the ith body, the force being applied at the mass center C_i. Thus, whereas $\mathbf{w}_i^W$ accounts for forces and moments exerted by both the environment and the actuators, including driving forces as well as dissipative effects, $\mathbf{w}_i^C$, whose sole function is to keep the links together, accounts for those forces and moments exerted by the neighboring links, which do not produce any mechanical work. Therefore, friction wrenches applied by the $(i-1)$st and the $(i+1)$st links onto the ith link are not included in $\mathbf{w}_i^C$; rather, they are included in $\mathbf{w}_i^W$.

Clearly, from the definitions of $\mathbf{M}_i$, $\boldsymbol{\mu}_i$, and $\mathbf{t}_i$, we have

$$\boldsymbol{\mu}_i = \mathbf{M}_i \mathbf{t}_i \tag{7.3}$$

Moreover, from Eq. (3.143),[1]

$$\dot{\boldsymbol{\mu}}_i = \mathbf{M}_i \dot{\mathbf{t}}_i + \mathbf{W}_i \boldsymbol{\mu}_i = \mathbf{M}_i \dot{\mathbf{t}}_i + \mathbf{W}_i \mathbf{M}_i \mathbf{t}_i \tag{7.4}$$

We now recall the Newton–Euler equations for a rigid body, namely,

$$\mathbf{I}_i \dot{\boldsymbol{\omega}}_i = -\boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i + \mathbf{n}_i^W + \mathbf{n}_i^C \tag{7.5a}$$

$$m_i \ddot{\mathbf{c}}_i = \mathbf{f}_i^W + \mathbf{f}_i^C \tag{7.5b}$$

which can be written in compact form using the foregoing six-dimensional twist and wrench arrays as well as the 6×6 inertia and angular-velocity dyads. We thus obtain the Newton–Euler equations of the ith body in the form

$$\mathbf{M}_i \dot{\mathbf{t}}_i = -\mathbf{W}_i \mathbf{M}_i \mathbf{t}_i + \mathbf{w}_i^W + \mathbf{w}_i^C \tag{7.5c}$$

7.3.2 The Euler–Lagrange Equations of Serial Manipulators

The Euler–Lagrange dynamical equations of a mechanical system are now recalled, as pertaining to serial manipulators. Thus, the mechanical system at hand has n degrees of freedom, its n independent generalized coordinates being the n joint variables, which are stored in the n-dimensional vector $\boldsymbol{\theta}$. We thus have

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial T}{\partial \boldsymbol{\theta}} = \boldsymbol{\phi} \tag{7.6}$$

where T is a scalar function denoting the *kinetic energy* of the system and $\boldsymbol{\phi}$ is the n-dimensional vector of *generalized force*. If some forces on the right-hand side stem from a potential V, we can, then decompose $\boldsymbol{\phi}$ into two parts, $\boldsymbol{\phi}_p$ and $\boldsymbol{\phi}_{\overline{p}}$, the former arising from V and termed the *conservative force* of the system; the latter is the *nonconservative force*. That is,

$$\boldsymbol{\phi}_p \equiv -\frac{\partial V}{\partial \boldsymbol{\theta}} \tag{7.7}$$

the above Euler–Lagrange equations thus becoming

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{\phi}_{\overline{p}} \tag{7.8}$$

where L is the *Lagrangian* of the system, defined as

$$L \equiv T - V \tag{7.9}$$

[1] See Exercise 7.1 for an extension of this relation to a system of n rigid bodies.

Moreover, the kinetic energy of the system is simply the sum of the kinetic energies of all the r links. Recalling Eq. (3.145), which gives the kinetic energy of a rigid body in terms of six-dimensional arrays, one has

$$T = \sum_{1}^{r} T_i = \sum_{1}^{r} \frac{1}{2} \mathbf{t}_i^T \mathbf{M}_i \mathbf{t}_i \tag{7.10}$$

whereas the vector of nonconservative generalized forces is given by

$$\boldsymbol{\phi}_{\overline{p}} \equiv \frac{\partial \Pi^A}{\partial \dot{\boldsymbol{\theta}}} - \frac{\partial \Delta}{\partial \dot{\boldsymbol{\theta}}} \tag{7.11}$$

in which Π^A and Δ denote the power supplied to the system and the *Rayleigh dissipation function*, or for brevity, the *dissipation function* of the system.

The first of these items is discussed below; the latter is only outlined in this section but is discussed extensively in Sect. 7.8. First, the wrench $\mathbf{w}_i^W$ is decomposed into two parts, $\mathbf{w}_i^A$ and $\mathbf{w}_i^D$, the former being the wrench supplied by the actuators and the latter being the wrench that arises from viscous and Coulomb friction, the gravity wrench being not needed here because gravity effects are considered in the potential $V(\boldsymbol{\theta})$. We thus call $\mathbf{w}_i^A$ the *active wrench* and $\mathbf{w}_i^D$ the dissipative wrench. Here, the wrenches supplied by the actuators are assumed to be prescribed functions of time. Moreover, these wrenches are supplied by single-dof actuators in the form of forces along a line of action or moments in a given direction, both line and direction being fixed to the two bodies that are coupled by an active joint. Hence, the actuator-supplied wrenches are dependent on the posture of the manipulator as well, but not on its twist. That is, the actuator wrenches are functions of both the vector of generalized coordinates, or joint variables, and time, but not of the generalized speeds, or joint-rates. Forces dependent on the latter to be considered here are assumed to be all *dissipative*. As a consequence, they can be readily incorporated into the mathematical model at hand via the dissipation function, to be discussed in Sect. 7.8. Note that feedback control schemes require actuator forces that are functions not only of the generalized coordinates, but also of the generalized speeds. These forces or moments are most easily incorporated into the underlying mathematical model, once this model is derived in the state-variable space, i.e., in the space of generalized coordinates and generalized speeds.

Thus, the power supplied to the ith link, Π_i^A, is readily computed as

$$\Pi_i^A = (\mathbf{w}_i^A)^T \mathbf{t}_i \tag{7.12a}$$

Similar to the kinetic energy, then, the power supplied to the overall system is simply the sum of the individual powers supplied to each link, and expressed as in Eq. (7.12a), i.e.,

$$\Pi^A \equiv \sum_{1}^{r} \Pi_i^A \tag{7.12b}$$

Further definitions are now introduced. These are the $6n$-dimensional vectors of *manipulator twist*, $\mathbf{t}$; *manipulator momentum*, $\boldsymbol{\mu}$; *manipulator constraint wrench*, $\mathbf{w}^C$; *manipulator active wrench*, $\mathbf{w}^A$; and *manipulator dissipative wrench*, $\mathbf{w}^D$. Additionally, the $6n \times 6n$ matrices of *manipulator mass*, $\mathbf{M}$, and *manipulator angular velocity*, $\mathbf{W}$, are also introduced below:

$$\mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_n \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_n \end{bmatrix}, \tag{7.13a}$$

$$\mathbf{w}^C = \begin{bmatrix} \mathbf{w}_1^C \\ \vdots \\ \mathbf{w}_n^C \end{bmatrix}, \quad \mathbf{w}^A = \begin{bmatrix} \mathbf{w}_1^A \\ \vdots \\ \mathbf{w}_n^A \end{bmatrix}, \quad \mathbf{w}^D = \begin{bmatrix} \mathbf{w}_1^D \\ \vdots \\ \mathbf{w}_n^D \end{bmatrix} \tag{7.13b}$$

$$\mathbf{M} = \operatorname{diag}(\mathbf{M}_1, \ldots, \mathbf{M}_n), \qquad \mathbf{W} = \operatorname{diag}(\mathbf{W}_1, \ldots, \mathbf{W}_n) \tag{7.13c}$$

It is now apparent that, from definitions (7.13b and 7.13c) and relation (7.3), we have

$$\boldsymbol{\mu} = \mathbf{M}\mathbf{t} \tag{7.14}$$

Moreover, from definitions (7.1) and (7.2),

$$\dot{\boldsymbol{\mu}} = \mathbf{M}\dot{\mathbf{t}} + \mathbf{W}\mathbf{M}\mathbf{t} \tag{7.15}$$

With the foregoing definitions, then, the kinetic energy of the manipulator takes a simple form, namely,

$$T = \frac{1}{2}\mathbf{t}^T\mathbf{M}\mathbf{t} \equiv \frac{1}{2}\mathbf{t}^T\boldsymbol{\mu} \tag{7.16}$$

which is a quadratic form in the system twist. Since the twist, on the other hand, is a linear function of the vector $\dot{\boldsymbol{\theta}}$ of joint rates, the kinetic energy turns out to be a quadratic form in the vector of joint rates. Moreover, we will assume that this form is *homogeneous* in $\dot{\boldsymbol{\theta}}$, i.e.,

$$T = \frac{1}{2}\dot{\boldsymbol{\theta}}^T\mathbf{I}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{7.17}$$

Notice that the above assumption implies that the base of the robot is fixed to an inertial base, and hence, when all joints are locked, the kinetic energy of the robot vanishes, which would not be the case if, for example, the robot were mounted on the International Space Station. If this were the case, then the kinetic energy would not vanish even if all robot joints were locked, which means that the foregoing

kinetic-energy expression would include a linear term in $\dot{\boldsymbol{\theta}}$ and a term independent of the joint-rates. In any event, it is apparent that

$$\mathbf{I}(\boldsymbol{\theta}) = \frac{\partial^2}{\partial \dot{\boldsymbol{\theta}}^2}(T) \tag{7.18}$$

which means that the $n \times n$ generalized inertia matrix is the *Hessian* matrix of the kinetic energy with respect to the vector of generalized speed.

Furthermore, the Euler–Lagrange equations can be written in the form

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial T}{\partial \boldsymbol{\theta}} + \frac{\partial V}{\partial \boldsymbol{\theta}} = \boldsymbol{\phi}_n \tag{7.19a}$$

Now, from the form of T given in Eq. (7.17), the partial derivatives appearing in the foregoing equation take the forms derived below:

$$\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}} = \mathbf{I}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$$

and hence,

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}}\right) = \mathbf{I}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \dot{\mathbf{I}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} \tag{7.19b}$$

Moreover, in order to calculate the second term of the left-hand side of Eq. (7.19a), we express the kinetic energy in the form

$$T = \frac{1}{2}\mathbf{p}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})^T \dot{\boldsymbol{\theta}} \tag{7.19c}$$

where $\mathbf{p}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the *generalized momentum* of the manipulator, defined as

$$\mathbf{p}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \equiv \mathbf{I}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{7.19d}$$

Hence,

$$\frac{\partial T}{\partial \boldsymbol{\theta}} = \frac{1}{2}\left(\frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}\right)^T \dot{\boldsymbol{\theta}} \tag{7.19e}$$

or

$$\frac{\partial T}{\partial \boldsymbol{\theta}} = \frac{1}{2}\left[\frac{\partial (\mathbf{I}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}\right]^T \dot{\boldsymbol{\theta}} \tag{7.19f}$$

the Euler–Lagrange equations thus taking on the alternative form

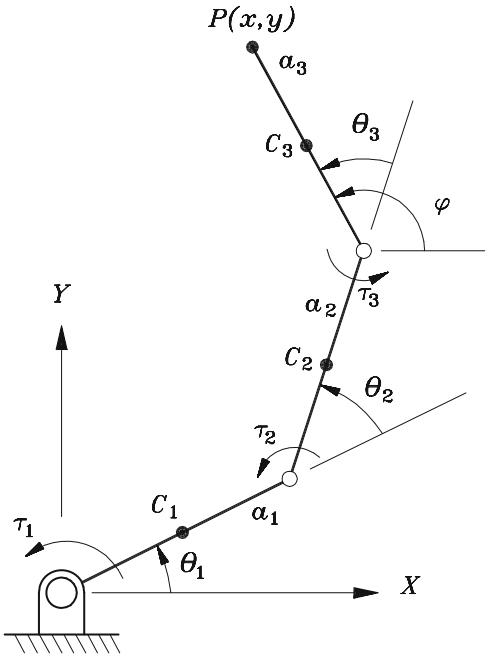

Fig. 7.1 A planar manipulator

$$\mathbf{I}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \dot{\mathbf{I}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \frac{1}{2}\left[\frac{\partial(\mathbf{I}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}\right]^T \dot{\boldsymbol{\theta}} + \frac{\partial V}{\partial \boldsymbol{\theta}} = \boldsymbol{\phi}_n \tag{7.20}$$

Example 7.3.1 (Euler–Lagrange Equations of a Planar Robot). Consider the manipulator of Fig. 7.1, with links designed so that their centers of mass, C_1, C_2, and C_3, are located at the midpoints of segments O_1O_2, O_2O_3, and O_3P, respectively. Moreover, the ith link has a mass m_i and a centroidal moment of inertia in a direction normal to the plane of motion I_i, while the joints are actuated by motors delivering torques τ_1, τ_2, and τ_3, the lubricant of the joints producing dissipative torques that we will neglect in this model. Under the assumption that gravity acts in the direction of $-Y$, find the associated Euler–Lagrange equations.

Solution: Here we recall the kinematic analysis of Sect. 5.7 and the definitions introduced therein for the analysis of planar motion. In this light, all vectors introduced below are two-dimensional, the scalar angular velocities of the links, ω_i, for $i = 1, 2, 3$, being

$$\omega_1 = \dot{\theta}_1, \quad \omega_2 = \dot{\theta}_1 + \dot{\theta}_2, \quad \omega_3 = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3$$

Moreover, the velocities of the centers of mass are

$$\dot{\mathbf{c}}_1 = \frac{1}{2}\dot{\theta}_1 \mathbf{E}\mathbf{a}_1$$

$$\dot{\mathbf{c}}_2 = \dot{\theta}_1 \mathbf{E}\mathbf{a}_1 + \frac{1}{2}(\dot{\theta}_1 + \dot{\theta}_2)\mathbf{E}\mathbf{a}_2$$

$$\dot{\mathbf{c}}_3 = \dot{\theta}_1 \mathbf{E}\mathbf{a}_1 + (\dot{\theta}_1 + \dot{\theta}_2)\mathbf{E}\mathbf{a}_2 + \frac{1}{2}(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)\mathbf{E}\mathbf{a}_3$$

the kinetic energy then becoming

$$T = \frac{1}{2}\sum_{1}^{3}(m_i \|\dot{\mathbf{c}}_i\|^2 + I_i \omega_i^2)$$

The squared magnitudes of the mass-center velocities are now computed using the expressions derived above. After simplifications, these yield

$$\|\dot{\mathbf{c}}_1\|^2 = \frac{1}{4}a_1^2\dot{\theta}_1^2$$

$$\|\dot{\mathbf{c}}_2\|^2 = a_1^2\dot{\theta}_1^2 + \frac{1}{4}a_2^2(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) + a_1a_2\cos\theta_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)$$

$$\begin{aligned}\|\dot{\mathbf{c}}_3\|^2 = {} & a_1^2\dot{\theta}_1^2 + a_2^2(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \\ & + \frac{1}{4}a_3^2(\dot{\theta}_1^2 + \dot{\theta}_2^2 + \dot{\theta}_3^2 + 2\dot{\theta}_1\dot{\theta}_2 + 2\dot{\theta}_1\dot{\theta}_3 + 2\dot{\theta}_2\dot{\theta}_3) \\ & + 2a_1a_2\cos\theta_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2) + a_1a_3\cos(\theta_2 + \theta_3)(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_1\dot{\theta}_3) \\ & + a_2a_3\cos\theta_3(\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_1\dot{\theta}_3 + \dot{\theta}_2\dot{\theta}_3)\end{aligned}$$

The kinetic energy of the whole manipulator thus becomes

$$T = \frac{1}{2}(I_{11}\dot{\theta}_1^2 + 2I_{12}\dot{\theta}_1\dot{\theta}_2 + 2I_{23}\dot{\theta}_2\dot{\theta}_3 + I_{22}\dot{\theta}_2^2 + 2I_{13}\dot{\theta}_1\dot{\theta}_3 + I_{33}\dot{\theta}_3^2)$$

with coefficients I_{ij}, for $i = 1, 2, 3$, and $j = i$ to 3 being the distinct entries of the 3×3 matrix of generalized inertia of the system. These entries are given below:

$$\begin{aligned}I_{11} \equiv {} & I_1 + I_2 + I_3 + \frac{1}{4}m_1a_1^2 + m_2\left(a_1^2 + \frac{1}{4}a_2^2 + a_1a_2c_2\right) \\ & + m_3\left(a_1^2 + a_2^2 + \frac{1}{4}a_3^2 + 2a_1a_2c_2 + a_1a_3c_{23} + a_2a_3c_3\right)\end{aligned}$$

$$I_{12} \equiv I_2 + I_3 + \frac{1}{2}\left[m_2\left(\frac{1}{2}a_2^2 + a_1a_2c_2\right)\right.$$

$$+ m_3\left(2a_2^2 + \frac{1}{2}a_3^2 + 2a_1a_2c_2 + a_1a_3c_{23} + 2a_2a_3c_3\right)\Big]$$

$$I_{13} \equiv I_3 + \frac{1}{2}m_3\left(\frac{1}{2}a_3^2 + a_1a_3c_{23} + a_2a_3c_3\right)$$

$$I_{22} \equiv I_2 + I_3 + \frac{1}{4}m_2a_2^2 + m_3\left(a_2^2 + \frac{1}{4}a_3^2 + a_2a_3c_3\right)$$

$$I_{23} \equiv I_3 + \frac{1}{2}m_3\left(\frac{1}{2}a_3^2 + a_2a_3c_3\right)$$

$$I_{33} \equiv I_3 + \frac{1}{4}m_3a_3^2$$

where c_i and c_{ij} stand for $\cos\theta_i$ and $\cos(\theta_i + \theta_j)$, respectively. From the foregoing expressions, it is apparent that the generalized inertia matrix is not a function of θ_1, which is only natural, for if the second and third joints are locked while leaving the first one free, the whole manipulator becomes a single rigid body pivoting about point O_1. Now, the polar moment of inertia of a rigid body in planar motion about a fixed point is constant, and hence, the first joint variable should not affect the generalized inertia matrix.

Furthermore, the potential energy of the manipulator is computed as the sum of the individual link potential energies, i.e.,

$$V = \frac{1}{2}m_1ga_1\sin\theta_1 + m_2g\left[a_1\sin\theta_1 + \frac{1}{2}a_2\sin(\theta_1 + \theta_2)\right]$$
$$+m_3g\left[a_1\sin\theta_1 + a_2\sin(\theta_1 + \theta_2) + \frac{1}{2}a_3\sin(\theta_1 + \theta_2 + \theta_3)\right]$$

while the total power delivered to the manipulator takes the form

$$\Pi = \tau_1\dot{\theta}_1 + \tau_2\dot{\theta}_2 + \tau_3\dot{\theta}_3$$

We now proceed to compute the various terms in Eq. (7.20). We already have $\mathbf{I}(\boldsymbol{\theta})$, but we do not have, as yet, its time-derivative. However, the entries of $\dot{\mathbf{I}}$ are merely the time-derivatives of the entries of $\mathbf{I}$. From the above expressions for these entries, their time-rates of change are readily calculated, namely,

$$\dot{I}_{11} = -m_2a_1a_2s_2\dot{\theta}_2 - m_3[2a_1a_2s_2\dot{\theta}_2 + a_1a_3s_{23}(\dot{\theta}_2 + \dot{\theta}_3) + a_2a_3s_3\dot{\theta}_3]$$

$$\dot{I}_{12} = \frac{1}{2}\{-m_2a_1a_2s_2\dot{\theta}_2 - m_3[2a_1a_2s_2\dot{\theta}_2 + a_1a_3s_{23}(\dot{\theta}_2 + \dot{\theta}_3) + 2a_2a_3s_3\dot{\theta}_3]\}$$

$$\dot{I}_{13} = -\frac{1}{2}m_3[a_1a_3s_{23}(\dot{\theta}_2 + \dot{\theta}_3) + a_2a_3s_3\dot{\theta}_3]$$

$$\dot{I}_{22} = -m_3 a_2 a_3 s_3 \dot{\theta}_3$$
$$\dot{I}_{23} = -\frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_3$$
$$\dot{I}_{33} = 0$$

with s_{ij} defined as $\sin(\theta_i + \theta_j)$. It should now be apparent that the time-rate of change of the generalized inertia matrix is independent of $\dot{\theta}_1$, as one should have expected, for this matrix is independent of θ_1. That is, if all joints but the first one are frozen, no matter how fast the first joint rotates, the manipulator moves as a single rigid body whose polar moment of inertia about O_1, the center of the first joint, is constant. As a matter of fact, I_{33} is constant for the same reason and $\dot{I}_{33}$ hence vanishes. We have, then,[2]

$$\dot{\mathbf{I}}\dot{\boldsymbol{\theta}} \equiv \boldsymbol{\iota} = \begin{bmatrix} \dot{I}_{11}\dot{\theta}_1 + \dot{I}_{12}\dot{\theta}_2 + \dot{I}_{13}\dot{\theta}_3 \\ \dot{I}_{12}\dot{\theta}_1 + \dot{I}_{22}\dot{\theta}_2 + \dot{I}_{23}\dot{\theta}_3 \\ \dot{I}_{13}\dot{\theta}_1 + \dot{I}_{23}\dot{\theta}_2 \end{bmatrix}$$

whose components, ι_i, for $i = 1, 2, 3$, are readily calculated as

$$\begin{aligned}
\iota_1 = & -[m_2 a_1 a_2 s_2 + m_3 a_1 (2a_2 s_2 + a_3 s_{23})]\dot{\theta}_1\dot{\theta}_2 - m_3 a_3 (a_1 s_{23} + a_2 s_3)\dot{\theta}_1\dot{\theta}_3 \\
& -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3 a_1 (2a_2 s_2 + a_3 s_{23})]\dot{\theta}_2^2 - m_3 a_3 (a_1 s_{23} + a_2 s_3)\dot{\theta}_2\dot{\theta}_3 \\
& -\frac{1}{2} m_3 a_3 (a_1 s_{23} + a_2 s_3)\dot{\theta}_3^2 \\
\iota_2 = & -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3 a_1 (2a_2 s_2 + a_3 s_{23})]\dot{\theta}_1\dot{\theta}_2 \\
& -\frac{1}{2} m_3 a_3 (a_1 s_{23} + a_2 s_3)\dot{\theta}_1\dot{\theta}_3 - m_3 a_2 a_3 s_3 \dot{\theta}_2\dot{\theta}_3 - \frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_3^2 \\
\iota_3 = & -\frac{1}{2} m_3 a_1 a_3 s_{23}\dot{\theta}_1\dot{\theta}_2 - \frac{1}{2} m_3 a_3 (a_1 s_{23} + a_2 s_3)\dot{\theta}_1\dot{\theta}_3 - \frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_2\dot{\theta}_3
\end{aligned}$$

The next term in the right-hand side of Eq. (7.20) now requires the calculation of the partial derivatives of vector $\mathbf{I}\dot{\boldsymbol{\theta}}$ with respect to the joint variables, which are computed below. Let

$$\frac{\partial(\mathbf{I}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \equiv \mathbf{I}'$$

[2] $\boldsymbol{\iota}$ is the Greek letter *iota* and denotes a vector; according to our notation, its components are ι_1, ι_2, and ι_3.

its entries being denoted by I'_{ij}. This matrix, in component form, is given by

$$\mathbf{I}' = \begin{bmatrix} 0 & I_{11,2}\dot{\theta}_1 + I_{12,2}\dot{\theta}_2 + I_{13,2}\dot{\theta}_3 & I_{11,3}\dot{\theta}_1 + I_{12,3}\dot{\theta}_2 + I_{13,3}\dot{\theta}_3 \\ 0 & I_{12,2}\dot{\theta}_1 + I_{22,2}\dot{\theta}_2 + I_{23,2}\dot{\theta}_3 & I_{12,3}\dot{\theta}_1 + I_{22,3}\dot{\theta}_2 + I_{23,3}\dot{\theta}_3 \\ 0 & I_{13,2}\dot{\theta}_1 + I_{23,2}\dot{\theta}_2 + I_{33,2}\dot{\theta}_3 & I_{13,3}\dot{\theta}_1 + I_{23,3}\dot{\theta}_2 + I_{33,3}\dot{\theta}_3 \end{bmatrix}$$

with the shorthand notation $I_{ij,k}$ indicating the partial derivative of I_{ij} with respect to θ_k. As the reader can verify, these entries are given as

$$\begin{aligned}
I'_{11} &= 0 \\
I'_{12} &= -[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_1 \\
&\quad -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_2 - \frac{1}{2}m_3 a_1 a_3 s_{23}\dot{\theta}_3 \\
I'_{13} &= -m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1 - \frac{1}{2}m_3(a_1 a_3 s_{23} + 2a_2 a_3 s_3)\dot{\theta}_2 \\
&\quad -\frac{1}{2}m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_3 \\
I'_{21} &= 0 \\
I'_{22} &= -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_1 \\
I'_{23} &= -\frac{1}{2}m_3(a_1 a_3 s_{23} + 2a_2 a_3 s_3)\dot{\theta}_1 - m_3 a_2 a_3 s_3 \dot{\theta}_2 - \frac{1}{2}m_3 a_2 a_3 s_3 \dot{\theta}_3 \\
I'_{31} &= 0 \\
I'_{32} &= -\frac{1}{2}m_3 a_1 a_3 s_{23}\dot{\theta}_1 \\
I'_{33} &= -\frac{1}{2}m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1 - \frac{1}{2}m_3 a_2 a_3 s_3 \dot{\theta}_2
\end{aligned}$$

Now, we define the three-dimensional vector $\boldsymbol{\gamma}$ below:

$$\boldsymbol{\gamma} \equiv \left[\frac{\partial(\mathbf{I}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}\right]^T \dot{\boldsymbol{\theta}}$$

its three components, γ_i, for $i = 1, 2, 3$, being

$$\begin{aligned}
\gamma_1 &= 0 \\
\gamma_2 &= -[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_1^2 \\
&\quad -[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_1 \dot{\theta}_2
\end{aligned}$$

$$\begin{aligned}\gamma_3 = &-m_3 a_1 a_3 s_{23} \dot{\theta}_1 \dot{\theta}_3 \\ &-m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1^2 - m_3(a_1 a_3 s_{23} + 2a_2 a_3 s_3)\dot{\theta}_1 \dot{\theta}_2 \\ &-m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1 \dot{\theta}_3 - m_3 a_2 a_3 s_3 \dot{\theta}_3^2 - m_3 a_2 a_3 s_3 \dot{\theta}_2 \dot{\theta}_3\end{aligned}$$

We now turn to the computation of the partial derivatives of the potential energy:

$$\begin{aligned}\frac{\partial V}{\partial \theta_1} &= \frac{1}{2} m_1 g a_1 c_1 + m_2 g \left(a_1 c_1 + \frac{1}{2} a_2 c_{12}\right) + m_3 g \left(a_1 c_1 + a_2 c_{12} + \frac{1}{2} a_3 c_{123}\right) \\ \frac{\partial V}{\partial \theta_2} &= \frac{1}{2} m_2 g a_2 c_{12} + m_3 g \left(a_2 c_{12} + \frac{1}{2} a_3 c_{123}\right) \\ \frac{\partial V}{\partial \theta_3} &= \frac{1}{2} m_3 g a_3 c_{123}\end{aligned}$$

The Euler–Lagrange equations thus reduce to

$$\begin{aligned}I_{11}\ddot{\theta}_1 + I_{12}\ddot{\theta}_2 + I_{13}\ddot{\theta}_3 + \iota_1 - \frac{1}{2}\gamma_1 + \frac{1}{2} m_1 g a_1 c_1 \\ + m_2 g(a_1 c_1 + \frac{1}{2} a_2 c_{12}) + m_3 g(a_1 c_1 + a_2 c_{12} + \frac{1}{2} a_3 c_{123}) = \tau_1 \\ I_{12}\ddot{\theta}_1 + I_{22}\ddot{\theta}_2 + I_{23}\ddot{\theta}_3 + \iota_2 - \frac{1}{2}\gamma_2 + \frac{1}{2} m_2 g a_2 c_{12} \\ + m_3 g(a_2 c_{12} + \frac{1}{2} a_3 c_{123}) = \tau_2 \\ I_{13}\ddot{\theta}_1 + I_{23}\ddot{\theta}_2 + I_{33}\ddot{\theta}_3 + \iota_3 - \frac{1}{2}\gamma_3 + \frac{1}{2} m_3 g a_3 c_{123} = \tau_3\end{aligned}$$

With this example, it becomes apparent that a straightforward differentiation procedure to derive the Euler–Lagrange equations of a robotic manipulator, or for that matter, of a mechanical system at large, is not practical. For example, these equations do not seem to lend themselves to symbolic manipulations for a six-axis manipulator of arbitrary architecture, given that they become quite cumbersome even for a three-axis planar manipulator with an architecture that is not so general. For this reason, procedures have been devised that lend themselves to an algorithmic treatment. We will study a procedure based on the *natural orthogonal complement* whereby the underlying equations are derived using matrix-times-vector multiplications.

7.3.3 Kane's Equations

Kane's equations (Kane and Levinson 1983), sometimes referred to as *D'Alembert's equations in Lagrangian form* are also useful in robot dynamics (Angeles et al. 1989).

A feature of Kane's equations is that they are derived from the free-body diagrams of the various rigid bodies constituting the multibody system at hand. Upon introducing generalized coordinates à la Lagrange, the mathematical model of the system is derived, which is equivalent to the underlying Euler–Lagrange equations. Kane's equations take a rather simple form, for an n-dof mechanical system, namely,

$$\boldsymbol{\phi} + \boldsymbol{\phi}^* = \mathbf{0}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\phi}^*$ are the n-dimensional vectors of generalized *active force* and *inertia force*, respectively. With the notation introduced above, these vectors are given by

$$\boldsymbol{\phi} = \sum_{i=1}^{r} \left[\left(\frac{\partial \dot{\mathbf{c}}_i}{\partial \dot{\mathbf{q}}} \right)^T \mathbf{f}_i + \left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{\mathbf{q}}} \right)^T \mathbf{n}_i \right] \tag{7.21a}$$

and

$$\boldsymbol{\phi}^* = -\sum_{i=1}^{r} \left[\left(\frac{\partial \dot{\mathbf{c}}_i}{\partial \dot{\mathbf{q}}} \right)^T m_i \ddot{\mathbf{c}}_i + \left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{\mathbf{q}}} \right)^T (\mathbf{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i) \right]. \tag{7.21b}$$

In the above expressions, $\dot{\mathbf{q}} = d\mathbf{q}/dt$ is the n-dimensional vector of *generalized speeds* in Kane's terminology, while the $n \times 3$ matrices $\partial \dot{\mathbf{c}}_i / \partial \dot{\mathbf{q}}$ and $\partial \boldsymbol{\omega}_i / \partial \dot{\mathbf{q}}$ are the *partial rates of change* of mass-center velocity and angular velocity of the ith rigid body.

7.4 Recursive Inverse Dynamics

The inverse dynamics problem associated with serial manipulators is studied here. We assume at the outset that the manipulator under study is of the serial type with $n+1$ links including the base link and n joints of either the revolute or the prismatic type.

The underlying algorithm consists of two steps: (a) *kinematic computations*, required to determine the twists of all the links and their time derivatives in terms of $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$; and (b) *dynamic computations*, required to determine both the constraint and the external wrenches. Each of these steps is described below, the aim here being to calculate the desired variables with as few computations as possible, for one purpose of inverse dynamics is to permit the real-time model-based control of the manipulator. Real-time performance requires, obviously, a low number of computations. For the sake of simplicity, we decided against discussing the algorithms with the lowest computational cost, mainly because these algorithms, fully discussed by Balafoutis and Patel (1991), rely heavily on tensor calculus, which we have not studied here. With the notation introduced in Chap. 4, revolute joints are referred to as R, prismatic joints as P.

7.4.1 Kinematics Computations: Outward Recursions

We will use the Denavit–Hartenberg (DH) notation introduced in Sect. 4.2 and hence will refer to Fig. 4.9 for the basic notation required for the kinematic analysis to be described first. Note that the calculation of each $\mathbf{Q}_i$ matrix, as given by Eq. (4.1e), requires four multiplications and zero additions.

Moreover, *every three-dimensional vector-component transfer from the $\mathcal{F}_i$ frame to the $\mathcal{F}_{i+1}$ frame requires a multiplication by $\mathbf{Q}_i^T$. Likewise, every component transfer from the $\mathcal{F}_{i+1}$ frame to the $\mathcal{F}_i$ frame requires a multiplication by $\mathbf{Q}_i$.* Therefore, we will need to account for the aforementioned component transfers, which we will generically term *coordinate transformations* between successive coordinate frames. We derive below the number of operations required for such transformations. If we have $[\mathbf{r}]_i \equiv [r_1, r_2, r_3]^T$ and we need $[\mathbf{r}]_{i+1}$, then we proceed as follows:

$$[\mathbf{r}]_{i+1} = \mathbf{Q}_i^T [\mathbf{r}]_i \tag{7.22}$$

and if we recall the form of $\mathbf{Q}_i$ from Eq. (4.1e), we then have

$$[\mathbf{r}]_{i+1} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\lambda_i \sin\theta_i & \lambda_i \cos\theta_i & \mu_i \\ \mu_i \sin\theta_i & -\mu_i \cos\theta_i & \lambda_i \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} r_1 \cos\theta_i + r_2 \sin\theta_i \\ -\lambda_i r + \mu_i r_3 \\ \mu_i r + \lambda_i r_3 \end{bmatrix} \tag{7.23a}$$

where $\lambda_i \equiv \cos\alpha_i$ and $\mu_i \equiv \sin\alpha_i$, while

$$r \equiv r_1 \sin\theta_i - r_2 \cos\theta_i \tag{7.23b}$$

Likewise, if we have $[\mathbf{v}]_{i+1} \equiv [v_1, v_2, v_3]^T$ and we need $[\mathbf{v}]_i$, we use the component transformation given below:

$$[\mathbf{v}]_i = \begin{bmatrix} \cos\theta_i & -\lambda_i \sin\theta_i & \mu_i \sin\theta_i \\ \sin\theta_i & \lambda_i \cos\theta_i & -\mu_i \cos\theta_i \\ 0 & \mu_i & \lambda_i \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1 \cos\theta_i - v \sin\theta_i \\ v_1 \sin\theta_i + v \cos\theta_i \\ v_2\mu_i + v_3\lambda_i \end{bmatrix} \tag{7.24a}$$

where

$$v \equiv v_2\lambda_i - v_3\mu_i \tag{7.24b}$$

It is now apparent that *every coordinate transformation between successive frames, whether forward or backward, requires eight multiplications and four additions.* Here, as in Chap. 5, we indicate the units of multiplications and additions with M and A, respectively.

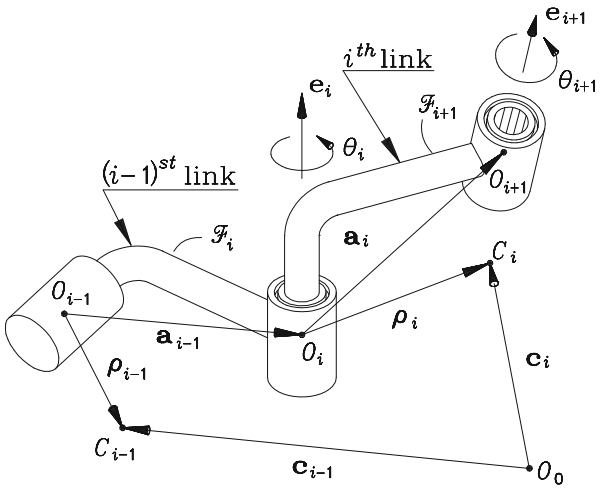

Fig. 7.2 A revolute joint

The angular velocity and acceleration of the ith link are computed recursively as follows:

$$\boldsymbol{\omega}_i = \begin{cases} \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i, & \text{if the } i\text{th joint is } R \\ \boldsymbol{\omega}_{i-1}, & \text{if the } i\text{th joint is } P \end{cases} \tag{7.25a}$$

$$\dot{\boldsymbol{\omega}}_i = \begin{cases} \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \dot{\theta}_i \mathbf{e}_i + \ddot{\theta}_i \mathbf{e}_i, & \text{if the } i\text{th joint is } R \\ \dot{\boldsymbol{\omega}}_{i-1}, & \text{if the } i\text{th joint is } P \end{cases} \tag{7.25b}$$

for $i = 1, 2, \ldots, n$, where $\boldsymbol{\omega}_0$ and $\dot{\boldsymbol{\omega}}_0$ are the angular velocity and angular acceleration of the base link. Note that Eqs. (7.25a and b) are frame-invariant; i.e., they are valid in *any* coordinate frame, as long as the same frame is used to represent all quantities involved. Below we derive the equivalent relations applicable when taking into account that quantities with a subscript i are available in $\mathcal{F}_{i+1}$-coordinates. Hence, operations involving quantities with different subscripts require a change of coordinates, which is taken care of by the corresponding rotation matrices.

In order to reduce the numerical complexity of the algorithm developed here, all vector and matrix quantities of the ith link will be expressed in $\mathcal{F}_{i+1}$. Note, however, that the two vectors $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ are fixed to the ith link, which is a potential source of confusion. Now, since $\mathbf{e}_i$ has an extremely simple form in $\mathcal{F}_i$, namely, $[0, 0, 1]^T$, this will be regarded as a vector of the $(i-1)$st link, which is fixed to $\mathcal{F}_i$—see, e.g., Fig. 7.2. Therefore, this vector, or multiples of it, will be added to vectors bearing the $(i-1)$st subscript without any coordinate transformation. Moreover, subscripted brackets, as introduced in Sect. 2.2, can be avoided if all

vector and matrix quantities subscripted with i, except for vector $\mathbf{e}_i$, are assumed to be expressed in $\mathcal{F}_{i+1}$. Furthermore, in view of the serial type of the underlying kinematic chain, only additions of quantities with two successive subscripts will appear in the relations below.

Quantities given in two successive frames can be added if both are expressed in the same frame, the obvious frame of choice being the frame of one of the two quantities. Hence, all we need to add two quantities with successive subscripts is to multiply one of these by a suitable orthogonal matrix. Additionally, in view of the *outwards* recursive nature of the foregoing kinematic relations, it is apparent that a transfer from $\mathcal{F}_i$- to $\mathcal{F}_{i+1}$-coordinates is needed, which can be accomplished by multiplying either $\mathbf{e}_i$ or any other vector with the $(i-1)$ subscript by matrix $\mathbf{Q}_i^T$. Hence, the angular velocities and accelerations are computed recursively, as indicated below:

$$\boldsymbol{\omega}_i = \begin{cases} \mathbf{Q}_i^T(\boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i), & \text{if the } i\text{th joint is } R \\ \mathbf{Q}_i^T \boldsymbol{\omega}_{i-1}, & \text{if the } i\text{th joint is } P \end{cases} \tag{7.26a}$$

$$\dot{\boldsymbol{\omega}}_i = \begin{cases} \mathbf{Q}_i^T(\dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \dot{\theta}_i \mathbf{e}_i + \ddot{\theta}_i \mathbf{e}_i), & \text{if the } i\text{th joint is } R \\ \mathbf{Q}_i^T \dot{\boldsymbol{\omega}}_{i-1}, & \text{if the } i\text{th joint is } P \end{cases} \tag{7.26b}$$

If the base link is an inertial frame, then

$$\boldsymbol{\omega}_0 = \mathbf{0}, \qquad \dot{\boldsymbol{\omega}}_0 = \mathbf{0} \tag{7.27}$$

Thus, calculating each $\boldsymbol{\omega}_i$ vector in $\mathcal{F}_{i+1}$ when $\boldsymbol{\omega}_{i-1}$ is given in $\mathcal{F}_i$ requires $8M$ and $5A$ if the ith joint is R; if it is P, the said calculation reduces to $8M$ and $4A$. Here, note that $\dot{\theta}_i \mathbf{e}_i = [\,0,\ 0,\ \dot{\theta}_i\,]^T$ in $\mathcal{F}_i$-coordinates, and hence, the vector addition of the upper right-hand side of Eq. (7.26a) requires only $1A$. Furthermore, in order to determine the number of operations required to calculate $\dot{\boldsymbol{\omega}}_i$ in $\mathcal{F}_{i+1}$ when $\dot{\boldsymbol{\omega}}_{i-1}$ is available in $\mathcal{F}_i$, we note that

$$[\,\mathbf{e}_i\,]_i = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{7.28}$$

Moreover, we let

$$[\,\boldsymbol{\omega}_{i-1}\,]_i = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{7.29}$$

Hence,

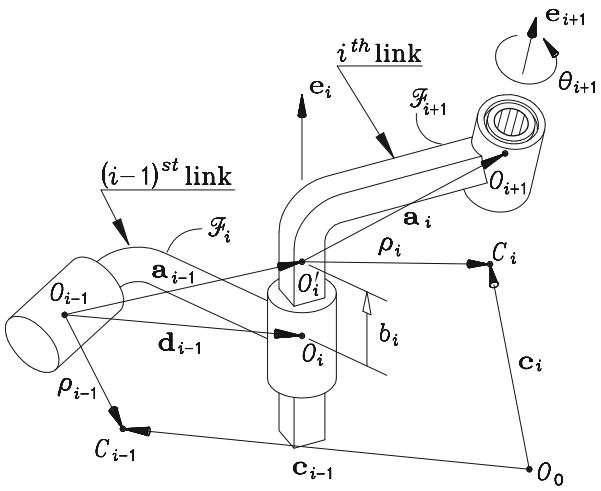

Fig. 7.3 A prismatic joint

$$[\boldsymbol{\omega}_{i-1} \times \dot{\theta}_i \mathbf{e}_i]_i = \begin{bmatrix} \dot{\theta}_i\, \omega_y \\ -\dot{\theta}_i\, \omega_x \\ 0 \end{bmatrix} \tag{7.30}$$

Furthermore, we note that

$$[\ddot{\theta}_i \mathbf{e}_i]_i = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_i \end{bmatrix} \tag{7.31}$$

and hence, the calculation of $\dot{\boldsymbol{\omega}}_i$ in $\mathcal{F}_{i+1}$ when $\dot{\boldsymbol{\omega}}_{i-1}$ is given in $\mathcal{F}_i$ requires $10M$ and $7A$ if the ith joint is R; if it is P, the same calculation requires $8M$ and $4A$.

Furthermore, let $\mathbf{c}_i$ be the position vector of C_i, the center of mass of the ith link, $\boldsymbol{\rho}_i$ being the vector directed from O_i to C_i, as shown in Figs. 7.2 and 7.3. The position vectors of two successive centers of mass thus observe the relationships

(a) if the ith joint is R, then from Fig. 7.2,

$$\boldsymbol{\delta}_{i-1} \equiv \mathbf{a}_{i-1} - \boldsymbol{\rho}_{i-1} \tag{7.32a}$$

$$\mathbf{c}_i = \mathbf{c}_{i-1} + \boldsymbol{\delta}_{i-1} + \boldsymbol{\rho}_i \tag{7.32b}$$

(b) if the ith joint is P, then from Fig. 7.3,

$$\boldsymbol{\delta}_{i-1} \equiv \mathbf{d}_{i-1} - \boldsymbol{\rho}_{i-1} \tag{7.32c}$$

$$\mathbf{c}_i = \mathbf{c}_{i-1} + \boldsymbol{\delta}_{i-1} + b_i \mathbf{e}_i + \boldsymbol{\rho}_i \tag{7.32d}$$

where point O_i, in this case, is a point of the $(i-1)$st link conveniently defined, as dictated by the particular geometry of the manipulator at hand. The foregoing freedom in the choice of O_i is a consequence of prismatic pairs having only a defined direction but no axis, properly speaking.

Notice that in the presence of a revolute pair at the ith joint, the difference $\mathbf{a}_{i-1} - \boldsymbol{\rho}_{i-1}$ is constant in $\mathcal{F}_i$. Likewise, in the presence of a prismatic pair at the same joint, the difference $\mathbf{d}_{i-1} - \boldsymbol{\rho}_{i-1}$ is constant in $\mathcal{F}_i$. Therefore, these differences are computed off-line, their evaluation not counting toward the computational complexity of the algorithm.

Upon differentiation of both sides of Eqs. (7.32b and d) with respect to time, we derive the corresponding relations between the velocities and accelerations of the centers of mass links $i-1$ and i, namely,

(a) if the ith joint is R,

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{\delta}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{\rho}_i \tag{7.33a}$$

$$\ddot{\mathbf{c}}_i = \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{\delta}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \boldsymbol{\delta}_{i-1}) + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{\rho}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{\rho}_i) \tag{7.33b}$$

(b) if the ith joint is P,[3]

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \tag{7.34a}$$

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} \tag{7.34b}$$

$$\mathbf{u}_i \equiv \boldsymbol{\delta}_{i-1} + b_i \mathbf{e}_i + \boldsymbol{\rho}_i \tag{7.34c}$$

$$\mathbf{v}_i \equiv \boldsymbol{\omega}_i \times \mathbf{u}_i \tag{7.34d}$$

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_{i-1} + \mathbf{v}_i + \dot{b}_i \mathbf{e}_i \tag{7.34e}$$

$$\ddot{\mathbf{c}}_i = \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{u}_i + \boldsymbol{\omega}_i \times (\mathbf{v}_i + 2\dot{b}_i \mathbf{e}_i) + \ddot{b}_i \mathbf{e}_i \tag{7.34f}$$

for $i = 1, 2, \ldots, n$, where $\dot{\mathbf{c}}_0$ and $\ddot{\mathbf{c}}_0$ are the velocity and acceleration of the center of mass the base link. If the latter is an inertial frame, then

$$\boldsymbol{\omega}_0 = \mathbf{0}, \quad \dot{\boldsymbol{\omega}}_0 = \mathbf{0}, \quad \dot{\mathbf{c}}_0 = \mathbf{0}, \quad \ddot{\mathbf{c}}_0 = \mathbf{0} \tag{7.35}$$

Expressions (7.32b)–(7.34f) are *invariant*, i.e., they hold in *any* coordinate frame, as long as all vectors involved are expressed in that frame. However, we have vectors that are naturally expressed in the $\mathcal{F}_i$ frame added to vectors

[3] The relations below are made apparent with the aid of Fig. 4.6.

expressed in the $\mathcal{F}_{i+1}$ frame, and hence, a coordinate transformation is needed. This coordinate transformation is taken into account in Algorithm 7.4.1, whereby the logical variable `R` is `true` if the ith joint is R; otherwise it is `false`.

In performing the foregoing calculations, we need the cross product of a vector $\mathbf{w}$ times $\mathbf{e}_i$ in $\mathcal{F}_i$ coordinates, the latter being simply $[\,\mathbf{e}_i\,]_i = [\,0,\ 0,\ 1\,]^T$, and hence, this cross product reduces to $[\,w_2,\ -w_1,\ 0\,]^T$, whereby w_k, for $k = 1, 2, 3$, are the x, y, and z $\mathcal{F}_i$-components of $\mathbf{w}$. This cross product, then, requires no multiplications and no additions. Likewise, vectors $b_i\mathbf{e}_i$, $\dot{b}_i\mathbf{e}_i$, and $\ddot{b}_i\mathbf{e}_i$ take the simple forms $[\,0,\ 0,\ b_i\,]^T$, $[\,0,\ 0,\ \dot{b}_i\,]^T$, and $[\,0,\ 0,\ \ddot{b}_i\,]^T$ in $\mathcal{F}_i$. Adding any of these vectors to any other vector in $\mathcal{F}_i$ then requires one single addition.

Algorithm 7.4.1 (Outward Recursions):

```
read {Q_i}_0^{n-1}, c_0, ω_0, ċ_0, ω̇_0, c̈_0, {ρ_i}_1^n, {δ_i}_0^{n-1}
For  i = 1 to n step 1 do
     update Q_i
     if R then
       c_i   ←  Q_i^T (c_{i-1} + δ_{i-1}) + ρ_i
       ω_i   ←  Q_i^T (ω_{i-1} + θ̇_i e_i)
       u_{i-1} ← ω_{i-1} × δ_{i-1}
       v_i   ←  ω_i × ρ_i
       ċ_i   ←  Q_i^T (ċ_{i-1} + u_{i-1}) + v_i
       ω̇_i   ←  Q_i^T (ω̇_{i-1} + ω_{i-1} × θ̇_i e_i + θ̈_i e_i)
       c̈_i   ←  Q_i^T (c̈_{i-1} + ω̇_{i-1} × δ_{i-1} + ω_{i-1} × u_{i-1})
                +ω̇_i × ρ_i + ω_i × v_i
     else
       u_i   ←  Q_i^T δ_{i-1} + ρ_i + b_i e_i
       c_i   ←  Q_i^T c_{i-1} + u_i
       ω_i   ←  Q_i^T ω_{i-1}
       v_i   ←  ω_i × u_i
       w_i   ←  ḃ_i e_i
       ċ_i   ←  Q_i^T ċ_{i-1} + v_i + w_i
       ω̇_i   ←  Q_i^T ω̇_{i-1}
       c̈_i   ←  Q_i^T c̈_{i-1} + ω̇_i × u_i + ω_i × (v_i + w_i + w_i) + b̈_i e_i
     endif
enddo
```

If, moreover, we take into account that the cross product of two arbitrary vectors requires $6M$ and $3A$, we then have the operation counts given below:

(a) If the ith joint is R,
$\mathbf{Q}_i$ requires $4M$ and $0A$
$\mathbf{c}_i$ requires $8M$ and $10A$
$\boldsymbol{\omega}_i$ requires $8M$ and $5A$

Table 7.1 Complexity of the kinematics computations

Item	M	A
$\{\mathbf{Q}_i\}_1^n$	$4n$	0
$\{\boldsymbol{\omega}_i\}_1^n$	$8n$	$5n$
$\{\dot{\mathbf{c}}_i\}_1^n$	$20n$	$16n$
$\{\dot{\boldsymbol{\omega}}_i\}_1^n$	$10n$	$7n$
$\{\ddot{\mathbf{c}}_i\}_1^n$	$32n$	$28n$
Total	$82n$	$66n$

$\dot{\mathbf{c}}_i$ requires $20M$ and $16A$
$\dot{\boldsymbol{\omega}}_i$ requires $10M$ and $7A$
$\ddot{\mathbf{c}}_i$ requires $32M$ and $28A$

(b) If the ith joint is P,
$\mathbf{Q}_i$ requires $4M$ and $0A$
$\mathbf{c}_i$ requires $16M$ and $15A$
$\boldsymbol{\omega}_i$ requires $8M$ and $4A$
$\dot{\mathbf{c}}_i$ requires $14M$ and $11A$
$\dot{\boldsymbol{\omega}}_i$ requires $8M$ and $4A$
$\ddot{\mathbf{c}}_i$ requires $20M$ and $19A$

The computational complexity for the forward recursions of the kinematics calculations for an n-revolute manipulator, as pertaining to various algorithms, are summarized in Table 7.1. Note that if some joints are P, then these figures become lower.

7.4.2 Dynamics Computations: Inward Recursions

The free-body diagram of an intermediate link appears in Fig. 7.4, that of the end-effector, or nth link, appearing in Fig. 7.5. Note that the EE is acted upon by a non-working constraint wrench, exerted through the nth pair, and a working wrench; the latter involves both active and dissipative forces and moments. Although dissipative forces and moments are difficult to model because of dry friction and striction, they can be readily incorporated into the dynamics model, once a suitable *constitutive model* for these items is available. Since these forces and moments depend only on joint variables and joint rates, they can be calculated once the kinematic variables are known. For the sake of simplicity, dissipative wrenches are not included here, their discussion being the subject of Sect. 7.8. Hence, the force and the moment that the $(i-1)$st link exerts on the ith link through the ith joint only produce nonworking constraint and active wrenches. That is, for a revolute pair, one has

$$\mathbf{n}_i^P = \begin{bmatrix} n_i^x \\ n_i^y \\ \tau_i \end{bmatrix}, \quad \mathbf{f}_i^P = \begin{bmatrix} f_i^x \\ f_i^y \\ f_i^z \end{bmatrix} \tag{7.36}$$

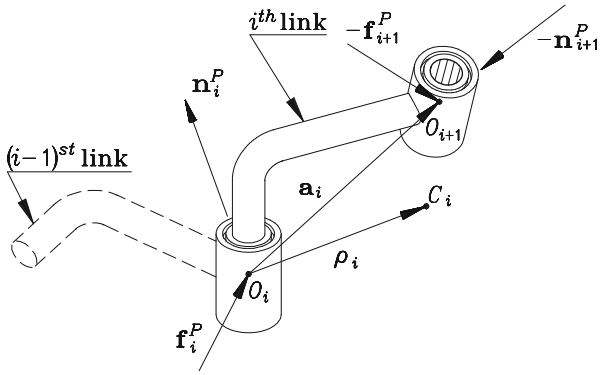

Fig. 7.4 Free-body diagram of the ith link

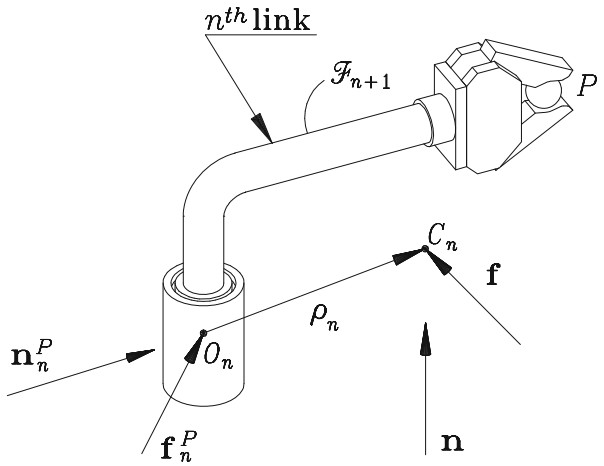

Fig. 7.5 Free-body diagram of the end-effector

in which n_i^x and n_i^y are the nonzero $\mathcal{F}_i$-components of the nonworking constraint moment exerted by the $(i-1)$st link on the ith link; obviously, this moment lies in a plane perpendicular to Z_i, whereas τ_i is the active torque applied by the motor at the said joint. Vector $\mathbf{f}_i^P$ contains only nonworking constraint forces.

For a prismatic pair, one has

$$\mathbf{n}^P = \begin{bmatrix} n_i^x \\ n_i^y \\ n_i^z \end{bmatrix}, \quad \mathbf{f}^P = \begin{bmatrix} f_i^x \\ f_i^y \\ \tau_i \end{bmatrix} \tag{7.37}$$

where vector $\mathbf{n}_i^P$ contains only nonworking constraint torques, while τ_i is now the active force exerted by the ith motor in the Z_i direction, f_i^x and f_i^y being the nonzero $\mathcal{F}_i$-components of the nonworking constraint force exerted by the ith joint on the ith link, which is perpendicular to the Z_i axis.

In the algorithm below, the driving torques or forces $\{\tau_i\}_1^n$, are computed via vectors $\mathbf{n}_i^P$ and $\mathbf{f}_i^P$. In fact, in the case of a revolute pair, τ_i is simply the third component of $\mathbf{n}_i^P$; in the case of a prismatic pair, τ_i is, accordingly, the third component of $\mathbf{f}_i^P$. From Fig. 7.5, the Newton–Euler equations of the end-effector are

$$\mathbf{f}_n^P = m_n \ddot{\mathbf{c}}_n - \mathbf{f} \tag{7.38a}$$

$$\mathbf{n}_n^P = \mathbf{I}_n \dot{\boldsymbol{\omega}}_n + \boldsymbol{\omega}_n \times \mathbf{I}_n \boldsymbol{\omega}_n - \mathbf{n} + \boldsymbol{\rho}_n \times \mathbf{f}_n^P \tag{7.38b}$$

where $\mathbf{f}$ and $\mathbf{n}$ are the external force and moment, the former being applied at the center of mass the end-effector. The Newton–Euler equations for the remaining links are derived based on the free-body diagram of Fig. 7.4, namely,

$$\mathbf{f}_i^P = m_i \ddot{\mathbf{c}}_i + \mathbf{f}_{i+1}^P \tag{7.38c}$$

$$\mathbf{n}_i^P = \mathbf{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i + \mathbf{n}_{i+1}^P + \boldsymbol{\delta}_i \times \mathbf{f}_{i+1}^P + \boldsymbol{\rho}_i \times \mathbf{f}_i^P \tag{7.38d}$$

with $\boldsymbol{\delta}_i$ defined as the difference $\mathbf{a}_i - \boldsymbol{\rho}_i$ in Eqs. (7.32a and c).

Once the $\mathbf{n}_i^P$ and $\mathbf{f}_i^P$ vectors are available, the actuator torques and forces, denoted by τ_i, are readily computed. In fact, if the ith joint is a revolute, then

$$\tau_i = \mathbf{e}_i^T \mathbf{n}_i^P \tag{7.39}$$

which does not require any further operations, for τ_i reduces, in this case, to the Z_i component of vector $\mathbf{n}_i^P$. Similarly, if the ith joint is prismatic, then the corresponding actuator force reduces to

$$\tau_i = \mathbf{e}_i^T \mathbf{f}_i^P \tag{7.40}$$

Again, the foregoing relations are written in invariant form. In order to perform the computations involved, transformations that transfer coordinates between two successive frames are required. Here, we have to keep in mind that the components of a vector expressed in the $(i+1)$st frame can be transferred to the ith frame by multiplying the vector array in $(i+1)$st coordinates by matrix $\mathbf{Q}_i$. In taking these coordinate transformations into account, we derive the Newton–Euler algorithm from the above equations, namely,

Table 7.2 Complexity of dynamics computations

Row #	M	A
1	3	3
2	30	27
5	$8(n-1)$	$4(n-1)$
6	$3(n-1)$	$3(n-1)$
7	$44(n-1)$	$37(n-1)$
Total	$55n-22$	$44n-14$

Algorithm 7.4.2 (Inward Recursions):

$\mathbf{f}_n^P \leftarrow m_n\ddot{\mathbf{c}}_n - \mathbf{f}$
$\mathbf{n}_n^P \leftarrow \mathbf{I}_n\dot{\boldsymbol{\omega}}_n + \boldsymbol{\omega}_n \times \mathbf{I}_n\boldsymbol{\omega}_n - \mathbf{n} + \boldsymbol{\rho}_n \times \mathbf{f}_n^P$
`If R then`
$\tau_n \leftarrow (\mathbf{Q}_n\mathbf{n}_n^P)_z$
`else`
$\tau_n \leftarrow (\mathbf{Q}_n\mathbf{f}_n^P)_z$
`For i=n-1 to 1 step −1 do`
$\boldsymbol{\phi}_{i+1} \leftarrow \mathbf{Q}_{i+1}\mathbf{f}_{i+1}^P$
$\mathbf{f}_i^P \leftarrow m_i\ddot{\mathbf{c}}_i + \boldsymbol{\phi}_{i+1}$
$\mathbf{n}_i^P \leftarrow \mathbf{I}_i\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i + \boldsymbol{\rho}_i \times \mathbf{f}_i^P + \mathbf{Q}_{i+1}\mathbf{n}_{i+1}^P + \boldsymbol{\delta}_i \times \boldsymbol{\phi}_{i+1}$
`If R then`
$\tau_i \leftarrow (\mathbf{Q}_i\mathbf{n}_i^P)_z$
`else`
$\tau_i \leftarrow (\mathbf{Q}_i\mathbf{f}_i^P)_z$ `enddo`

Note that, within the do-loop of the foregoing algorithm, the vectors to the left of the arrow are expressed in the ith frame, while $\mathbf{f}_{i+1}^P$ and $\mathbf{n}_{i+1}^P$, to the right of the arrow, are expressed in the $(i+1)$st frame.

In calculating the computational complexity of this algorithm, note that the $\mathbf{a}_i - \boldsymbol{\rho}_i$ term is constant in the $(i+1)$st frame, and hence, it is computed *off-line*. Thus, its computation need not be accounted for. A summary of computational costs is given in Table 7.2 for an n-revolute manipulator, with the row number[4] indicating the step in Algorithm 7.4.2.

The total numbers of multiplications M_d and additions A_d required by the foregoing algorithm are readily obtained, with the result shown below:

$$M_d = 55n - 22, \quad A_d = 44n - 14 \tag{7.41}$$

In particular, for a six-revolute manipulator, one has

$$n = 6, \quad M_d = 308, \quad A_d = 250 \tag{7.42}$$

[4]Only rows involving floating-point operations are counted here.

Table 7.3 Complexity of different algorithms for inverse dynamics

Author(s)	Methods	Multiplications	Additions
Hollerbach (1980)	E–L	$412n - 277$	$320n - 201$
Luh et al. (1980)	N–E	$150n - 48$	$131n - 48$
Walker and Orin (1982)	N–E	$137n - 22$	$101n - 11$
Khalil et al. (1986)	N–E	$105n - 92$	$94n - 86$
Angeles et al. (1989)	Kane	$105n - 109$	$90n - 105$
Balafoutis and Patel (1991)	Tensor	$93n - 69$	$81n - 65$
Li and Sankar (1992)	E–L	$88n - 69$	$76n - 66$

If the kinematics computations are accounted for, then the Newton–Euler algorithm given above for the inverse dynamics of n-revolute manipulators requires M multiplications and A additions, as given below:

$$M = 137n - 22, \quad A = 110n - 14 \tag{7.43}$$

The foregoing number of multiplications is identical to that reported by Walker and Orin (1982); however, the number of additions is slightly higher than Walker and Orin's figure, namely, $101n - 11$.

Thus, the inverse dynamics of a six-revolute manipulator requires 800 multiplications and 646 additions. These computations can be performed in a few microseconds using a modern processor. Clearly, if the aforementioned algorithms are tailored to suit particular architectures, then they can be further simplified. Note that, in the presence of a prismatic pair in the jth joint, the foregoing complexity is reduced. In fact, if this is the case, the Newton–Euler equations for the jth link remain as in Eqs. (7.38c and d) for the ith link, the only difference appearing in the implementing algorithm, which is simplified, in light of the results derived in discussing the kinematics calculations.

The incorporation of gravity in the Newton–Euler algorithm is done most economically by following the idea proposed by Luh et al. (1980), namely, by declaring that the inertial base undergoes an acceleration $-\mathbf{g}$, where $\mathbf{g}$ denotes the acceleration of gravity. That is

$$\ddot{\mathbf{c}}_0 = -\mathbf{g} \tag{7.44}$$

the gravitational accelerations thus propagating forward to the EE. A comparison of various algorithms with regard to their computational complexity is displayed in Table 7.3 for an n-revolute manipulator. For $n = 6$, the corresponding figures appear in Table 7.4.

Table 7.4 Complexity of different algorithms for inverse dynamics, for $n = 6$

Author(s)	Methods	Multiplications ($n = 6$)	Additions ($n = 6$)
Hollerbach (1980)	E–L	2195	1719
Luh et al. (1980)	N–E	852	738
Walker and Orin (1982)	N–E	800	595
Hollerbach and Sahar (1983)	N–E	688	558
Kane and Levinson (1983)	Kane	646	394
Khalil et al. (1986)	N–E	538	478
Angeles et al. (1989)	Kane	521	435
Balafoutis and Patel (1991)	Tensor	489	420
Li and Sankar (1992)	E–L	459	390

7.5 The Natural Orthogonal Complement

In simulation studies, we need to integrate the system of ordinary differential equations (ODE) describing the dynamics of a robotic mechanical system. This system is known as the *mathematical model* of the system at hand. Note that the Newton–Euler equations derived above for a serial manipulator do not constitute the mathematical model because we cannot use the recursive relations derived therein to set up the underlying ODE *directly*. What we need is a model relating the *state* of the system with its external generalized forces, of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{7.45}$$

where $\mathbf{x}$ is the *state vector*, $\mathbf{u}$ is the *input* or *control vector*, $\mathbf{x}_0$ is the state vector at a certain time t_0, and $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is a nonlinear function of $\mathbf{x}$ and $\mathbf{u}$, derived from the dynamics of the system. The state of a dynamical system is defined, in turn, as *the set of variables that separate the past from the future of the system* (Bryson and Ho 1975). Thus, if we take t_0 as the present time, we can predict from Eqs. (7.45) the future states of the system upon integration of the initial-value problem at hand, even if we do not know the complete past history of the system in full detail. Now, if we regard the vector $\boldsymbol{\theta}$ of independent joint variables and its time-rate of change, $\dot{\boldsymbol{\theta}}$, as the vectors of generalized coordinates and generalized speeds, then an obvious definition of $\mathbf{x}$ is

$$\mathbf{x} \equiv \left[\boldsymbol{\theta}^T \ \dot{\boldsymbol{\theta}}^T\right]^T \tag{7.46}$$

The n generalized coordinates, then, define the configuration of the system, while their time-derivatives determine its generalized momentum, a concept defined in Eq. (7.19d). Hence, knowing $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ at t_0, we can predict the future values of these variables, at $t > t_0$, with the aid of Eqs. (7.45).

Below we will derive the mathematical model, Eq. (7.45), explicitly, as pertaining to serial manipulators, in terms of the kinematic structure of the system and its inertial properties, i.e., the mass, center of mass coordinates, and inertia matrix of each of its bodies. To this end, we first write the underlying system of uncoupled Newton–Euler equations for each link. We have $n + 1$ links numbered from 0 to n, which are coupled by n kinematic pairs. Moreover, the base link 0 need not be an inertial frame; if it is noninertial, then the force and moment exerted by the environment upon it must be known. For ease of presentation, we will assume in this section that the base frame is inertial, the modifications needed to handle a noninertial base frame to be introduced in Sect. 7.5.2.

We now recall the Newton–Euler equations of the ith body in six-dimensional form, Eqs. (7.5c), which we reproduce below for quick reference:

$$\mathbf{M}_i \dot{\mathbf{t}}_i = -\mathbf{W}_i \mathbf{M}_i \mathbf{t}_i + \mathbf{w}_i^W + \mathbf{w}_i^C, \quad i = 1, \ldots, n \tag{7.47}$$

Furthermore, the definitions of Eqs. (7.13b and c) are recalled. Apparently, $\mathbf{M}$ and $\mathbf{W}$ are now $6n \times 6n$ matrices, while $\mathbf{t}$, $\mathbf{w}^C$, $\mathbf{w}^A$, and $\mathbf{w}^D$ are all $6n$-dimensional vectors. Then, the foregoing $6n$ scalar equations for the n moving links take the simple form

$$\mathbf{M}\dot{\mathbf{t}} = -\mathbf{W}\mathbf{M}\mathbf{t} + \mathbf{w}^A + \mathbf{w}^G + \mathbf{w}^D + \mathbf{w}^C \tag{7.48}$$

in which $\mathbf{w}^W$ has been decomposed into its active, gravitational, and dissipative parts $\mathbf{w}^A$, $\mathbf{w}^G$, and $\mathbf{w}^D$, respectively. Now, since gravity acts at the center of mass of a body, the gravity wrench $\mathbf{w}_i^G$ acting on the ith link takes the form

$$\mathbf{w}_i^G = \begin{bmatrix} \mathbf{0} \\ m_i \mathbf{g} \end{bmatrix} \tag{7.49}$$

The mathematical model displayed in Eq. (7.48) represents the $6n$ *uncoupled* Newton–Euler equations of the overall manipulator. The following step of this derivation consists in reducing the uncoupled system of equations to a set of n coupled equations of motion. To this end, the coupling between every two consecutive links is represented as a *linear homogeneous system* of algebraic equations on the link twists. Moreover, we note that all kinematic pairs allow a relative one-degree-of-freedom motion between the coupled bodies. We can then express the kinematic constraints of the system in *linear homogeneous form* in the $6n$-dimensional vector of manipulator twist, namely,

$$\mathbf{K}\mathbf{t} = \mathbf{0} \tag{7.50}$$

with $\mathbf{K}$ being a $6n \times 6n$ matrix, to be derived in Sect. 7.5.1. What is important to note at the moment is that the *kinematic constraint equations*, or *constraint equations*, for brevity, Eqs. (7.50), consist of a system of $6n$ scalar equations, i.e., six scalar

equations for each joint, for the manipulator at hand has n joints. Moreover, when the system is in motion, $\mathbf{t}$ is different from zero, and hence, matrix $\mathbf{K}$ is bound to be singular. In fact, the dimension of the null space of $\mathbf{K}$, termed its *nullity*, is exactly equal to n, the degree of freedom of the manipulator. That is, not every possible $6n$-dimensional twist is *feasible*; to be so, a twist must lie in the null space of matrix $\mathbf{K}$. Furthermore, since the nonworking constraint wrench $\mathbf{w}^C$ produces no work on the manipulator, its sole function being to keep the links together, the power developed by this wrench on $\mathbf{t}$, for any feasible motion of the manipulator, is zero, i.e.,

$$\mathbf{t}^T \mathbf{w}^C = 0 \tag{7.51}$$

On the other hand, if the two sides of Eq. (7.50) are transposed and then multiplied by a $6n$-dimensional vector $\boldsymbol{\lambda}$, one has

$$\mathbf{t}^T \mathbf{K}^T \boldsymbol{\lambda} = 0 \tag{7.52}$$

Upon comparing Eqs. (7.51) and (7.52), it is apparent that $\mathbf{w}^C$ is of the form

$$\mathbf{w}^C = \mathbf{K}^T \boldsymbol{\lambda} \tag{7.53}$$

More formally, the scalar product of $\mathbf{w}^C$ and $\mathbf{t}$, as stated by Eq. (7.51), vanishes, and hence, $\mathbf{t}$ lies in the null space of $\mathbf{K}$, as stated by Eq. (7.50). This means that $\mathbf{w}^C$ lies in the range of $\mathbf{K}^T$, as stated in Eq. (7.53). The following step will be to represent $\mathbf{t}$ as a linear transformation of the independent generalized speeds, i.e., as

$$\mathbf{t} = \mathbf{T}\dot{\boldsymbol{\theta}} \tag{7.54}$$

with $\mathbf{T}$ defined as a $6n \times n$ matrix that can be fairly termed the *twist-shaping matrix*. Moreover, the above mapping will be referred to as the *twist–shape relations*. The derivation of expressions for matrices $\mathbf{K}$ and $\mathbf{T}$ will be described in detail in Sect. 7.5.1 below. Now, upon substitution of Eq. (7.54) into Eq. (7.50), we obtain

$$\mathbf{KT}\dot{\boldsymbol{\theta}} = \mathbf{0} \tag{7.55a}$$

Furthermore, since the degree of freedom of the manipulator is n, the n generalized speeds $\{\dot{\theta}_i\}_1^n$ can be assigned arbitrarily. However, while doing this, Eq. (7.55a) has to hold. Thus, the only possibility for this to happen is that the product $\mathbf{KT}$ vanish, i.e.,

$$\mathbf{KT} = \mathbf{O} \tag{7.55b}$$

where $\mathbf{O}$ denotes the $6n \times n$ zero matrix. The above equation states that $\mathbf{T}$ is an *orthogonal complement* of $\mathbf{K}$. Because of the particular form of choosing this complement—see Eq. (7.54)—we refer to $\mathbf{T}$ as the *natural orthogonal complement* of $\mathbf{K}$ (Angeles and Lee 1988).

In the final step of this method, $\dot{\mathbf{t}}$ of Eq. (7.48) is obtained from Eq. (7.54), namely,

$$\dot{\mathbf{t}} = \mathbf{T}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{T}}\dot{\boldsymbol{\theta}} \tag{7.56}$$

Further, both sides of the uncoupled equations (7.48) are multiplied from the left by $\mathbf{T}^T$, the effect being that the term $\mathbf{w}^C$ is eliminated. Indeed, if $\mathbf{w}^C$ is expressed as appearing in Eq. (7.53), we have

$$\mathbf{T}^T\mathbf{w}^C = \mathbf{T}^T\mathbf{K}^T\boldsymbol{\lambda} \equiv (\mathbf{KT})^T\boldsymbol{\lambda}$$

However, by virtue of Eq. (7.55b), the matrix coefficient of $\boldsymbol{\lambda}$ in the foregoing expression vanishes, thereby obtaining n independent equations free of nonworking constraint wrenches. These are nothing but the governing equations of the manipulator, namely,

$$\mathbf{I}\ddot{\boldsymbol{\theta}} = -\mathbf{T}^T(\mathbf{M}\dot{\mathbf{T}} + \mathbf{WMT})\dot{\boldsymbol{\theta}} + \mathbf{T}^T(\mathbf{w}^A + \mathbf{w}^D + \mathbf{w}^G) \tag{7.57}$$

where $\mathbf{I}$, the positive definite $n \times n$ *generalized inertia matrix* of the manipulator, is defined as

$$\mathbf{I} \equiv \mathbf{T}^T\mathbf{MT} \tag{7.58}$$

which is identical to the inertia matrix derived using the Euler–Lagrange equations, with $\boldsymbol{\theta}$ as the vector of generalized coordinates. Now, we let $\boldsymbol{\tau}$ and $\boldsymbol{\delta}$ denote the n-dimensional vectors of active and dissipative generalized force. Moreover, we let $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}$ be the n-dimensional vector of *quadratic* terms of inertia force. These items are defined as

$$\begin{aligned} \boldsymbol{\tau} \equiv \mathbf{T}^T\mathbf{w}^A, \quad \boldsymbol{\delta} \equiv \mathbf{T}^T\mathbf{w}^D, \quad \boldsymbol{\gamma} = \mathbf{T}^T\mathbf{w}^G, \\ \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \equiv \mathbf{T}^T\mathbf{M}\dot{\mathbf{T}} + \mathbf{T}^T\mathbf{WMT} \end{aligned} \tag{7.59}$$

Clearly, the sum $\boldsymbol{\tau} + \boldsymbol{\delta}$ produces $\boldsymbol{\phi}_{\overline{p}}$, the generalized force defined in Eq. (7.11). Thus, the Euler–Lagrange equations of the system take the form

$$\mathbf{I}\ddot{\boldsymbol{\theta}} = -\mathbf{C}\dot{\boldsymbol{\theta}} + \boldsymbol{\tau} + \boldsymbol{\delta} + \boldsymbol{\gamma} \tag{7.60}$$

If, moreover, a static wrench $\mathbf{w}^W$ acts onto the end-effector, with the force applied at the operation point, then its effect onto the above model is taken into account as indicated in Eq. (5.50). Thus, a term $\mathbf{J}^T\mathbf{w}^W$ is added to the right-hand side of the above model:

$$\mathbf{I}\ddot{\boldsymbol{\theta}} = -\mathbf{C}\dot{\boldsymbol{\theta}} + \boldsymbol{\tau} + \boldsymbol{\delta} + \boldsymbol{\gamma} + \mathbf{J}^T\mathbf{w}^W \tag{7.61}$$

As a matter of fact, $\boldsymbol{\delta}$ is defined in Eq. (7.59) only for conceptual reasons. In practice, this term is most readily calculated once a dissipation function in terms of the generalized coordinates and generalized speeds is available, as described in Sect. 7.8. Thus, $\boldsymbol{\delta}$ is computed as

$$\boldsymbol{\delta} = -\frac{\partial \Delta}{\partial \dot{\boldsymbol{\theta}}} \tag{7.62}$$

It is pointed out that the first term of the right-hand side of Eq. (7.60) is *quadratic* in $\dot{\boldsymbol{\theta}}$ because matrix $\mathbf{C}$, defined in Eq. (7.59), is linear in $\dot{\boldsymbol{\theta}}$. In fact, the first term of that expression is linear in a factor $\dot{\mathbf{T}}$ that is, in turn, linear in $\dot{\boldsymbol{\theta}}$. Moreover, the second term of the same expression is linear in $\mathbf{W}$, which is linear in $\dot{\boldsymbol{\theta}}$ as well. However, $\mathbf{C}$ is *nonlinear* in $\boldsymbol{\theta}$. Because of the quadratic nature of that term, it is popularly known as the vector of *Coriolis and centrifugal forces*, whereas the left-hand side of that equation is given the name of vector of *inertia forces*. Properly speaking, both the left-hand side and the first term of the right-hand side of Eq. (7.60) arise from inertia forces.

Example 7.5.1 (A Minimum-Time Trajectory). A pick-and-place operation is to be performed with a n-axis manipulator in the shortest possible time. Moreover, the maneuver is defined so that the n-dimensional vector of joint variables is given by a common shape function $s(x)$, with $0 \leq x \leq 1$ and $0 \leq s \leq 1$, which is prescribed. Thus, for a fixed n-dimensional vector $\boldsymbol{\theta}_0$, the time-history of the joint-variable vector, $\boldsymbol{\theta}(t)$, is given by

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}_0 + s\left(\frac{t}{T}\right)\Delta\boldsymbol{\theta}, \quad 0 \leq t \leq T$$

with T defined as the time taken by the maneuver, while $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}$ are the values of the joint-variable vector at the pick- and the place-postures of the manipulator, respectively. These vectors are computed upon solving the inverse-displacement problem, as explained in Chap. 4. Furthermore, the load-carrying capacity of the manipulator is specified in terms of the maximum torques delivered by the motors, namely,

$$|\tau_i| \leq \overline{\tau}_i, \quad \text{for} \quad i = 1, \ldots, n$$

where the constant values $\overline{\tau}_i$ are to be found from data supplied by the manufacturer. In order to keep the analysis simple, we neglect power loses in this example. Find the minimum time in which the maneuver can take place.

Solution: Let us first calculate the vector of joint-rates and its time-derivative:

$$\dot{\boldsymbol{\theta}}(t) = \frac{1}{T}s'(x)\Delta\boldsymbol{\theta}, \quad \ddot{\boldsymbol{\theta}}(t) = \frac{1}{T^2}s''(x)\Delta\boldsymbol{\theta}, \quad x \equiv \frac{t}{T}$$

Now we substitute the above values into the mathematical model of Eq. (7.60), with $\boldsymbol{\delta}(t) = \mathbf{0}$ and $\boldsymbol{\gamma}(t) = \mathbf{0}$, thereby obtaining

$$\boldsymbol{\tau} = \mathbf{I}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} = \frac{1}{T^2}[s''(x)\mathbf{I}(x)\Delta\boldsymbol{\theta} + s'^2(x)\mathbf{C}(x, \Delta\boldsymbol{\theta})\Delta\boldsymbol{\theta}] \equiv \frac{1}{T^2}\mathbf{f}(x)$$

with $\mathbf{f}(x)$ defined, of course, as

$$\mathbf{f}(x) \equiv [\mathbf{I}(x)s''(x) + \mathbf{C}(x)s'^2(x)]\Delta\boldsymbol{\theta}$$

the $1/T^2$ factor in the term of Coriolis and centrifugal forces stemming from the quadratic nature of the $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}$ term. What we now have is the vector of motor torques, $\boldsymbol{\tau}$, expressed as a function of the scalar argument x. Now, let $f_i(x)$ be the ith component of vector $\mathbf{f}(x)$, and

$$F_i \equiv \max_x\{|f_i(x)|\}, \quad \text{for} \quad i = 1, \ldots, n$$

We would then like to have each value F_i produce the maximum available torque $\overline{\tau}_i$, namely,

$$\overline{\tau}_i = \frac{F_i}{T^2}, \quad i = 1, \ldots n$$

and hence, for each joint we have a value T_i of T given by

$$T_i^2 \equiv \frac{F_i}{\overline{\tau}_i}, \quad i = 1, \ldots n$$

Obviously, the minimum value sought, $T_{\min}$, is nothing but the maximum of the foregoing values, i.e.,

$$T_{\min} = \max_i\{T_i\}_1^n$$

thereby completing the solution.

7.5.1 *Derivation of Constraint Equations and Twist–Shape Relations*

In order to illustrate the general ideas behind the method of the natural orthogonal complement, we derive below the underlying kinematic constraint equations and the twist–shape relations. We first note, from Eq. (7.25a), that the relative angular velocity of the ith link with respect to the $(i-1)$st link, $\boldsymbol{\omega}_i - \boldsymbol{\omega}_{i-1}$, is $\dot{\theta}_i \mathbf{e}_i$. Thus, if

matrix $\mathbf{E}_i$ is defined as the cross-product matrix of vector $\mathbf{e}_i$, then, the angular velocities of two successive links obey a simple relation, namely,

$$\mathbf{E}_i(\boldsymbol{\omega}_i - \boldsymbol{\omega}_{i-1}) = \mathbf{0} \tag{7.63}$$

Furthermore, we rewrite now Eq. (7.33a) in the form

$$\dot{\mathbf{c}}_i - \dot{\mathbf{c}}_{i-1} + \mathbf{R}_i\boldsymbol{\omega}_i + \mathbf{D}_{i-1}\boldsymbol{\omega}_{i-1} = \mathbf{0} \tag{7.64}$$

where $\mathbf{D}_i$ and $\mathbf{R}_i$ are defined as the cross-product matrices of vectors $\boldsymbol{\delta}_i$, defined in Sect. 7.4.1 as $\mathbf{a}_i - \boldsymbol{\rho}_i$, and $\boldsymbol{\rho}_i$, respectively. In particular, when the first link is inertial, Eqs. (7.63 and b), as pertaining to the first link, reduce to

$$\mathbf{E}_1\boldsymbol{\omega}_1 = \mathbf{0} \tag{7.65a}$$

$$\dot{\mathbf{c}}_1 + \mathbf{R}_1\boldsymbol{\omega}_1 = \mathbf{0} \tag{7.65b}$$

Now, Eqs. (7.63) and (7.64), as well as their counterparts for $i = 1$, Eqs. (7.65a and b), are further expressed in terms of the link twists, thereby producing the constraints below:

$$\mathbf{K}_{11}\mathbf{t}_1 = \mathbf{0} \tag{7.66a}$$

$$\mathbf{K}_{i,i-1}\mathbf{t}_{i-1} + \mathbf{K}_{ii}\mathbf{t}_i = \mathbf{0}, \quad i = 1, \ldots, n \tag{7.66b}$$

with $\mathbf{K}_{11}$ and $\mathbf{K}_{ij}$, for $i = 2, \ldots, n$ and $j = i-1, i$, defined as

$$\mathbf{K}_{11} \equiv \begin{bmatrix} \mathbf{E}_1 & \mathbf{O} \\ \mathbf{R}_1 & \mathbf{1} \end{bmatrix} \tag{7.67a}$$

$$\mathbf{K}_{i,i-1} \equiv \begin{bmatrix} -\mathbf{E}_i & \mathbf{O} \\ \mathbf{D}_{i-1} & -\mathbf{1} \end{bmatrix} \tag{7.67b}$$

$$\mathbf{K}_{ii} \equiv \begin{bmatrix} \mathbf{E}_i & \mathbf{O} \\ \mathbf{R}_i & \mathbf{1} \end{bmatrix} \tag{7.67c}$$

where $\mathbf{1}$ and $\mathbf{O}$ denote the 3×3 identity and zero matrices, respectively. Furthermore, from Eqs. (7.66a and b) and (7.67a–c), it is apparent that matrix $\mathbf{K}$ appearing in Eq. (7.55b) takes on the form

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{O}_6 & \mathbf{O}_6 & \cdots & \mathbf{O}_6 & \mathbf{O}_6 \\ \mathbf{K}_{21} & \mathbf{K}_{22} & \mathbf{O}_6 & \cdots & \mathbf{O}_6 & \mathbf{O}_6 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{O}_6 & \mathbf{O}_6 & \mathbf{O}_6 & \cdots & \mathbf{K}_{n-1,n-1} & \mathbf{O}_6 \\ \mathbf{O}_6 & \mathbf{O}_6 & \mathbf{O}_6 & \cdots & \mathbf{K}_{n,n-1} & \mathbf{K}_{nn} \end{bmatrix} \tag{7.68}$$

with $\mathbf{O}_6$ denoting the 6×6 zero matrix.

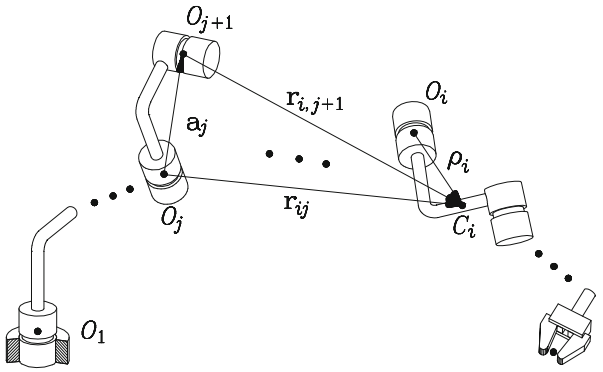

Fig. 7.6 Kinematic subchain comprising links $j,\ j+1\ldots,\ i$

Further, the link-twists are expressed as linear combinations of the joint-rate vector $\dot{\boldsymbol{\theta}}$. To this end, we define the $6\times n$ *partial Jacobian* $\mathbf{J}_i$ as the matrix mapping the joint-rate vector $\dot{\boldsymbol{\theta}}$ into the twist $\mathbf{t}_i$ of that link, i.e.,

$$\mathbf{J}_i\dot{\boldsymbol{\theta}} = \mathbf{t}_i \tag{7.69}$$

whose jth column, $\mathbf{t}_{ij}$, is given, for $i,\ j = 1,\ 2,\ \ldots,\ n$, by

$$\mathbf{t}_{ij} = \begin{cases} \begin{bmatrix} \mathbf{e}_j \\ \mathbf{e}_j \times \mathbf{r}_{ij} \end{bmatrix}, & \text{if } j \le i; \\ \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, & \text{otherwise.} \end{cases} \tag{7.70}$$

with $\mathbf{r}_{ij}$ illustrated in Fig. 7.6 and defined, for $i,\ j = 1,\ldots,n$, as

$$\mathbf{r}_{ij} \equiv \begin{cases} \mathbf{a}_j + \mathbf{a}_{j+1} + \cdots + \mathbf{a}_{i-1} + \boldsymbol{\rho}_i, & \text{if } j < i; \\ \boldsymbol{\rho}_i, & \text{if } j = i; \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{7.71}$$

It is noteworthy that, for a given i and a given $j \le i$, a submanipulator of $i-(j-1)$ axes is obtained. The $\{\mathbf{r}_{ij}\}_{j=1}^{i}$ vectors are the counterparts of the $\{\mathbf{r}_i\}_1^n$ vectors of Sect. 5.2

We can thus readily express the twist $\mathbf{t}_i$ of the ith link as a linear combination of the first i joint rates, namely,

$$\mathbf{t}_i = \dot{\theta}_1\mathbf{t}_{i1} + \dot{\theta}_2\mathbf{t}_{i2} + \cdots + \dot{\theta}_i\mathbf{t}_{ii}, \quad i = 1,\ldots,n \tag{7.72}$$

and hence, matrix $\mathbf{T}$ of Eq. (7.54) takes the form

$$\mathbf{T} \equiv \begin{bmatrix} \mathbf{t}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{t}_{21} & \mathbf{t}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{t}_{n1} & \mathbf{t}_{n2} & \cdots & \mathbf{t}_{nn} \end{bmatrix} \tag{7.73}$$

As a matter of verification, one can readily prove that the product of matrix $\mathbf{T}$, as given by Eq. (7.73), by matrix $\mathbf{K}$, as given by Eq. (7.68), vanishes, and hence, relation (7.55b) holds.

The kinematic constraint equations on the twists, for the case in which the ith joint is prismatic, are derived likewise. In this case, we use Eqs. (7.34a and e), with the latter rewritten more conveniently for our purposes, namely,

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \tag{7.74a}$$

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\delta}_{i-1} + \boldsymbol{\rho}_i + b_i \mathbf{e}_i) + \dot{b}_i \mathbf{e}_i \tag{7.74b}$$

We now introduce one further definition:

$$\mathbf{R}'_i \equiv \mathbf{D}'_{i-1} + \mathbf{R}_i \tag{7.75}$$

where $\mathbf{D}'_{i-1}$ is the cross-product matrix of vector $\boldsymbol{\delta}_{i-1}$, defined in Sect. 7.4.1 as $\mathbf{d}_{i-1} - \boldsymbol{\rho}_{i-1}$, while $\mathbf{R}_i$ is the cross-product matrix of $\boldsymbol{\rho}_i + b_i \mathbf{e}_i$. Hence, Eq. (7.74b) can be rewritten as

$$\dot{\mathbf{c}}_i - \dot{\mathbf{c}}_{i-1} + \mathbf{R}'_i \boldsymbol{\omega}_i - \dot{b}_i \mathbf{e}_i = \mathbf{0} \tag{7.76}$$

Upon multiplication of both sides of Eq. (7.76) by $\mathbf{E}_i$, the term in $\dot{b}_i$ cancels, and we obtain

$$\mathbf{E}_i (\dot{\mathbf{c}}_i - \dot{\mathbf{c}}_{i-1} + \mathbf{R}'_i \boldsymbol{\omega}_i) = \mathbf{0} \tag{7.77}$$

Hence, Eqs. (7.74a) and (7.77) can now be regrouped in a single six-dimensional linear homogeneous equation in the twists, namely,

$$\mathbf{K}'_{i,i-1} \mathbf{t}_{i-1} + \mathbf{K}'_{ii} \mathbf{t}_i = \mathbf{0} \tag{7.78}$$

the associated matrices being defined below:

$$\mathbf{K}'_{i,i-1} \equiv \begin{bmatrix} -\mathbf{1} & \mathbf{O} \\ \mathbf{O} & -\mathbf{E}_i \end{bmatrix} \tag{7.79a}$$

$$\mathbf{K}'_{ii} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{E}_i \mathbf{R}'_i & \mathbf{E}_i \end{bmatrix} \tag{7.79b}$$

with $\mathbf{1}$ and $\mathbf{O}$ defined already as the 3×3 identity and zero matrices, respectively. If the first joint is prismatic, then the corresponding constraint equation takes on the form

$$\mathbf{K}'_{11}\mathbf{t}_1 = \mathbf{0} \tag{7.80}$$

with $\mathbf{K}'_{11}$ defined as

$$\mathbf{K}'_{11} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{E}_1 \end{bmatrix} \tag{7.81}$$

Furthermore, if the kth pair is prismatic and $1 \leq k \leq i$, then the twist $\mathbf{t}_i$ of the ith link changes to

$$\mathbf{t}_i = \dot{\theta}_1\mathbf{t}_{i1} + \cdots + \dot{b}_k\mathbf{t}'_{ik} + \cdots + \dot{\theta}_i\mathbf{t}_{ii}, \quad i = 1, \ldots, n \tag{7.82}$$

where $\mathbf{t}'_{ik}$ is defined as

$$\mathbf{t}'_{ik} \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_k \end{bmatrix} \tag{7.83}$$

In order to set up Eq. (7.60), then all we now need is $\dot{\mathbf{T}}$, which is computed below. Two cases will be distinguished again, namely, whether the joint at hand is a revolute or a prismatic pair. In the first case, from Eq. (7.70) one readily derives, for $i, j = 1, 2, \ldots, n$,

$$\dot{\mathbf{t}}_{ij} = \begin{cases} \begin{bmatrix} \boldsymbol{\omega}_j \times \mathbf{e}_j \\ (\boldsymbol{\omega}_j \times \mathbf{e}_j) \times \mathbf{r}_{ij} + \mathbf{e}_j \times \dot{\mathbf{r}}_{ij} \end{bmatrix}, & \text{if } j \leq i; \\ \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, & \text{otherwise} \end{cases} \tag{7.84}$$

where, from Eq. (7.71),

$$\dot{\mathbf{r}}_{ij} = \boldsymbol{\omega}_j \times \mathbf{a}_j + \cdots + \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{\rho}_i \tag{7.85}$$

On the other hand, if the kth pair is prismatic and $1 \leq k \leq i$, then from Eq. (7.83), the time-rate of change of $\mathbf{t}'_{ik}$ becomes

$$\dot{\mathbf{t}}'_{ik} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\omega}_k \times \mathbf{e}_k \end{bmatrix} \tag{7.86}$$

thereby completing the desired derivations.

Note that the natural orthogonal complement can also be used for the inverse dynamics calculations. In this case, if the manipulator is subjected to a gravity field, then the twist-rate of the first link will have to be modified by adding a nonhomogeneous term to it, thereby accounting for the gravity-acceleration terms. This issue is discussed in Sect. 7.7.

7.5.2 Noninertial Base Link

Noninertial bases occur in space applications, e.g., in the case of a manipulator mounted on a space platform or on the space shuttle. A noninertial base can be readily handled with the use of the natural orthogonal complement, as discussed in this subsection. Since the base is free of attachments to an inertial frame, we have to add its six degrees of freedom (dof) to the n dof of the rest of the manipulator. Correspondingly, $\mathbf{t}$, $\mathbf{w}^C$, $\mathbf{w}^A$, and $\mathbf{w}^D$ now become $6(n+1)$-dimensional vectors. In particular, $\mathbf{t}$ takes the form

$$\mathbf{t} = \left[\mathbf{t}_0^T \ \mathbf{t}_1^T \ \dots \ \mathbf{t}_n^T\right]^T \tag{7.87}$$

with $\mathbf{t}_0$ defined as the twist of the base. Furthermore, the vector of independent generalized speeds, $\dot{\boldsymbol{\theta}}$, is now of dimension $n+6$, its first six components being those of $\mathbf{t}_0$, the other n remaining as in the previous case. Thus, $\dot{\boldsymbol{\theta}}$ has the components shown below:

$$\dot{\boldsymbol{\theta}} \equiv \left[\mathbf{t}_0^T \ \dot{\theta}_1 \ \dots \ \dot{\theta}_n\right]^T \tag{7.88}$$

Correspondingly, $\mathbf{T}$ becomes a $6(n+1) \times (n+6)$ matrix, namely,

$$\mathbf{T} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O}' & \mathbf{T}' \end{bmatrix} \tag{7.89}$$

where $\mathbf{1}$ is the 6×6 identity matrix, $\mathbf{O}$ denotes the $6 \times n$ zero matrix, $\mathbf{O}'$ represents the $6n \times 6$ zero matrix, and $\mathbf{T}'$ is the $6n \times n$ matrix defined in Eq. (7.73) as $\mathbf{T}$. Otherwise, the model remains as in the case of an inertial base.

A word of caution is in order here. Because of the presence of the twist vector $\mathbf{t}_0$ in the definition of the vector of generalized speeds above, the latter cannot, properly speaking, be regarded as a time-derivative. Indeed, as studied in Chap. 3, the angular velocity appearing in the twist vector is not a time-derivative. Hence, the vector of independent generalized speeds defined in Eq. (7.88) is represented instead by $\mathbf{v}$, which does not imply a time-derivative, namely,

$$\mathbf{v} = \left[\mathbf{t}_0^T \ \dot{\theta}_1 \ \cdots \ \dot{\theta}_n\right]^T \tag{7.90}$$

7.6 Manipulator Forward Dynamics

Forward dynamics is needed either for purposes of simulation or for the model-based control of manipulators (Craig 1989), and hence, a fast calculation of the joint-variable time-histories $\boldsymbol{\theta}(t)$ is needed. These time-histories are calculated from the model displayed in Eq. (7.61), reproduced below for quick reference, in terms of vector $\boldsymbol{\theta}(t)$, i.e.,

$$\mathbf{I}\ddot{\boldsymbol{\theta}} = -\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{\tau}(t) + \boldsymbol{\delta}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \boldsymbol{\gamma}(\boldsymbol{\theta}) + \mathbf{J}^T \mathbf{w}^W \tag{7.91}$$

Clearly, what is at stake here is the calculation of $\ddot{\boldsymbol{\theta}}$ from the foregoing model. Indeed, the right-hand side of Eq. (7.91) can be calculated with the aid of the Newton–Euler recursive algorithm, as we will describe below, and needs no further discussion for the time being. Now, the calculation of $\ddot{\boldsymbol{\theta}}$ from Eq. (7.91) is similar to the calculation of $\dot{\boldsymbol{\theta}}$ from the relation between the joint-rates and the twist, derived in Sect. 5.2. From the discussion in that section, such calculations take a number of floating-point operations, or *flops*, that is proportional to n^3, and is thus said to have a complexity of $O(n^3)$—read "order n^3." In real-time calculations, we would like to have a computational scheme of $O(n)$. In attempting to derive such schemes, Walker and Orin (1982) proposed a procedure that they called the *composite rigid-body method*, whereby the number of flops is minimized by cleverly calculating $\mathbf{I}(\boldsymbol{\theta})$ and the right-hand side of Eq. (7.91) by means of the recursive Newton–Euler algorithm. In their effort, they produced an $O(n^2)$ algorithm to calculate $\ddot{\boldsymbol{\theta}}$. Thereafter, Featherstone (1983) proposed an $O(n)$ algorithm that is based, however, on the assumption that Coriolis and centrifugal forces are negligible. The same author reported an improvement to the aforementioned algorithm, namely, the *articulated-body method*, that takes into account Coriolis and centrifugal forces (Featherstone 1987.) The outcome, for a n-revolute manipulator, is an algorithm requiring $300n - 267$ multiplications and $279n - 259$ additions. For $n = 6$, these figures yield 1,533 multiplications and 1,415 additions. Li (1989) reported an $O(n^2)$ algorithm leading to 783 multiplications and 670 additions.

In this subsection, we illustrate the application of the method of the natural orthogonal complement to the modeling of an n-axis serial manipulator for purposes of simulation. While this algorithm gives an $O(n^3)$ complexity, its derivation is straightforward and gives, for a six-axis manipulator, a computational cost similar to that of Featherstone's, namely, 1,596 multiplications and 1,263 additions. Moreover, a clever definition of coordinate frames leads to even lower figures, i.e., 1,353 multiplications and 1,165 additions, as reported by Angeles (1988). Further developments on robot dynamics using the natural orthogonal complement have been reported by Saha (1997, 1999, 2008), who proposed the *decoupled* natural orthogonal complement as a means to enable the real-time inversion of the mass matrix.

The manipulator at hand is assumed to be constituted by n moving links coupled by n kinematic pairs of the revolute or prismatic types. Again, for brevity, the

base link is assumed to be inertial, noninertial bases being readily incorporated as described in Sect. 7.5.2. For the sake of conciseness, we will henceforth consider only manipulators mounted on an inertial base. Moreover, we assume that the generalized coordinates $\boldsymbol{\theta}$ and the generalized speeds $\dot{\boldsymbol{\theta}}$ are known at an instant t_k, along with the driving torque $\boldsymbol{\tau}(t)$, for $t \geq t_k$, and of course, the DH and the inertial parameters of the manipulator are assumed to be known as well. Based on the foregoing information, then, $\ddot{\boldsymbol{\theta}}$ is evaluated at t_k and, with a suitable integration scheme, the values of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ are determined at instant t_{k+1}. Obviously, the governing equation (7.60) enables us to solve for $\ddot{\boldsymbol{\theta}}(t_k)$. This requires, of course, the *inversion* of the $n \times n$ matrix of generalized inertia $\mathbf{I}$. Since the said matrix is positive-definite, solving for $\ddot{\boldsymbol{\theta}}$ from Eq. (7.60) can be done economically using the *Cholesky-decomposition* algorithm (Dahlquist and Björck 1974). The sole remaining task is, then, the computation of $\mathbf{I}$, the quadratic inertia term $\mathbf{C}\dot{\boldsymbol{\theta}}$, and the dissipative torque $\boldsymbol{\delta}$. The last of these is dependent on the manipulator and the constitutive model adopted for the representation of viscous and Coulomb friction forces and will not be considered at this stage. Models for dissipative forces will be studied in Sect. 7.8. Thus, the discussion below will focus on the computation of $\mathbf{I}$ and $\mathbf{C}\dot{\boldsymbol{\theta}}$ appearing in the mathematical model of Eq. (7.91).

Next, the $6n \times 6n$ matrix $\mathbf{M}$ is factored as

$$\mathbf{M} = \mathbf{H}^T\mathbf{H} \tag{7.92}$$

which is possible because $\mathbf{M}$ is at least positive-semidefinite. In particular, for manipulators of the type at hand, $\mathbf{M}$ is positive-definite if no link-mass is neglected. Moreover, due to the diagonal-block structure of this matrix, its factoring is straightforward. In fact, $\mathbf{H}$ is given simply by

$$\mathbf{H} = \mathrm{diag}(\mathbf{H}_1, \ldots, \mathbf{H}_n) \tag{7.93}$$

each 6×6 block $\mathbf{H}_i$ of Eq. (7.93) being given, in turn, as

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{N}_i & \mathbf{O} \\ \mathbf{O} & n_i\mathbf{1} \end{bmatrix} \tag{7.94}$$

with $\mathbf{1}$ and $\mathbf{O}$ defined as the 3×3 identity and zero matrices, respectively. We thus have

$$\mathbf{M}_i = \mathbf{H}_i^T\mathbf{H}_i \tag{7.95}$$

Furthermore, $\mathbf{N}_i$ can be obtained from the Cholesky decomposition of $\mathbf{I}_i$, while n_i is the *positive* square root of m_i, i.e.,

$$\mathbf{I}_i = \mathbf{N}_i^T\mathbf{N}_i, \quad m_i = n_i^2 \tag{7.96}$$

Now, since each 6×6 $\mathbf{M}_i$ block is constant in body-fixed coordinates, the above factoring can be done off-line. From the foregoing definitions, then, the $n \times n$ matrix of generalized inertia $\mathbf{I}$ can now be expressed as

$$\mathbf{I} = \mathbf{P}^T \mathbf{P} \tag{7.97}$$

where $\mathbf{P}$ is defined, in turn, as the $6n \times n$ matrix given below:

$$\mathbf{P} \equiv \mathbf{H}\mathbf{T} \tag{7.98}$$

The computation of $\mathbf{P}$ is now discussed. If we recall the structure of $\mathbf{T}$ from Eq. (7.73) and that of $\mathbf{H}$ from Eq. (7.93), along with the definition of $\mathbf{P}$, Eq. (7.98), we readily obtain

$$\mathbf{P} = \begin{bmatrix} \mathbf{H}_1\mathbf{t}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{H}_2\mathbf{t}_{21} & \mathbf{H}_2\mathbf{t}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_n\mathbf{t}_{n1} & \mathbf{H}_n\mathbf{t}_{n2} & \cdots & \mathbf{H}_n\mathbf{t}_{nn} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{p}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{p}_{21} & \mathbf{p}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_{n1} & \mathbf{p}_{n2} & \cdots & \mathbf{p}_{nn} \end{bmatrix} \tag{7.99}$$

with $\mathbf{0}$ denoting the six-dimensional zero vector. Moreover, each of the above nontrivial six-dimensional arrays $\mathbf{p}_{ij}$ is given as

$$\mathbf{p}_{ij} \equiv \mathbf{H}_i\mathbf{t}_{ij} = \begin{cases} \begin{bmatrix} \mathbf{N}_i\mathbf{e}_j \\ n_i\mathbf{e}_j \times \mathbf{r}_{ij} \end{bmatrix} & \text{if the } j\text{th joint is } R \\ \begin{bmatrix} \mathbf{0} \\ n_i\mathbf{e}_j \end{bmatrix} & \text{if the } j\text{th joint is } P \end{cases} \tag{7.100}$$

Thus, the (i, j) entry of $\mathbf{I}$ is computed as the sum of the inner products of the (k, i) and the (k, j) blocks of $\mathbf{P}$, for $k = j, \ldots, n$, i.e.,

$$I_{ij} = I_{ji} = \sum_{k=j}^{n} \mathbf{p}_{ki}^T \mathbf{p}_{kj} \tag{7.101}$$

with both $\mathbf{p}_{ki}$ and $\mathbf{p}_{kj}$ expressed in $\mathcal{F}_{k+1}$-coordinates, i.e., in kth-link coordinates. Now, the Cholesky decomposition of $\mathbf{I}$ can be expressed as

$$\mathbf{I} = \mathbf{L}^T\mathbf{L} \tag{7.102}$$

where $\mathbf{L}$ is an $n \times n$ lower-triangular matrix with positive diagonal entries. Moreover, Eq. (7.91) is now rewritten as

$$\mathbf{L}^T\mathbf{L}\ddot{\boldsymbol{\theta}} = -(\mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{J}^T\mathbf{w}^W - \boldsymbol{\gamma}) + \boldsymbol{\delta} + \boldsymbol{\tau} \tag{7.103}$$

If we now recall Eq. (7.91), it is apparent that the term inside the parentheses in the right-hand side of the above equation is nothing but the torque required to produce the motion prescribed by the current values of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$, in the absence of dissipative wrenches and with zero joint accelerations, when the manipulator is acted upon by a static wrench $\mathbf{w}^W$. That is, if we call $\overline{\boldsymbol{\tau}}$ the torque $\boldsymbol{\tau}$ of Eq. (7.91) under the foregoing conditions, then

$$\mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{J}^T\mathbf{w}^W - \boldsymbol{\gamma} = \boldsymbol{\tau}|_{\mathbf{w}^D=\mathbf{0},\ddot{\boldsymbol{\theta}}=\mathbf{0}} \equiv \overline{\boldsymbol{\tau}} \tag{7.104}$$

which is most efficiently computed from inverse dynamics, using the recursive Newton–Euler algorithm, as described in Sect. 7.4. Now Eq. (7.102) is solved for $\ddot{\boldsymbol{\theta}}$ in two steps, namely,

$$\mathbf{L}^T\mathbf{x} = -\overline{\boldsymbol{\tau}} + \boldsymbol{\tau} + \boldsymbol{\delta} \tag{7.105a}$$

$$\mathbf{L}\ddot{\boldsymbol{\theta}} = \mathbf{x} \tag{7.105b}$$

In the above equations, then, $\mathbf{x}$ is first computed from Eq. (7.105a) by backward substitution. With $\mathbf{x}$ known, $\ddot{\boldsymbol{\theta}}$ is computed from Eq. (7.105b) by forward substitution, thereby completing the computation of $\ddot{\boldsymbol{\theta}}$. The complexity of the foregoing algorithm is discussed in Sect. 7.6.2.

Alternatively, $\ddot{\boldsymbol{\theta}}$ can be calculated in two steps from two linear systems of equations, the first one underdetermined, the second overdetermined. Indeed, if we let the product $\mathbf{P}\ddot{\boldsymbol{\theta}}$ be denoted by $\mathbf{y}$, then the dynamics model of the manipulator, Eq. (7.60), along with the factoring of Eq. (7.97), leads to

$$\mathbf{P}^T\mathbf{y} = -\overline{\boldsymbol{\tau}} + \boldsymbol{\tau} + \boldsymbol{\delta} \tag{7.106a}$$

$$\mathbf{P}\ddot{\boldsymbol{\theta}} = \mathbf{y} \tag{7.106b}$$

Thus, in the above equations, $\mathbf{y}$ is calculated first as the *minimum-norm* solution of Eq. (7.106a); then, the desired value of $\ddot{\boldsymbol{\theta}}$ is calculated as the *least-square approximation* of Eq. (7.106b). These two solutions are computed most efficiently using an orthogonalization algorithm that reduces matrix $\mathbf{P}$ to upper-triangular form (Golub and Van Loan 1989). A straightforward calculation based on the explicit calculation of the generalized inverses involved is not recommended, because of the frequent numerical ill-conditioning incurred. Two orthogonalization procedures, one based on *Householder reflections*, the other on the Gram–Schmidt procedure, for the computation of both the least-square approximation of an overdetermined system of equations and the minimum-norm solution of its underdetermined counterpart are outlined in Appendix B.

The complexity of the foregoing calculations is discussed in Sect. 7.6.2, based on the Cholesky decomposition of the generalized inertia matrix, details on the alternative approach being available elsewhere (Angeles and Ma 1988).

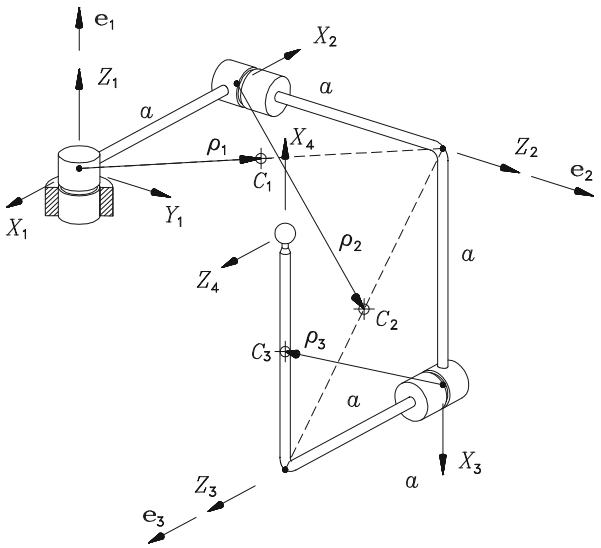

Fig. 7.7 Mass-center locations of the robot of Fig. 4.23

Example 7.6.1 (Dynamics of a Spatial Three-Revolute Robot). The robot of Fig. 4.17 is reproduced in Fig. 7.7, in a form that is kinematically equivalent to the sketch of that figure, but more suitable for the purposes of this example. For this robot, (a) find its inertia matrix at the configuration depicted in that figure; (b) find the time-rate of change of the inertia matrix under a maneuver whereby $\dot{\theta}_1 = \dot{\theta}_2 = \dot{\theta}_3 = p\ \mathrm{s}^{-1}$ and $\ddot{\theta}_1 = \ddot{\theta}_2 = \ddot{\theta}_3 = 0$; and (*iii*) under the same maneuver, find the centrifugal and Coriolis terms of its governing equation. Furthermore, assume that all links are identical and *dynamically isotropic*. What we mean by "dynamically isotropic" is that the moment of inertia of all three links about their centers of mass are proportional to the 3×3 identity matrix, the proportionality factor being I. Moreover, all three links are designed so that the center of mass of each is located as shown in Fig. 7.7.

Solution:

(a) Henceforth, we represent all vectors and matrices with respect to the $\mathcal{F}_1$-frame of Fig. 7.7, while denoting by **i**, **j**, and **k** the unit vectors parallel to the X_1, Y_1, and Z_1 axes, respectively. Under these conditions, we have, for the unit vectors parallel to the revolute axes,

$$\mathbf{e}_1 = \mathbf{k}, \quad \mathbf{e}_2 = \mathbf{j}, \quad \mathbf{e}_3 = \mathbf{i}$$

while vector $\mathbf{a}_i$ is directed from the origin of $\mathcal{F}_i$ to that of $\mathcal{F}_{i+1}$, for $i = 1, 2, 3$. Hence,

$$\mathbf{a}_1 = -a\mathbf{i}, \quad \mathbf{a}_2 = a(\mathbf{j} - \mathbf{k}), \quad \mathbf{a}_3 = a(\mathbf{i} + \mathbf{k})$$

Likewise, the position vectors of the centers of mass, $\boldsymbol{\rho}_i$, for $i = 1$, 2, and 3, with respect to the origins of their respective frames, are given by

$$\boldsymbol{\rho}_1 = \frac{1}{2}a(-\mathbf{i} + \mathbf{j})$$
$$\boldsymbol{\rho}_2 = \frac{1}{2}a(\mathbf{i} + 2\mathbf{j} - \mathbf{k})$$
$$\boldsymbol{\rho}_3 = \frac{1}{2}a(2\mathbf{i} + \mathbf{k})$$

We can now calculate the various six-dimensional arrays $\mathbf{t}_{ij}$, for $i = 1$, 2, 3, and $j = 1$ till i, i.e.,

$$\mathbf{t}_{11} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_1 \times \boldsymbol{\rho}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{k} \\ -(a/2)(\mathbf{i} + \mathbf{j}) \end{bmatrix}$$
$$\mathbf{t}_{21} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_1 \times (\mathbf{a}_1 + \boldsymbol{\rho}_2) \end{bmatrix} = \begin{bmatrix} \mathbf{k} \\ -(a/2)(2\mathbf{i} + \mathbf{j}) \end{bmatrix}$$
$$\mathbf{t}_{22} = \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_2 \times \boldsymbol{\rho}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{j} \\ -(a/2)(\mathbf{i} + \mathbf{k}) \end{bmatrix}$$
$$\mathbf{t}_{31} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_1 \times (\mathbf{a}_1 + \mathbf{a}_2 + \boldsymbol{\rho}_3) \end{bmatrix} = \begin{bmatrix} \mathbf{k} \\ -a\mathbf{i} \end{bmatrix}$$
$$\mathbf{t}_{32} = \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_2 \times (\mathbf{a}_2 + \boldsymbol{\rho}_3) \end{bmatrix} = \begin{bmatrix} \mathbf{j} \\ -(a/2)(\mathbf{i} + 2\mathbf{k}) \end{bmatrix}$$
$$\mathbf{t}_{33} = \begin{bmatrix} \mathbf{e}_3 \\ \mathbf{e}_3 \times \boldsymbol{\rho}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{i} \\ -(a/2)\mathbf{j} \end{bmatrix}$$

and so, the 18×3 matrix $\mathbf{T}$ is given by

$$\mathbf{T} = \begin{bmatrix} \mathbf{k} & \mathbf{0} & \mathbf{0} \\ -(a/2)(\mathbf{i} + \mathbf{j}) & \mathbf{0} & \mathbf{0} \\ \mathbf{k} & \mathbf{j} & \mathbf{0} \\ -(a/2)(2\mathbf{i} + \mathbf{j}) & -(a/2)(\mathbf{i} + \mathbf{k}) & \mathbf{0} \\ \mathbf{k} & \mathbf{j} & \mathbf{i} \\ -a\mathbf{i} & -(a/2)(\mathbf{i} + 2\mathbf{k}) & -(a/2)\mathbf{j} \end{bmatrix}$$

Moreover, the 6×6 inertia dyad of the ith link takes the form

$$\mathbf{M}_i = \begin{bmatrix} I\mathbf{1} & \mathbf{O} \\ \mathbf{O} & m\mathbf{1} \end{bmatrix}, \quad i = 1, 2, 3$$

with **1** and **O** denoting the 3×3 identity and zero matrices, respectively. Thus, the 18×18 system mass matrix is given as

$$\mathbf{M} = \text{diag}(\mathbf{M}_1,\ \mathbf{M}_2,\ \mathbf{M}_3)$$

and the 3×3 generalized inertia matrix $\mathbf{I}$ of the manipulator is

$$\mathbf{I} = \mathbf{T}^T\mathbf{M}\mathbf{T}$$

whose entries are given by

$$\begin{aligned}
I_{11} &= \mathbf{t}_{11}^T\mathbf{M}_1\mathbf{t}_{11} + \mathbf{t}_{21}^T\mathbf{M}_2\mathbf{t}_{21} + \mathbf{t}_{31}^T\mathbf{M}_3\mathbf{t}_{31} \\
I_{12} &= \mathbf{t}_{21}^T\mathbf{M}_2\mathbf{t}_{22} + \mathbf{t}_{31}^T\mathbf{M}_3\mathbf{t}_{32} = I_{21} \\
I_{13} &= \mathbf{t}_{31}^T\mathbf{M}_3\mathbf{t}_{33} = I_{31} \\
I_{22} &= \mathbf{t}_{22}^T\mathbf{M}_2\mathbf{t}_{22} + \mathbf{t}_{32}^T\mathbf{M}_3\mathbf{t}_{32} \\
I_{23} &= \mathbf{t}_{32}^T\mathbf{M}_3\mathbf{t}_{33} = I_{32} \\
I_{33} &= \mathbf{t}_{33}^T\mathbf{M}_3\mathbf{t}_{33}
\end{aligned}$$

Upon expansion, the foregoing expressions yield

$$\mathbf{I} = \frac{1}{4}ma^2\begin{bmatrix} 11 & 4 & 0 \\ 4 & 7 & 0 \\ 0 & 0 & 1 \end{bmatrix} + I\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) Now, the time-rate of change of $\mathbf{I}$, $\dot{\mathbf{I}}$, is calculated as

$$\dot{\mathbf{I}} = \mathbf{T}^T\mathbf{M}\dot{\mathbf{T}} + \dot{\mathbf{T}}^T\mathbf{M}\mathbf{T} + \mathbf{T}^T(\mathbf{W}\mathbf{M} - \mathbf{M}\mathbf{W})\mathbf{T}$$

We proceed first to compute $\dot{\mathbf{T}}$. This time-derivative is nothing but the 18×3 matrix whose entries are the time-derivatives of the entries of $\mathbf{T}$, namely, $\dot{\mathbf{t}}_{ij}$, as given in Eq. (7.84), which is reproduced below for quick reference:

$$\dot{\mathbf{t}}_{ij} = \begin{bmatrix} \boldsymbol{\omega}_j \times \mathbf{e}_j \\ (\boldsymbol{\omega}_j \times \mathbf{e}_j) \times \mathbf{r}_{ij} + \mathbf{e}_j \times \dot{\mathbf{r}}_{ij} \end{bmatrix}$$

where $\dot{\mathbf{r}}_{ij}$ is given, in turn, by

$$\dot{\mathbf{r}}_{ij} = \boldsymbol{\omega}_j \times \mathbf{a}_j + \ldots + \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{\rho}_i$$

Hence, we will need vectors $\boldsymbol{\omega}_i$, for $i = 1, 2$, and 3. These are calculated below:

$$\boldsymbol{\omega}_1 = \dot{\theta}_1\mathbf{e}_1 = p\mathbf{k}$$

$$\boldsymbol{\omega}_2 = \dot{\theta}_1 \mathbf{e}_1 + \dot{\theta}_2 \mathbf{e}_2 = p(\mathbf{j} + \mathbf{k})$$
$$\boldsymbol{\omega}_3 = \dot{\theta}_1 \mathbf{e}_1 + \dot{\theta}_2 \mathbf{e}_2 + \dot{\theta}_3 \mathbf{e}_3 = p(\mathbf{i} + \mathbf{j} + \mathbf{k})$$

We have, therefore,

$$\dot{\mathbf{t}}_{11} = \begin{bmatrix} \dot{\mathbf{e}}_1 \\ \dot{\mathbf{e}}_1 s \times \boldsymbol{\rho}_1 + \mathbf{e}_1 \times \dot{\boldsymbol{\rho}}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_1 \times (\boldsymbol{\omega}_1 \times \boldsymbol{\rho}_1) \end{bmatrix} = p \begin{bmatrix} \mathbf{0} \\ (1/2)a(\mathbf{i} - \mathbf{j}) \end{bmatrix}$$

$$\dot{\mathbf{t}}_{21} = \begin{bmatrix} \dot{\mathbf{e}}_1 \\ \dot{\mathbf{e}}_1 \times (\mathbf{a}_1 + \boldsymbol{\rho}_2) + \mathbf{e}_1 \times (\dot{\mathbf{a}}_1 + \dot{\boldsymbol{\rho}}_2) \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_1 \times (\boldsymbol{\omega}_1 \times \mathbf{a}_1 + \boldsymbol{\omega}_2 \times \boldsymbol{\rho}_2) \end{bmatrix} = p \begin{bmatrix} \mathbf{0} \\ (1/2)a\mathbf{j} \end{bmatrix}$$

$$\dot{\mathbf{t}}_{22} = \begin{bmatrix} \dot{\mathbf{e}}_2 \\ \dot{\mathbf{e}}_2 \times \boldsymbol{\rho}_2 + \mathbf{e}_2 \times \dot{\boldsymbol{\rho}}_2 \end{bmatrix}$$
$$= \begin{bmatrix} p\mathbf{e}_1 \times \mathbf{e}_2 \\ (p\mathbf{e}_1 \times \mathbf{e}_2) \times \boldsymbol{\rho}_2 + \mathbf{e}_2 \times [p(\mathbf{e}_1 + \mathbf{e}_2) \times \boldsymbol{\rho}_2] \end{bmatrix}$$
$$= p \begin{bmatrix} -\mathbf{i} \\ -(1/2)a(\mathbf{i} + \mathbf{j} - \mathbf{k}) \end{bmatrix}$$

$$\dot{\mathbf{t}}_{31} = \begin{bmatrix} \dot{\mathbf{e}}_1 \\ \dot{\mathbf{e}}_1 \times (\mathbf{a}_1 + \mathbf{a}_2 + \boldsymbol{\rho}_3) + \mathbf{e}_1 \times (\dot{\mathbf{a}}_1 + \dot{\mathbf{a}}_2 + \dot{\boldsymbol{\rho}}_3) \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_1 \times (\boldsymbol{\omega}_1 \times \mathbf{a}_1 + \boldsymbol{\omega}_2 \times \mathbf{a}_2 + \boldsymbol{\omega}_3 \times \boldsymbol{\rho}_3) \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_1 \times [p\mathbf{e}_1 \times \mathbf{a}_1 + p(\mathbf{e}_1 + \mathbf{e}_2) \times \mathbf{a}_2 + p(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) \times \boldsymbol{\rho}_3] \end{bmatrix}$$
$$= p \begin{bmatrix} \mathbf{0} \\ -a\mathbf{j} \end{bmatrix}$$

$$\dot{\mathbf{t}}_{32} = \begin{bmatrix} \dot{\mathbf{e}}_2 \\ \dot{\mathbf{e}}_2 \times (\mathbf{a}_2 + \boldsymbol{\rho}_3) + \mathbf{e}_2 \times (\dot{\mathbf{a}}_2 + \dot{\boldsymbol{\rho}}_3) \end{bmatrix}$$
$$= \begin{bmatrix} p\mathbf{e}_1 \times \mathbf{e}_2 \\ (p\mathbf{e}_1 \times \mathbf{e}_2) \times (\mathbf{a}_2 + \boldsymbol{\rho}_3) + p\mathbf{e}_2 \times [(\mathbf{e}_1 + \mathbf{e}_2) \times (\mathbf{a}_2 + \boldsymbol{\rho}_3)] \end{bmatrix}$$
$$= p \begin{bmatrix} -\mathbf{i} \\ -(1/2)a(2\mathbf{i} + \mathbf{j} - \mathbf{k}) \end{bmatrix}$$

$$\dot{\mathbf{t}}_{33} = \begin{bmatrix} \dot{\mathbf{e}}_3 \\ \dot{\mathbf{e}}_3 \times \boldsymbol{\rho}_3 + \mathbf{e}_3 \times \dot{\boldsymbol{\rho}}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_2 \times \mathbf{e}_3 \\ (\boldsymbol{\omega}_2 \times \mathbf{e}_3) \times \boldsymbol{\rho}_3 + \mathbf{e}_3 \times (\boldsymbol{\omega}_3 \times \boldsymbol{\rho}_3) \end{bmatrix}$$

$$= \begin{bmatrix} p(\mathbf{e}_1 + \mathbf{e}_2) \times \mathbf{e}_3 \\ p[(\mathbf{e}_1 + \mathbf{e}_2) \times \mathbf{e}_3] \times \boldsymbol{\rho}_3 + p\mathbf{e}_3 \times [(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) \times \boldsymbol{\rho}_3] \end{bmatrix}$$

$$\dot{\mathbf{t}}_{33} = \begin{bmatrix} p(\mathbf{e}_2 - \mathbf{e}_1) \\ p(\mathbf{e}_2 - \mathbf{e}_1) \times \boldsymbol{\rho}_3 + p[(\mathbf{e}_3 \cdot \boldsymbol{\rho}_3)(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) - \boldsymbol{\rho}_3] \end{bmatrix}$$

$$= p \begin{bmatrix} \mathbf{j} - \mathbf{k} \\ (1/2)a(\mathbf{i} - \mathbf{k}) \end{bmatrix}$$

Now, let

$$\overline{\mathbf{P}} \equiv \mathbf{T}^T \mathbf{M} \dot{\mathbf{T}}$$

whose entries are displayed below:

$$\begin{aligned}
p_{11} &= \mathbf{t}_{11}^T \mathbf{M}_1 \dot{\mathbf{t}}_{11} + \mathbf{t}_{21}^T \mathbf{M}_2 \dot{\mathbf{t}}_{21} + \mathbf{t}_{31}^T \mathbf{M}_3 \dot{\mathbf{t}}_{31} \\
p_{12} &= \mathbf{t}_{21}^T \mathbf{M}_2 \dot{\mathbf{t}}_{22} + \mathbf{t}_{31}^T \mathbf{M}_3 \dot{\mathbf{t}}_{32} \\
p_{13} &= \mathbf{t}_{31}^T \mathbf{M}_3 \dot{\mathbf{t}}_{33} \\
p_{21} &= \mathbf{t}_{22}^T \mathbf{M}_2 \dot{\mathbf{t}}_{21} + \mathbf{t}_{32}^T \mathbf{M}_3 \dot{\mathbf{t}}_{31} \\
p_{22} &= \mathbf{t}_{22}^T \mathbf{M}_2 \dot{\mathbf{t}}_{22} + \mathbf{t}_{32}^T \mathbf{M}_3 \dot{\mathbf{t}}_{32} \\
p_{23} &= \mathbf{t}_{32}^T \mathbf{M}_3 \dot{\mathbf{t}}_{33} \\
p_{31} &= \mathbf{t}_{33}^T \mathbf{M}_3 \dot{\mathbf{t}}_{31} \\
p_{32} &= \mathbf{t}_{33}^T \mathbf{M}_3 \dot{\mathbf{t}}_{32} \\
p_{33} &= \mathbf{t}_{33}^T \mathbf{M}_3 \dot{\mathbf{t}}_{33}
\end{aligned}$$

Upon performing the foregoing operations, we end up with

$$\mathbf{T}^T \mathbf{M} \dot{\mathbf{T}} = p \begin{bmatrix} -(1/4)a^2 m & (7/4)a^2 m & -(1/2)a^2 m - I \\ -(1/2)a^2 m & 0 & (1/4)a^2 m + I \\ (1/2)a^2 m & (1/4)a^2 m - I & 0 \end{bmatrix} \equiv \overline{\mathbf{P}}$$

the second term of the above expression for $\dot{\mathbf{I}}$ simply being $\overline{\mathbf{P}}^T$. In order to compute the third term, we need the products **WM** and **MW**. However, it is apparent that the latter is the negative of the transpose of the former, and so, all we need is one of the two terms. Furthermore, note that since both matrices **M** and **W** are block-diagonal, their product is block-diagonal as well, namely,

$$\mathbf{WM} = \text{diag}(\mathbf{W}_1 \mathbf{M}_1,\ \mathbf{W}_2 \mathbf{M}_2,\ \mathbf{W}_3 \mathbf{M}_3)$$

where for $i = 1$, 2, and 3,

$$\mathbf{W}_i = \begin{bmatrix} \boldsymbol{\Omega}_i & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$$

with $\mathbf{O}$ denoting the 3×3 zero matrix, while $\boldsymbol{\Omega}_i$ is the cross-product matrix of vector $\boldsymbol{\omega}_i$. Moreover,

$$\mathbf{W}_i\mathbf{M}_i = \begin{bmatrix} I\boldsymbol{\Omega}_i & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$$

Therefore, $\mathbf{W}_i\mathbf{M}_i$ is skew-symmetric; as a consequence, $\mathbf{WM}$ is also skew-symmetric, and the difference $\mathbf{WM} - \mathbf{MW}$ vanishes. Hence, in this particular case, $\dot{\mathbf{I}}$ reduces to

$$\dot{\mathbf{I}} = \mathbf{P} + \mathbf{P}^T$$

That is,

$$\dot{\mathbf{I}} = p \begin{bmatrix} -(1/2)a^2m & (5/4)a^2m & -I \\ (5/4)a^2m & 0 & a^2m + I \\ -I & (1/2)a^2m & 0 \end{bmatrix}$$

(c) Now, the term of Coriolis and centrifugal forces can be computed in two ways, namely, (a) as $(\mathbf{T}^T\mathbf{M}\dot{\mathbf{T}} + \mathbf{T}^T\mathbf{W}\mathbf{M}\mathbf{T})\dot{\boldsymbol{\theta}}$, and (b) by using the Newton–Euler algorithm with $\ddot{\theta}_i = 0$, for $i = 1, 2$, and 3. We proceed in these two ways in order to verify the correctness of our results.

In proceeding with the first alternative, we already have the first term in the foregoing parentheses; the second term is now computed. First, we note that

$$\mathbf{WMT} = \begin{bmatrix} \mathbf{W}_1\mathbf{M}_1\mathbf{t}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_2\mathbf{M}_2\mathbf{t}_{21} & \mathbf{W}_2\mathbf{M}_2\mathbf{t}_{22} & \mathbf{0} \\ \mathbf{W}_3\mathbf{M}_3\mathbf{t}_{31} & \mathbf{W}_3\mathbf{M}_3\mathbf{t}_{32} & \mathbf{W}_3\mathbf{M}_3\mathbf{t}_{33} \end{bmatrix}$$

with $\mathbf{0}$ defined as the six-dimensional zero vector. The foregoing nontrivial six-dimensional arrays are computed below:

$$\mathbf{W}_1\mathbf{M}_1\mathbf{t}_{11} = \begin{bmatrix} I\boldsymbol{\Omega}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ -(a/2)(\mathbf{i}+\mathbf{j}) \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_1\mathbf{k} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_2\mathbf{M}_2\mathbf{t}_{21} = \begin{bmatrix} I\boldsymbol{\Omega}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ -(a/2)(2\mathbf{i}+\mathbf{j}) \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_2\mathbf{k} \\ \mathbf{0} \end{bmatrix} = pI \begin{bmatrix} \mathbf{i} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_2\mathbf{M}_2\mathbf{t}_{22} = \begin{bmatrix} I\boldsymbol{\Omega}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{j} \\ -(a/2)(\mathbf{i}+\mathbf{k}) \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_2\mathbf{j} \\ \mathbf{0} \end{bmatrix}$$

$$= \begin{bmatrix} pI(\mathbf{j}+\mathbf{k})\times\mathbf{j} \\ \mathbf{0} \end{bmatrix} = pI \begin{bmatrix} -\mathbf{i} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_3\mathbf{M}_3\mathbf{t}_{31} = \begin{bmatrix} I\boldsymbol{\Omega}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ -a\mathbf{i} \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_3\mathbf{k} \\ \mathbf{0} \end{bmatrix}$$

$$= \begin{bmatrix} pI(\mathbf{i}+\mathbf{j}+\mathbf{k})\times\mathbf{k} \\ \mathbf{0} \end{bmatrix} = pI \begin{bmatrix} \mathbf{i}-\mathbf{j} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_3\mathbf{M}_3\mathbf{t}_{32} = \begin{bmatrix} I\boldsymbol{\Omega}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{j} \\ -(a/2)(\mathbf{i}+2\mathbf{k}) \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_3\mathbf{j} \\ \mathbf{0} \end{bmatrix}$$

$$= \begin{bmatrix} pI(\mathbf{i}+\mathbf{j}+\mathbf{k})\times\mathbf{j} \\ \mathbf{0} \end{bmatrix} = pI \begin{bmatrix} -\mathbf{i}+\mathbf{k} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_3\mathbf{M}_3\mathbf{t}_{33} = \begin{bmatrix} I\boldsymbol{\Omega}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{i} \\ -(a/2)\mathbf{j} \end{bmatrix} = \begin{bmatrix} I\boldsymbol{\Omega}_3\mathbf{i} \\ \mathbf{0} \end{bmatrix}$$

$$= \begin{bmatrix} pI(\mathbf{i}+\mathbf{j}+\mathbf{k})\times\mathbf{i} \\ \mathbf{0} \end{bmatrix} = pI \begin{bmatrix} \mathbf{j}-\mathbf{k} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{0}$ now denotes the three-dimensional zero vector. Therefore,

$$\mathbf{WMT} = pI \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{i} & -\mathbf{i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{i}-\mathbf{j} & -\mathbf{i}+\mathbf{k} & \mathbf{j}-\mathbf{k} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

and hence,

$$\mathbf{T}^T\mathbf{WMT} = pI \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

which turns out to be skew-symmetric. Notice, however, that this will not always be the case. The reason why the above product turned out to be skew-symmetric in this example is that the individual matrices $\mathbf{W}_i$ and $\mathbf{M}_i$ commute, a consequence of the assumed inertial isotropy, which leads to the isotropy of matrices $\mathbf{I}_i$, for $i = 1$, 2, and 3. Now, we have

$$\mathbf{T}^T\mathbf{M}\dot{\mathbf{T}} + \mathbf{T}^T\mathbf{WMT} = p\mathbf{A}$$

with $\mathbf{A}$ defined as

$$\mathbf{A} \equiv \begin{bmatrix} -(3/4)a^2m & (7/4)a^2m + I & -(1/2)a^2m - 2I \\ -(1/2)a^2m - I & 0 & (1/4)a^2m + 2I \\ (3/4)a^2m + I & (1/4)a^2m - 2I & 0 \end{bmatrix}$$

Hence, the term of Coriolis and centrifugal forces is

$$(\mathbf{T}^T\mathbf{M}\dot{\mathbf{T}} + \mathbf{T}^T\mathbf{W}\mathbf{M}\mathbf{T})\dot{\boldsymbol{\theta}} = p^2 \begin{bmatrix} (1/2)a^2m - I \\ -(1/4)a^2m + I \\ a^2m - I \end{bmatrix}$$

thereby completing the desired calculations.

Now, in order to verify the correctness of the above results, we will compute the same term using the Newton–Euler algorithm. To this end, we set $\ddot{\theta}_i = 0$, for $i = 1, 2$, and 3, in that algorithm, and calculate the desired expression as the torque required to produce the joint rates given above.

Since we have already calculated the angular velocities, we will skip these calculations here and limit ourselves to center-of-mass velocities, angular accelerations, and center-of-mass accelerations. We thus have

$$\dot{\mathbf{c}}_1 = \boldsymbol{\omega}_1 \times \boldsymbol{\rho}_1 = p\mathbf{k} \times \left(-\frac{1}{2}a\right)(\mathbf{i} - \mathbf{j}) = -\frac{1}{2}ap(\mathbf{i} + \mathbf{j})$$

$$\begin{aligned} \dot{\mathbf{c}}_2 &= \dot{\mathbf{c}}_1 + \boldsymbol{\omega}_1 \times (\mathbf{a}_1 - \boldsymbol{\rho}_1) + \boldsymbol{\omega}_2 \times \boldsymbol{\rho}_2 \\ &= \frac{1}{2}ap[-\mathbf{i} - \mathbf{j} - \mathbf{k} \times (\mathbf{i} + \mathbf{j}) + (\mathbf{j} + \mathbf{k}) \times (\mathbf{i} + \mathbf{j} - \mathbf{k})] = -\frac{1}{2}ap(3\mathbf{i} + \mathbf{j} + \mathbf{k}) \end{aligned}$$

$$\begin{aligned} \dot{\mathbf{c}}_3 &= \dot{\mathbf{c}}_2 + \boldsymbol{\omega}_2 \times (\mathbf{a}_2 - \boldsymbol{\rho}_2) + \boldsymbol{\omega}_3 \times \boldsymbol{\rho}_3 \\ &= -\frac{1}{2}ap[3\mathbf{i} + \mathbf{j} + \mathbf{k} + (\mathbf{j} + \mathbf{k}) \times (\mathbf{i} + \mathbf{k}) - (\mathbf{i} + \mathbf{j} + \mathbf{k}) \times (2\mathbf{i} + \mathbf{k})] \\ &= -\frac{1}{2}ap(3\mathbf{i} + \mathbf{j} + 2\mathbf{k}) \end{aligned}$$

Now, the acceleration calculations are implemented recursively, which yields

$$\dot{\boldsymbol{\omega}}_1 = \ddot{\theta}_1\mathbf{e}_1 = \mathbf{0}$$

$$\dot{\boldsymbol{\omega}}_2 = \dot{\boldsymbol{\omega}}_1 + \boldsymbol{\omega}_1 \times \dot{\theta}_2\mathbf{e}_2 = p^2\mathbf{k} \times \mathbf{j} = -p^2\mathbf{i}$$

$$\dot{\boldsymbol{\omega}}_3 = \dot{\boldsymbol{\omega}}_2 + \boldsymbol{\omega}_2 \times \dot{\theta}_3\mathbf{e}_3 = -p^2\mathbf{i} + p^2(\mathbf{j} + \mathbf{k}) \times \mathbf{i} = -p^2(\mathbf{i} - \mathbf{j} + \mathbf{k})$$

$$\ddot{\mathbf{c}}_1 = \dot{\boldsymbol{\omega}}_1 \times \boldsymbol{\rho}_1 + \boldsymbol{\omega}_1 \times (\boldsymbol{\omega}_1 \times \boldsymbol{\rho}_1) = ap^2\mathbf{k} \times \left[\mathbf{k} \times \frac{1}{2}(-\mathbf{i} + \mathbf{j})\right] = \frac{1}{2}ap^2(\mathbf{i} - \mathbf{j})$$

$$\begin{aligned} \ddot{\mathbf{c}}_2 = \ddot{\mathbf{c}}_1 &+ \dot{\boldsymbol{\omega}}_1 \times (\mathbf{a}_1 - \boldsymbol{\rho}_1) + \boldsymbol{\omega}_1 \times [\boldsymbol{\omega}_1 \times (\mathbf{a}_1 - \boldsymbol{\rho}_1)] + \dot{\boldsymbol{\omega}}_2 \times \boldsymbol{\rho}_2 \\ &+ \boldsymbol{\omega}_2 \times (\boldsymbol{\omega}_2 \times \boldsymbol{\rho}_2) = \frac{1}{2}ap^2(\mathbf{i} - \mathbf{j}) + \mathbf{0} + \frac{1}{2}ap^2(\mathbf{i} + \mathbf{j}) \end{aligned}$$

$$-\frac{1}{2}ap^2(\mathbf{j}+2\mathbf{k})+\frac{1}{2}ap^2(-2\mathbf{i}-3\mathbf{j}+3\mathbf{k})=\frac{1}{2}ap^2(-4\mathbf{j}+\mathbf{k})$$

$$\begin{aligned}\ddot{\mathbf{c}}_3 &= \ddot{\mathbf{c}}_2+\dot{\boldsymbol{\omega}}_2\times(\mathbf{a}_2-\boldsymbol{\rho}_2)+\boldsymbol{\omega}_2\times[\boldsymbol{\omega}_2\times(\mathbf{a}_2-\boldsymbol{\rho}_2)]+\dot{\boldsymbol{\omega}}_3\times\boldsymbol{\rho}_3\\ &= +\boldsymbol{\omega}_3\times(\boldsymbol{\omega}_3\times\boldsymbol{\rho}_3)=\frac{1}{2}ap^2(-4\mathbf{j}+\mathbf{k})-\frac{1}{2}ap^2\mathbf{j}+\frac{1}{2}ap^2(2\mathbf{i}-\mathbf{j}+\mathbf{k})\\ &\quad+\frac{1}{2}ap^2(\mathbf{i}-\mathbf{j}-2\mathbf{k})+\frac{1}{2}ap^2(-3\mathbf{i}+3\mathbf{j})=-2ap^2\mathbf{j}\end{aligned}$$

With the foregoing values, we can now implement the inward Newton–Euler recursions, namely,

$$\begin{aligned}\mathbf{f}_3^P &= m_3\ddot{\mathbf{c}}_3-\mathbf{f}=-m(2ap^2\mathbf{j})-\mathbf{0}=-2amp^2\mathbf{j}\\ \mathbf{n}_3^P &= \mathbf{I}_3\dot{\boldsymbol{\omega}}_3+\boldsymbol{\omega}_3\times\mathbf{I}_3\boldsymbol{\omega}_3-\mathbf{n}+\boldsymbol{\rho}_3\times\mathbf{f}_3^P\\ &= -Ip^2(\mathbf{i}-\mathbf{j}+\mathbf{k})+\mathbf{0}-\mathbf{0}-a^2mp^2(-\mathbf{i}+2\mathbf{k})\\ &= -Ip^2(\mathbf{i}-\mathbf{j}+\mathbf{k})+a^2mp^2(\mathbf{i}-2\mathbf{k})\\ \mathbf{f}_2^P &= m_2\ddot{\mathbf{c}}_2+\mathbf{f}_3^P=\frac{1}{2}amp^2(-4\mathbf{j}+\mathbf{k})-amp^2\mathbf{j}=\frac{1}{2}amp^2(-6\mathbf{j}+\mathbf{k})\\ \mathbf{n}_2^P &= \mathbf{I}_2\dot{\boldsymbol{\omega}}_2+\boldsymbol{\omega}_2\times\mathbf{I}_2\boldsymbol{\omega}_2+\mathbf{n}_3^P+(\mathbf{a}_2-\boldsymbol{\rho}_2)\times\mathbf{f}_3^P+\boldsymbol{\rho}_2\times\mathbf{f}_2^P\\ &= -p^2I\mathbf{i}+\mathbf{0}-Ip^2(\mathbf{i}-\mathbf{j}+\mathbf{k})+\frac{1}{2}a^2mp^2(\mathbf{i}-2\mathbf{k})+a^2mp^2\mathbf{i}\\ &\quad+\frac{1}{4}a^2mp^2(-4\mathbf{i}-\mathbf{j}-6\mathbf{k})=-Ip^2(2\mathbf{i}-\mathbf{j}+\mathbf{k})+\frac{1}{4}a^2mp^2(2\mathbf{i}-\mathbf{j}-10\mathbf{k})\\ \mathbf{f}_1^P &= m_1\ddot{\mathbf{c}}_1+\mathbf{f}_2^P=\frac{1}{2}amp^2(\mathbf{i}-\mathbf{j})+\frac{1}{2}amp^2(-6\mathbf{j}+\mathbf{k})\\ &= \frac{1}{2}amp^2(\mathbf{i}-7\mathbf{j}+\mathbf{k})\\ \mathbf{n}_1^P &= \mathbf{I}_1\dot{\boldsymbol{\omega}}_1+\boldsymbol{\omega}_1\times\mathbf{I}_1\boldsymbol{\omega}_1+\mathbf{n}_2^P+(\mathbf{a}_1-\boldsymbol{\rho}_1)\times\mathbf{f}_2^P+\boldsymbol{\rho}_1\times\mathbf{f}_1^P\\ &= \mathbf{0}+\mathbf{0}-p^2I(2\mathbf{i}-\mathbf{j}+\mathbf{k})+\frac{1}{4}a^2mp^2(2\mathbf{i}-\mathbf{j}-10\mathbf{k})\\ &\quad-\frac{1}{4}a^2mp^2(\mathbf{i}-\mathbf{j}-6\mathbf{k})+\frac{1}{4}a^2mp^2(\mathbf{i}+\mathbf{j}-6\mathbf{k})\\ &= -Ip^2(2\mathbf{i}-\mathbf{j}+\mathbf{k})+\frac{1}{4}a^2mp^2(2\mathbf{i}+\mathbf{j}+2\mathbf{k})\end{aligned}$$

and hence,

$$\tau_3=\mathbf{n}_3^P\cdot\mathbf{e}_3=-Ip^2+a^2mp^2$$

$$\tau_2 = \mathbf{n}_2^P \cdot \mathbf{e}_2 = Ip^2 - \frac{1}{4}a^2mp^2$$

$$\tau_1 = \mathbf{n}_1^P \cdot \mathbf{e}_1 = -Ip^2 + \frac{1}{2}a^2mp^2$$

thereby completing the calculation of the term containing Coriolis and centrifugal forces, i.e.,

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} = \begin{bmatrix} -Ip^2 + (1/2)a^2mp^2 \\ Ip^2 - (1/4)a^2mp^2 \\ -Ip^2 + a^2mp^2 \end{bmatrix}$$

As the reader can verify, the natural orthogonal complement and the Newton–Euler algorithm produce the same result. In the process, the reader may have realized that when performing calculations by hand, the Newton–Euler algorithm is more prone to errors than the natural orthogonal complement, which is more systematic, for it is based on matrix–times–vector multiplications.

7.6.1 Planar Manipulators

The application of the natural orthogonal complement to planar manipulators is straightforward. Here, we assume that the manipulator at hand is composed of n links coupled by n joints of the revolute or the prismatic type. Moreover, for conciseness, we assume that the first link, labeled the base, is fixed to an inertial frame. We now adopt the planar representation of the twists and wrenches introduced in Sect. 5.7; that is, we define the twist of the ith link and the wrench acting on it as three-dimensional arrays, namely,

$$\mathbf{t}_i \equiv \begin{bmatrix} \omega_i \\ \dot{\mathbf{c}}_i \end{bmatrix}, \quad \mathbf{w}_i \equiv \begin{bmatrix} n_i \\ \mathbf{f}_i \end{bmatrix} \tag{7.107}$$

where ω_i is the scalar angular velocity of this link; $\dot{\mathbf{c}}_i$ is the two-dimensional velocity of its center of mass, C_i; n_i is the scalar moment acting on the link; and $\mathbf{f}_i$ is the two-dimensional force acting at C_i. Moreover, the inertia dyad is now a 3×3 matrix, i.e.,

$$\mathbf{M}_i \equiv \begin{bmatrix} I_i & \mathbf{0}^T \\ \mathbf{0} & m_i\mathbf{1} \end{bmatrix} \tag{7.108}$$

with I_i defined as the scalar moment of inertia of the ith link about an axis passing through its center of mass, in the direction normal to the plane of motion, while $\mathbf{0}$ is the two-dimensional zero vector and $\mathbf{1}$ is the 2×2 identity matrix.

Furthermore, the Newton–Euler equations of the ith link take on the forms

$$n_i = I_i \dot{\omega}_i \tag{7.109a}$$

$$\mathbf{f}_i = m_i \ddot{\mathbf{c}}_i \tag{7.109b}$$

and so, these equations can now be cast in the form

$$\mathbf{M}_i \dot{\mathbf{t}}_i = \mathbf{w}_i^W + \mathbf{w}_i^C, \quad i = 1, \ldots, n \tag{7.110}$$

where we have decomposed the total wrench acting on the ith link into its *working* component $\mathbf{w}_i^W$, supplied by the environment and accounting for motor and joint dissipative torques, and $\mathbf{w}_i^C$, the nonworking constraint wrench, supplied by the neighboring links via the coupling joints. The latter, it is recalled, develop no power, their sole role being to keep the links together. An essential difference from the general six-dimensional counterpart of the foregoing equation, namely, Eq. (7.48), is the lack of a quadratic term in ω_i in Eq. (7.109a) and consequently, the lack of a $\mathbf{W}_i \mathbf{M}_i \mathbf{t}_i$ term in Eq. (7.110).

Upon assembling the foregoing $3n$ equations of motion, we obtain a system of $3n$ uncoupled equations in the form

$$\mathbf{M}\dot{\mathbf{t}} = \mathbf{w}^W + \mathbf{w}^C$$

Now, the wrench $\mathbf{w}^W$ accounts for active forces and moments exerted on the manipulator, and so we can decompose this wrench into an actuator-supplied wrench $\mathbf{w}^A$ and a gravity wrench $\mathbf{w}^G$.

In the next step of the formulation, we set up the kinematic constraints in linear homogeneous form, as in Eq. (7.50), with the difference that now, in the presence of n kinematic pairs of the revolute or the prismatic type, $\mathbf{K}$ is a $3n \times 3n$ matrix. Moreover, we set up the twist–shape relations in the form of Eq. (7.56), except that now, $\mathbf{T}$ is a $3n \times n$ matrix. The derivation of the Euler–Lagrange equations for planar motion using the natural orthogonal complement, then, parallels that of general three-dimensional motion, the model sought taking the form

$$\mathbf{I}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} = \boldsymbol{\tau} + \boldsymbol{\gamma} + \boldsymbol{\delta} \tag{7.111a}$$

with the definitions

$$\mathbf{I}(\boldsymbol{\theta}) \equiv \mathbf{T}^T \mathbf{M} \mathbf{T}, \quad \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \equiv \mathbf{T}^T \mathbf{M} \dot{\mathbf{T}}, \tag{7.111b}$$

$$\boldsymbol{\tau} \equiv \mathbf{T}^T \mathbf{w}^A, \quad \boldsymbol{\gamma} \equiv \mathbf{T}^T \mathbf{w}^G, \quad \boldsymbol{\delta} \equiv \mathbf{T}^T \mathbf{w}^D \tag{7.111c}$$

We can illustrate best this formulation with the aid of the example below.

Example 7.6.2 (Dynamics of a Planar Three-Revolute Robot). Derive the model of the robot of Fig. 7.1, under the assumptions of Example 7.3.1, but now using the natural orthogonal complement.

Solution: We start by deriving all kinematics-related variables, and thus,

$$\omega_1 = \dot{\theta}_1, \quad \omega_2 = \dot{\theta}_1 + \dot{\theta}_2, \quad \omega_3 = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3$$

Furthermore,

$$\begin{aligned}
\mathbf{t}_1 &= \dot{\theta}_1 \mathbf{t}_{11} \\
\mathbf{t}_2 &= \dot{\theta}_1 \mathbf{t}_{21} + \dot{\theta}_2 \mathbf{t}_{22} \\
\mathbf{t}_3 &= \dot{\theta}_1 \mathbf{t}_{31} + \dot{\theta}_2 \mathbf{t}_{32} + \dot{\theta}_3 \mathbf{t}_{33}
\end{aligned}$$

where

$$\mathbf{t}_{11} = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{s}_{11} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}\boldsymbol{\rho}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ (1/2)\mathbf{E}\mathbf{a}_1 \end{bmatrix}$$

$$\mathbf{t}_{21} = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{s}_{21} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_1 + \boldsymbol{\rho}_2) \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_1 + (1/2)\mathbf{a}_2) \end{bmatrix}$$

$$\mathbf{t}_{22} = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{s}_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}\boldsymbol{\rho}_2 \end{bmatrix} = \begin{bmatrix} 1 \\ (1/2)\mathbf{E}\mathbf{a}_2 \end{bmatrix}$$

$$\mathbf{t}_{31} = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{s}_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_1 + \mathbf{a}_2 + \boldsymbol{\rho}_3) \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_1 + \mathbf{a}_2 + (1/2)\mathbf{a}_3) \end{bmatrix}$$

$$\mathbf{t}_{32} = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{s}_{32} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_2 + \boldsymbol{\rho}_3) \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{E}(\mathbf{a}_2 + (1/2)\mathbf{a}_3) \end{bmatrix}$$

$$\mathbf{t}_{33} = \begin{bmatrix} 1 \\ \mathbf{E}\boldsymbol{\rho}_3 \end{bmatrix} = \begin{bmatrix} 1 \\ (1/2)\mathbf{E}\mathbf{a}_3 \end{bmatrix}$$

and hence, the 9×3 twist-shaping matrix $\mathbf{T}$ becomes

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ (1/2)\mathbf{E}\mathbf{a}_1 & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 \\ \mathbf{E}(\mathbf{a}_1 + (1/2)\mathbf{a}_2) & (1/2)\mathbf{E}\mathbf{a}_2 & \mathbf{0} \\ 1 & 1 & 1 \\ \mathbf{E}(\mathbf{a}_1 + \mathbf{a}_2 + (1/2)\mathbf{a}_3) & \mathbf{E}(\mathbf{a}_2 + (1/2)\mathbf{a}_3) & (1/2)\mathbf{E}\mathbf{a}_3 \end{bmatrix}$$

The 9×9 matrix of inertia dyads of this manipulator now takes the form

$$\mathbf{M} = \mathrm{diag}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3)$$

with each 3×3 $\mathbf{M}_i$ matrix defined as

$$\mathbf{M}_i \equiv \begin{bmatrix} I_i & \mathbf{0}^T \\ \mathbf{0} & m_i \mathbf{1} \end{bmatrix}$$

Now, the 3×3 generalized inertia matrix is readily derived as

$$\mathbf{I} \equiv \mathbf{T}^T \mathbf{M} \mathbf{T}$$

whose entries are given below:

$$\begin{aligned}
I_{11} &= \mathbf{t}_{11}^T \mathbf{M}_1 \mathbf{t}_{11} + \mathbf{t}_{21}^T \mathbf{M}_2 \mathbf{t}_{21} + \mathbf{t}_{31}^T \mathbf{M}_3 \mathbf{t}_{31} \\
I_{12} &= \mathbf{t}_{21}^T \mathbf{M}_2 \mathbf{t}_{22} + \mathbf{t}_{31}^T \mathbf{M}_3 \mathbf{t}_{32} = I_{21} \\
I_{13} &= \mathbf{t}_{31}^T \mathbf{M}_3 \mathbf{t}_{33} = I_{31} \\
I_{22} &= \mathbf{t}_{22}^T \mathbf{M}_2 \mathbf{t}_{22} + \mathbf{t}_{32}^T \mathbf{M}_3 \mathbf{t}_{32} \\
I_{23} &= \mathbf{t}_{32}^T \mathbf{M}_3 \mathbf{t}_{33} = I_{32} \\
I_{33} &= \mathbf{t}_{33}^T \mathbf{M}_3 \mathbf{t}_{33}
\end{aligned}$$

Upon expansion, the above entries result in exactly the same expressions as those derived in Example 7.3.1, thereby confirming the correctness of the two derivations. Furthermore, the next term in the Euler–Lagrange equations is derived below. Here, we will need $\dot{\mathbf{T}}$, which is readily derived from the above expression for $\mathbf{T}$. In deriving this time-derivative, we note that in general, for $i = 1, 2, 3$,

$$\dot{\mathbf{a}}_i = \omega_i \mathbf{E} \mathbf{a}_i, \quad \mathbf{E}^2 \mathbf{a}_i = -\mathbf{a}_i$$

and hence,

$$\dot{\mathbf{T}} = -\begin{bmatrix}
0 & 0 & 0 \\
(1/2)\dot{\theta}_1 \mathbf{a}_1 & \mathbf{0} & \mathbf{0} \\
0 & 0 & 0 \\
\dot{\theta}_1 \mathbf{a}_1 + (1/2)\dot{\theta}_{12} \mathbf{a}_2 & (1/2)\dot{\theta}_{12} \mathbf{a}_2 & \mathbf{0} \\
0 & 0 & 0 \\
\dot{\theta}_1 \mathbf{a}_1 + \dot{\theta}_{12} \mathbf{a}_2 + (1/2)\dot{\theta}_{123} \mathbf{a}_3 & \dot{\theta}_{12} \mathbf{a}_2 + (1/2)\dot{\theta}_{123} \mathbf{a}_3 & (1/2)\dot{\theta}_{123} \mathbf{a}_3
\end{bmatrix}$$

where $\dot{\theta}_{12}$ and $\dot{\theta}_{123}$ stand for $\dot{\theta}_1 + \dot{\theta}_2$ and $\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3$, respectively.

We now can perform the product $\mathbf{T}^T \mathbf{M} \dot{\mathbf{T}}$, whose (i, j) entry will be represented as μ_{ij}. Below we display the expressions for these entries:

$$\mu_{11} = -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})]\dot{\theta}_2 - \frac{m_3}{2}(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_3$$

$$\mu_{12} = -\frac{1}{2}[m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})](\dot{\theta}_1 + \dot{\theta}_2)$$

$$
\begin{aligned}
&-\frac{1}{2}m_3(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_3\\
\mu_{13} &= -\frac{1}{2}m_3(a_1a_3s_{23}+a_2a_3s_3)(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\\
\mu_{21} &= \frac{1}{2}[m_2a_1a_2s_2+m_3(2a_1a_2s_2+a_1a_3s_{23})]\dot{\theta}_1-\frac{1}{2}m_3a_2a_3s_3\dot{\theta}_3\\
\mu_{22} &= -\frac{1}{2}m_3a_2a_3s_3\dot{\theta}_3\\
\mu_{23} &= -\frac{1}{2}m_3a_2a_3s_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)\\
\mu_{31} &= \frac{1}{2}m_3[(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_1+a_2a_3s_3\dot{\theta}_2]\\
\mu_{32} &= \frac{1}{2}m_3a_2a_3s_3(\dot{\theta}_1+\dot{\theta}_2)\\
\mu_{33} &= 0
\end{aligned}
$$

Now, let us define

$$\boldsymbol{\nu} \equiv \mathbf{T}^T\mathbf{M}\dot{\mathbf{T}}\dot{\boldsymbol{\theta}}$$

whose three components are given below:

$$
\begin{aligned}
\nu_1 &= -[m_2a_1a_2s_2+m_3(2a_1a_2s_2+a_1a_3s_{23})]\dot{\theta}_1\dot{\theta}_2-m_3(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_1\dot{\theta}_3\\
&\quad -\frac{1}{2}[m_2a_1a_2s_2+m_3(2a_1a_2s_2+a_1a_3s_{23})]\dot{\theta}_2^2\\
&\quad -m_3(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_2\dot{\theta}_3-\frac{1}{2}m_3(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_3^2\\
\nu_2 &= \frac{1}{2}[m_2a_1a_2s_2+m_3(2a_1a_2s_2+a_1a_3s_{23})]\dot{\theta}_1^2-m_3a_2a_3s_3\dot{\theta}_1\dot{\theta}_3\\
&\quad -m_3a_2a_3s_3\dot{\theta}_2\dot{\theta}_3-\frac{1}{2}m_3a_2a_3s_3\dot{\theta}_3^2\\
\nu_3 &= \frac{1}{2}m_3(a_1a_3s_{23}+a_2a_3s_3)\dot{\theta}_1^2+m_3a_2a_3s_3\dot{\theta}_1\dot{\theta}_2+\frac{1}{2}m_3a_2a_3s_3\dot{\theta}_3^2
\end{aligned}
$$

The mathematical model sought, thus, takes the form

$$\mathbf{I}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}+\boldsymbol{\nu}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})=\boldsymbol{\tau}+\boldsymbol{\gamma}$$

where $\boldsymbol{\delta}=\mathbf{0}$ because we have not included dissipation. Moreover, $\boldsymbol{\gamma}$ is derived as described below: Let $\mathbf{w}_i^G$ be the gravity wrench acting on the ith link, $\mathbf{w}^G$ then being

$$\mathbf{w}^G = \begin{bmatrix} \mathbf{w}_1^G \\ \mathbf{w}_2^G \\ \mathbf{w}_3^G \end{bmatrix}$$

and

$$\mathbf{w}_1^G = \begin{bmatrix} 0 \\ -m_1 g\mathbf{j} \end{bmatrix}, \quad \mathbf{w}_2^G = \begin{bmatrix} 0 \\ -m_2 g\mathbf{j} \end{bmatrix}, \quad \mathbf{w}_3^G = \begin{bmatrix} 0 \\ -m_3 g\mathbf{j} \end{bmatrix}$$

Therefore,

$$\boldsymbol{\gamma} = \mathbf{T}^T \mathbf{w}^G$$

i.e.,

$$\boldsymbol{\gamma} = \frac{g}{2} \begin{bmatrix} m_1 \mathbf{a}_1^T \mathbf{E}\mathbf{j} + m_2(2\mathbf{a}_1 + \mathbf{a}_2)^T \mathbf{E}\mathbf{j} + m_3[2(\mathbf{a}_1 + \mathbf{a}_2) + \mathbf{a}_3)^T \mathbf{E}\mathbf{j} \\ m_2 \mathbf{a}_1^T \mathbf{E}\mathbf{j} + m_3(2\mathbf{a}_2 + \mathbf{a}_3)^T \mathbf{E}\mathbf{j} \\ m_3 \mathbf{a}_3^T \mathbf{E}\mathbf{j} \end{bmatrix}$$

But

$$\begin{aligned} \mathbf{a}_1^T \mathbf{E}\mathbf{j} &= -\mathbf{a}_1^T \mathbf{i} = -a_1 \cos\theta_1 \\ \mathbf{a}_2^T \mathbf{E}\mathbf{j} &= -\mathbf{a}_2^T \mathbf{i} = -a_2 \cos(\theta_1 + \theta_2) \\ \mathbf{a}_3^T \mathbf{E}\mathbf{j} &= -\mathbf{a}_3^T \mathbf{i} = -a_3 \cos(\theta_1 + \theta_2 + \theta_3) \end{aligned}$$

Hence,

$$\boldsymbol{\gamma} = \frac{g}{2} \begin{bmatrix} -m_1 a_1 c_1 - 2m_2(a_1 c_1 + a_2 c_{12}) - 2m_3(a_1 c_1 + a_2 c_{12} + a_3 c_{123}) \\ -m_2 a_2 c_{12} - 2m_3(a_2 c_{12} + a_3 c_{123}) \\ -m_3 a_3 c_{123} \end{bmatrix}$$

with the definitions for c_1, c_{12}, and c_{123} introduced in Example 7.3.1. As the reader can verify, the foregoing model is identical to the model derived with the Euler–Lagrange equations in that example.

7.6.2 Algorithm Complexity

The complexity of this algorithm is analyzed with regard to the three items involved, namely, (a) the evaluation of $\mathbf{L}$, (b) the solution of systems (7.105a and b), and (c) the computation of $\overline{\boldsymbol{\tau}}$.

The evaluation of $\mathbf{L}$ involves, in turn, the three following steps: (a) the computation of $\mathbf{P}$; (b) the computation of $\mathbf{I}$; and (c) the Cholesky decomposition of $\mathbf{I}$ into the product $\mathbf{L}^T\mathbf{L}$.

Algorithm 7.6.1:

```
For j=1 to n step 1 do
```
$$\mathbf{r}_{jj} \leftarrow [\boldsymbol{\rho}_j]_{j+1}$$
$$\mathbf{p}_{jj} \leftarrow \begin{bmatrix} \mathbf{N}_j\mathbf{e}_j \\ n_j\mathbf{e}_j \times \mathbf{r}_{jj} \end{bmatrix}_{j+1}$$
```
    For i=j+1 to n step 1 do
```
$$\mathbf{e}_j \leftarrow \mathbf{Q}_i^T[\mathbf{e}_j]_i$$
```
        if R then
```
$$\mathbf{r}_{ij} \leftarrow \mathbf{Q}_i^T[\mathbf{r}_{i-1,j} + \boldsymbol{\delta}_{i-1}]_i + [\boldsymbol{\rho}_i]_{i+1}$$
$$\mathbf{p}_{ij} \leftarrow \begin{bmatrix} \mathbf{N}_i\mathbf{e}_j \\ n_i\mathbf{e}_j \times \mathbf{r}_{ij} \end{bmatrix}_{i+1}$$
```
        else
```
$$\mathbf{p}_{ij} \leftarrow \begin{bmatrix} \mathbf{0} \\ n_i\mathbf{e}_j \end{bmatrix}_{i+1}$$
```
        endif
    enddo
enddo
```

(i.a) In the computation of $\mathbf{P}$, it is recalled that $\mathbf{H}_i$, $\mathbf{a}_i$, and $\boldsymbol{\rho}_i$, and consequently, $\boldsymbol{\delta}_i \equiv \mathbf{a}_i - \boldsymbol{\rho}_i$, are constant in $\mathcal{F}_{i+1}$, which is the frame fixed to the ith link. Moreover, at each step of the algorithm, both revolute and prismatic pairs are considered. If the jth joint is a revolute, then the logical variable `R` is `true`; if this joint is prismatic, then `R` is `false`. Additionally, it is recalled that $\mathbf{e}_{i+1}$, in $\mathcal{F}_i$-coordinates, is simply the last column of $\mathbf{Q}_i$. The columnwise evaluation of $\mathbf{P}$, with each $\mathbf{p}_{ij}$ array in $\mathcal{F}_{i+1}$-coordinates, is described in Algorithm 7.6.1. Note that in this algorithm, $\mathbf{r}_{ij}$ is calculated recursively from $\mathbf{r}_{i-1,j}$. To do this, we use the relation between these two vectors, as displayed in Fig. 7.8.

(i.b) Now we go on to the computation of $\mathbf{I}$, as described in Algorithm 7.6.2. In that algorithm, the subscripted brackets indicate that the vectors inside these brackets are represented in $\mathcal{F}_{k+1}$ coordinates.

(i.c) Because the Cholesky decomposition of a positive-definite matrix is a standard item, it is not discussed here. This step completes the computation of $\mathbf{L}$.

(ii) The solution of systems (7.105a and 7.105b) is a standard issue as well, and hence, needs no further discussion.

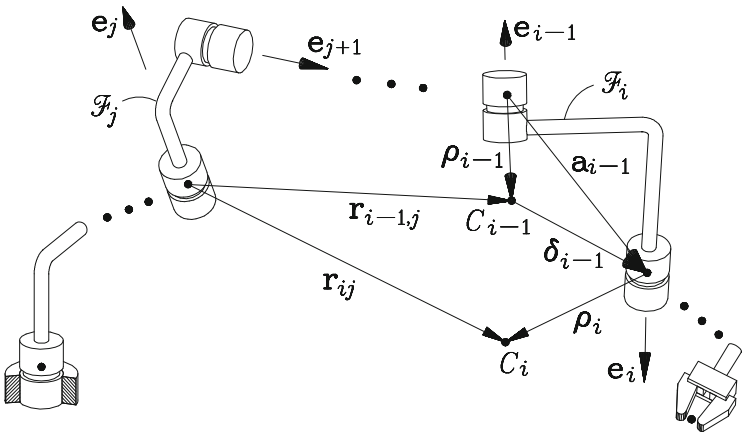

Fig. 7.8 Recursive calculation of vectors $\mathbf{r}_{ij}$

(iii) The term $\overline{\boldsymbol{\tau}}$ is computed using the recursive Newton–Euler formulation, as discussed in Sect. 7.4. To do this, we calculate $\overline{\boldsymbol{\tau}}$ by setting $\ddot{\boldsymbol{\theta}} = \mathbf{0}$ in that procedure, which introduces a slight simplification of the complexity of the inverse-dynamics algorithm.

Algorithm 7.6.2:

```
For j=1 to n step 1 do
```

$$I_{jj} \leftarrow \sum_{k=j}^{n} [\, \mathbf{p}_{kj}^T \mathbf{p}_{kj} \,]_{k+1}$$

```
    For i=j+1 to n step 1 do
```

$$I_{ij} \leftarrow I_{ji} \leftarrow \sum_{k=i}^{n} [\, \mathbf{p}_{ki}^T \mathbf{p}_{kj} \,]_{k+1}$$

```
    enddo
enddo
```

Below we determine the computational complexity of each of the foregoing steps.

(i.a) This step includes Algorithm 7.6.1, which involves two nested do-loops. The first statement of the outermost loop involves no floating-point operations; the second statement involves (a) one multiplication of a matrix by a vector, (b) one cross product, and (c) one multiplication of a scalar, $n_j (= \sqrt{m_j})$, by a vector, $[\, \mathbf{e}_j \,]_{j+1}$, which, according to Eq. (4.15), equals $\mathbf{o}_j$, the third row of $\mathbf{Q}_j$. In light of Eq. (4.7), this vector involves only two nonzero components, the product $n_j [\, \mathbf{e}_j \,]_{j+1}$ thus consuming only two multiplications

and zero additions. Of the last three items, (a) is done off-line, for the matrix and the vector factors are both constant in $\mathcal{F}_{j+1}$-coordinates, and so, this operation is not counted. Moreover, item (b) is nothing but the cross product of vector $[\mathbf{e}_j]_{j+1} \equiv [0, 0, 1]^T$ by vector $\mathbf{r}_{jj}$. A similar operation was already discussed in connection with Algorithm 4.1 and was found to involve zero floating-point operations, for the result is, simply, $[\mathbf{e}_j \times \mathbf{r}_{jj}]_{j+1} = [-y, x, 0]^T$, with x and y denoting the X_{j+1} and Y_{j+1} components of $\mathbf{r}_{jj}$. Hence, item (b) requires no floating-point operations, while item (c) requires $2n$ multiplications and zero additions.

The innermost do-loop, as pertaining to revolute manipulators, involves two coordinate transformations between *two consecutive coordinate frames*, from $\mathcal{F}_i$- to $\mathcal{F}_{i+1}$-coordinates, plus one vector sum, which consumes $16(n-i)$ multiplications and $11(n-i)$ additions; this loop also consumes one matrix-times-vector multiplication, one cross product and one scalar-times-vector multiplication, which requires $18(n-i)$ multiplications and $9(n-i)$ additions. Thus, the total numbers of operations required by this step, for an n-revolute manipulator, are M_{ia} multiplications and A_{ia} additions, as given below:

$$M_{ia} = 2n + \sum_{i=1}^{n} 34(n-i) = 17n^2 - 15n \tag{7.112a}$$

$$A_{ia} = \sum_{i=1}^{n} 20(n-i) = 10n^2 - 10n \tag{7.112b}$$

the presence of prismatic pairs reducing the above figures.

(i.b) This step, summarized in Algorithm 7.6.2, is also composed of two do-loops, each containing the inner product of two six-dimensional arrays, and hence, requires six multiplications and five additions. Moreover, in the outermost do-loop, this operation is performed n times, whereas in the innermost loop, $\sum_{i=1}^{n}(n-i)$ times, i.e., $n(n-1)/2$ times. Thus, the step requires M_{ib} multiplications and A_{ib} additions, as given below:

$$M_{ib} = 3n^2 + 3n, \quad A_{ib} = \frac{5}{2}n^2 + \frac{5}{2}n \tag{7.113}$$

(i.c) This step performs the Cholesky decomposition of a $n \times n$ symmetric and positive-definite matrix, a standard operation that requires M_{ic} multiplications and A_{ic} additions (Dahlquist and Björck 1974), namely,

$$M_{ic} = \frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n, \quad A_{ic} = \frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n \tag{7.114}$$

(ii) In this step, the two triangular systems of equations, Eqs. (7.105a and b), are solved first for $\mathbf{x}$ and then for $\ddot{\boldsymbol{\theta}}$. The numbers of operations it takes to solve each of the two systems, as derived by Dahlquist and Björck (1974),

are repeated below for quick reference; these are labelled M_{ii} and A_{ii}, respectively, i.e.,

$$M_{ii} = n^2, \quad A_{ii} = n^2 - n \tag{7.115}$$

(iii) In this step, $\overline{\boldsymbol{\tau}}$ is computed from inverse dynamics, with $\mathbf{w}^D = \mathbf{0}$ and $\ddot{\boldsymbol{\theta}} = \mathbf{0}$. If this calculation is done with the Newton–Euler formulation, we then have the computational costs given in Eq. (7.43), and reproduced below for quick reference:

$$M_{iii} = 137n - 22, \quad A_{iii} = 110n - 14 \tag{7.116}$$

Because of the simplifications introduced by setting the joint accelerations equal to zero, the foregoing figures are, in fact, slightly lower than those required by the general recursive Newton–Euler algorithm.

Thus, the total numbers of multiplications and additions required for the forward dynamics of a n-revolute, serial manipulator are

$$M_f = \frac{1}{6}n^3 + \frac{37}{2}n^2 + \frac{367}{3}n - 22, \quad A_f = \frac{1}{6}n^3 + \frac{23}{2}n^2 + \frac{298}{3}n - 14 \tag{7.117}$$

In particular, for a six-revolute manipulator, one obtains

$$M_f = 1,450, \quad A_f = 1,068 \tag{7.118}$$

Upon introducing a modified Denavit–Hartenberg labeling of coordinate frames and a very careful management of the computations involved the number of floating-point operations became 1,353 multiplications and 1,165 additions (Angeles and Ma 1988). Nevertheless, the total number of operations, 2,518, remained the same.

7.6.3 Simulation

The purpose of the algorithm introduced above is to enable us to predict the behavior of a given manipulator under given initial conditions, applied torques, and applied loads. The ability of predicting this behavior is important for several reasons: for example, in design, we want to know whether with a given selection of motors, the manipulator will be able to perform a certain typical task in a given time frame; in devising feedback control schemes, where stability is a major concern, the control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Hence, a facility capable of predicting the behavior of a robotic manipulator, or of a system at large, for that matter, becomes imperative.

The procedure whereby the motion of the manipulator is determined from initial conditions and applied torques and loads is known as *simulation*. Since we start

with a second-order n-dimensional system of ODE in the joint variables of the manipulator, we have to integrate this system in order to determine the time-histories of all joint variables, which are grouped in vector $\boldsymbol{\theta}$. With current software available, this task has become routine work, the user being freed from the quite demanding task of writing code for integrating systems of ODE. Below we discuss a few issues pertaining to the implementation of the simulation-related algorithms available in commercial software packages.

As a rule, simulation code requires that the user supply a state-variable model of the form of Eq. (7.45), with the state-variable vector, or state-vector for brevity, $\mathbf{x}$, and the *input* or *control vector* $\mathbf{u}$ defined as

$$\mathbf{x} \equiv \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\psi} \end{bmatrix}, \quad \mathbf{u}(t) = \boldsymbol{\tau}(t) \tag{7.119}$$

With the above definitions, then we can write the state-variable equations, or state equations for brevity, in the form of Eq. (7.45), with $\mathbf{f}(\mathbf{x}, \boldsymbol{\tau})$ given by

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\tau}) \equiv \begin{bmatrix} \boldsymbol{\psi} \\ -\mathbf{I}(\boldsymbol{\theta})^{-1}[\mathbf{C}(\boldsymbol{\theta}, \boldsymbol{\psi})\boldsymbol{\psi} - \boldsymbol{\delta}(\boldsymbol{\theta}, \boldsymbol{\psi}) - \boldsymbol{\gamma}(\boldsymbol{\theta})] + \mathbf{I}(\boldsymbol{\theta})^{-1}\boldsymbol{\tau}(t) \end{bmatrix} \tag{7.120}$$

thereby obtaining a system of $2n$ first-order ODE in the state-variable vector $\mathbf{x}$ defined above. Various methods are available to solve the ensuing initial-value problem, all of them being based on a discretization of the time variable. That is, if the behavior of the system is desired in the interval $t_0 \leq t \leq t_F$, then the software implementing these methods provides *approximations* $\{\mathbf{y}_k\}_1^N$ to the state-variable vector at a discrete set of instants, $\{t_k\}_0^N$, with $t_N \equiv t_F$.

The variety of methods available to solve the underlying initial-value problem can be classified into two main categories, *explicit methods* and *implicit methods*. The former provide $\mathbf{y}_{k+1}$ *explicitly* in terms of previously computed values. On the contrary, implicit methods provide $\mathbf{y}_{k+1}$ in terms of previously computed values $\mathbf{y}_k$, $\mathbf{y}_{k-1}$, ..., etc., *and* $\mathbf{y}_{k+1}$ itself. For example, in the simplest of implicit methods, namely, the *backward Euler method*, we can approximate the integral of $\mathbf{f}$ in the interval $t_k \leq t \leq t_{k+1}$ by resorting to the *trapezoidal rule* (Kahaner et al. 1989), which leads to the expression

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) \tag{7.121}$$

In Eq. (7.121), h_k is the *current* time-step $t_{k+1} - t_k$ and $\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})$ can be an arbitrary function of $\mathbf{y}_{k+1}$. If this function is nonlinear in the said variable, then, a *direct*—as opposed to *iterative*—computation of $\mathbf{y}_{k+1}$ is very unlikely. Hence, most likely an iterative scheme must be implemented at every integration stage of an implicit method. While this feature might render implicit schemes unattractive, they offer interesting advantages. Indeed, the iterative procedure mentioned above requires a tolerance to decide when and whether the procedure has converged. The

convergence criterion imposed thus brings about a self-correcting effect that helps keep the unavoidable *truncation error* under control. This error is incurred when approximating both the time derivative $\dot{\mathbf{x}}$ and the integral of $\mathbf{f}$ by floating-point operations.

Current software provides routines for both implicit and explicit methods, the user having to decide which method to invoke. Of the explicit methods in use, by far the most common ones are the *Runge–Kutta methods*. Of these, there are several versions, depending on the number of evaluations of the function $\mathbf{f}(t_i, \mathbf{y}_i)$, for various values of i, that they require. A two-stage Runge–Kutta method, for example, requires two function evaluations, while a four-stage Runge–Kutta method requires four. The self-correcting feature of implicit methods, not present in Runge–Kutta methods—to be sure, *implicit Runge–Kutta methods* also exist (Gear 1971), but these are less common than their explicit counterparts—is compensated for by a clever strategy that consists in computing $\mathbf{y}_{k+1}$ using two Runge–Kutta schemes of different numbers of stages. What is at stake here is the magnitude of the *local error* in computing $\mathbf{y}_{k+1}$, under the assumption that $\mathbf{y}_k$ is error-free. Here, the magnitude of the error is of order h^p, where p is the *order* of the method in use. In Runge–Kutta methods, the order of the method is identical to its number of stages. In general, a method is said to be of order p if it is capable of computing *exactly* the integral of an ordinary differential equation, provided that the solution is known to be a pth-degree polynomial. Now, upon computing $\mathbf{y}_{k+1}$ using two Runge–Kutta schemes with N and $N + 1$ stages, we can compare the two computed values reported by each method, namely, $\mathbf{y}_{k+1}^N$ and $\mathbf{y}_{k+1}^{N+1}$. If a *norm* of the difference of these two values is smaller than a user-prescribed tolerance, then the step size in use is acceptable. If not, then the step size is halved, and the process is repeated until the foregoing norm is within the said tolerance. The most common Runge–Kutta methods are those combining two and three stages and those combining four and five.

A drawback of Runge–Kutta methods is their inability to deal with what are known as *stiff systems*, first identified by Gear (1971). As defined by Shampine and Gear (1979), a system of ordinary differential equations is said to be stiff if it is not unstable and its linear part—i.e., the linear part of the series expansion of $\mathbf{f}$, evaluated at the current instant—comprises a coefficient matrix that has an eigenvalue with a negative real part whose absolute value is much greater than that of the other eigenvalues. In other words, stiff systems of ODE are stable systems with very different time scales. Thus, stiff systems are not inherently difficult to integrate, but they require a special treatment. Gear's method, which is implicit, provides exactly the means to handle stiff systems. However, methods like Runge–Kutta's, with excellent performance for nonstiff systems, perform rather poorly for stiff systems, and the other way around. The mathematical models that arise in robotic mechanical systems are likely to be stiff because of the various orders of magnitude of the physical parameters involved. For example, robotic manipulators are provided, usually, with links close to the base that are heavy and with links far from the base that are light. As a consequence, when simulating robotic mechanical systems, a provision must be made for numerical stiffness.

Commercial software for scientific computations offers Runge–Kutta methods of various orders, with combinations thereof. For example, IMSL offers excellent FORTRAN routines, like `IVPRK`, for the implementation of Runge–Kutta methods, while Matlab's Simulink toolbox offers the C functions `ode23` and `ode45` for the implementation of second-and-third and fourth-and-fifth-order Runge–Kutta methods. With regard to stiff systems, IMSL offers a subroutine, `IVPAG`, implementing both Adams's and Gear's methods, while Simulink offers the `adams` and `gear` functions for the implementation of either of these. Since Matlab is written in C, communication between Matlab and FORTRAN programs is not as direct as when using IMSL, which may be disappointing to FORTRAN users. Details on linking FORTRAN code with Matlab and other related issues are discussed in the pertinent literature (Etter 1997). Moreover, the FORTRAN `SDRIV2` subroutine (Kahaner et al. 1989) comprises features that allow it to handle both stiff and nonstiff systems.

7.7 Incorporation of Gravity into the Dynamics Equations

Manipulators subjected to gravity fields have been discussed in Sect. 7.4 in connection with the Newton–Euler algorithm and with Kane's equations. As found in that section, gravitational forces can be incorporated into the underlying models without introducing any major modifications that would increase the computational load if the method of Luh et al. (1980) is adopted. Within this approach, gravitational forces are taken into account by defining the acceleration of the center of mass of the 0th link, the base link, as equal to $-\mathbf{g}$, the negative of the gravity-acceleration vector. The effect of this approach is to propagate the gravity effect into all the links composing the manipulator. Thus, the kinematics algorithm of Sect. 7.4 need not be modified in order to include gravity forces, for all that is needed is to declare

$$[\,\ddot{\mathbf{c}}_0\,]_1 \quad \leftarrow \quad [\,-\mathbf{g}\,]_1 \tag{7.122}$$

If inverse dynamics is computed with the natural orthogonal complement, then the twist-rate of the first link will have to be modified by adding a nonhomogeneous term to it, thereby accounting for the gravity-acceleration terms. That is,

$$\dot{\mathbf{t}}_1 \quad \leftarrow \quad \ddot{\theta}_1 \mathbf{t}_{11} + \dot{\theta}_1 \dot{\mathbf{t}}_{11} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{g} \end{bmatrix} \tag{7.123}$$

Otherwise, the foregoing algorithms require no modifications. Furthermore, with regard to simulation, it is pointed out that the $\overline{\boldsymbol{\tau}}$ term defined in Eq. (7.104), and appearing in the right-hand side of Eq. (7.105a), is computed from inverse dynamics with zero frictional forces and zero joint accelerations.

7.8 The Modeling of Dissipative Forces

Broadly speaking, frictional forces are of two basic types, namely, (a) viscous forces and (b) *Coulomb*, or dry-friction, forces. The latter occur when contact between two solids takes place directly, the former when contact between the solids takes place via a viscous fluid, e.g., a lubricant. In the analysis of viscous fluids, a basic assumption is that the relative velocity between the fluid and the solid vanishes at the fluid–solid interface, i.e., at the solid boundary confining the fluid. Hence, a *velocity gradient* appears within the fluid, which is responsible for the power dissipation inside it. In fact, not all the velocity gradient within the fluid, but only its *symmetric part*, is responsible for power dissipation; the *skew-symmetric part* of the velocity gradient accounts for a rigid-body rotation of a small fluid element. Thus, if a velocity field $\mathbf{v}(\mathbf{r}, t)$ is defined within a region $\mathcal{R}$ occupied by a viscous fluid, for a point of the fluid of position vector $\mathbf{r}$ at a time t, then, the velocity gradient $\text{grad}(\mathbf{v}) \equiv \partial \mathbf{v}/\partial \mathbf{r}$, can be decomposed as

$$\text{grad}(\mathbf{v}) = \mathbf{D} + \mathbf{W} \tag{7.124}$$

where $\mathbf{D}$ and $\mathbf{W}$ are the symmetric and the skew-symmetric parts of the velocity gradient, i.e.,

$$\mathbf{D} \equiv \frac{1}{2}[\,\text{grad}(\mathbf{v}) + \text{grad}^T(\mathbf{v})\,], \quad \mathbf{W} \equiv \frac{1}{2}[\,\text{grad}(\mathbf{v}) - \text{grad}^T(\mathbf{v})\,] \tag{7.125}$$

The kinematic interpretation of $\mathbf{D}$ and $\mathbf{W}$ is given below: The former accounts for a *distorsion* of an infinitesimally small spherical element of fluid into a three-axis ellipsoid, the ratios of the time *rates* of change of the lengths of the three axes being identical to the ratios of the real eigenvalues of $\mathbf{D}$; the latter accounts for the angular velocity of the ellipsoid as a rigid-body. Clearly, both $\mathbf{D}$ and $\mathbf{W}$ change from point to point within the fluid and also from time to time, i.e.,

$$\mathbf{D} = \mathbf{D}(\mathbf{r}, t), \quad \mathbf{W} = \mathbf{W}(\mathbf{r}, t) \tag{7.126}$$

Since the skew-symmetric matrix $\mathbf{W}$ accounts only for the rotation of a differential element of fluid as a rigid body, it cannot be responsible for any energy dissipation, and hence, the only part that is responsible for this is $\mathbf{D}$. In fact, for a *linearly viscous, incompressible* fluid of viscosity coefficient μ, the power dissipated within $\mathcal{R}$ is given by

$$\Pi^D = \int_{\mathcal{R}} \mu \text{tr}(\mathbf{D}^2) d\mathcal{R} \tag{7.127}$$

Now, if the motion of the lubricant separating the two cylindrical surfaces of a revolute pair is modeled as a purely tangential velocity field (Currie 1993), which assumes that the two cylinders remain concentric, then the foregoing expression for Π^D leads to the dissipation function

$$\Delta = \frac{1}{2}\beta\dot{\theta}^2 \tag{7.128}$$

where $\dot{\theta}$ is the relative angular speed between the two cylinders and the coefficient β is a function of the lubricant viscosity and the geometry of the kinematic pair at hand. If the kinematic pair under study is prismatic, then we can model the motion of the lubricant between the two prismatic surfaces as a *Couette flow* between a pair of parallel surfaces of the sides of the prism. Under these conditions, then, the associated dissipation function Δ takes on the same form of that given for a revolute pair in Eq. (7.128), in which the sole difference is that $\dot{\theta}$ changes to $\dot{b}$, the time rate of change of the associated joint variable. Of course, $\dot{b}$ is the relative speed between the two prismatic surfaces. Thus in any event, the dissipation function of the ith joint due to linearly viscous effects can be written as

$$\Delta_i = \frac{1}{2}\beta_i\dot{\theta}_i^2 \tag{7.129}$$

where $\dot{\theta}_i$ changes to $\dot{b}_i$ if the ith pair is prismatic. The dissipation function thus arising then reduces to

$$\Delta = \sum_1^n \Delta_i = \frac{1}{2}\dot{\boldsymbol{\theta}}^T\mathbf{B}\dot{\boldsymbol{\theta}} \tag{7.130}$$

where the constant $n \times n$ matrix $\mathbf{B}$ is given by

$$\mathbf{B} = \text{diag}(\beta_1,\ \beta_2, \ldots,\ \beta_n) \tag{7.131}$$

and hence, the generalized force $\boldsymbol{\delta}^V$ associated with linearly viscous effects is *linear* in the vector of joint rates, $\dot{\boldsymbol{\theta}}$, i.e.,

$$\boldsymbol{\delta}^V \equiv -\frac{\partial\Delta}{\partial\dot{\boldsymbol{\theta}}} = -\mathbf{B}\dot{\boldsymbol{\theta}} \tag{7.132}$$

and so, $\Delta = -(1/2)\Pi^D$.

Coulomb, or dry friction, is much more difficult to model. If δ_i^C denotes either the dissipative torque produced by Coulomb friction at a revolute or the dissipative force produced by Coulomb friction at a prismatic joint, and $\dot{\theta}_i$ the associated joint rate, then, the simplest model for the resulting generalized Coulomb-friction force is

$$\delta_i^C = -\tau_i^C\,\text{sgn}(\dot{\theta}_i) \tag{7.133}$$

where sgn$(\cdot)$ denotes the *signum function*, which is defined as $+1$ or -1, depending on whether its argument is positive or negative, and τ_i^C is a positive constant representing a torque for revolute joints or a force for prismatic joints. The numerical

value of this constant is to be determined experimentally. The foregoing model leads to a simple expression for the associated dissipation function, namely,

$$\Delta_i^C = \tau_i^C |\dot{\theta}_i| \tag{7.134}$$

The Coulomb dissipation function for the overall manipulator is, then,

$$\Delta^C = \sum_1^n \tau_i^C |\dot{\theta}_i| \tag{7.135}$$

The foregoing simplified model of Coulomb friction forces is applicable when the relative speed between the two surfaces in contact is high. However, at low relative speed, that model becomes inaccurate. In robotics applications, where typical end-effector maximum speeds are of the order of 1 m/s, relative speeds are obviously low, and hence, a more accurate model should be introduced. Such a model should account for the empirical observation that Coulomb frictional forces are higher at low relative speeds and become constant at very high relative speeds. A model taking this fact into account has the form

$$\delta_i^C = -(\tau_i^C + \epsilon_i e^{-\gamma_i |\dot{\theta}_i|}) \text{sgn}(\dot{\theta}_i) \tag{7.136}$$

where γ_i, and ϵ_i are constants associated with the ith joint and are to be determined experimentally. The foregoing expression readily leads to the dissipation function associated with the same joint, namely,

$$\Delta_i^C = \tau_i^C |\dot{\theta}_i| + \frac{\epsilon_i}{\gamma_i}(1 - e^{-\gamma_i |\dot{\theta}_i|}) \tag{7.137}$$

and hence, the Coulomb dissipation function of the overall manipulator becomes

$$\Delta^C = \sum_1^n \left[\tau_i^C |\dot{\theta}_i| + \frac{\epsilon_i}{\gamma_i}(1 - e^{-\gamma_i |\dot{\theta}_i|}) \right] \tag{7.138}$$

Dissipation functions are very useful. On the one hand, they allow us to obtain associated generalized frictional forces when these are difficult, if not impossible, to express in formula form. On the other hand, since dissipation functions represent nonrecoverable forms of power, their integrals over time yield the dissipated energy. Moreover, the energy dissipated into unrecoverable heat can be estimated from an energy balance, and hence, the parameters associated with that dissipation function can be estimated with suitable identification techniques, once a suitable model for a dissipation function is available. Furthermore, the said parameters appear in the generalized frictional forces as well. For this reason, knowing these parameters is essential for the modeling of the corresponding generalized frictional forces.

7.9 Exercises

7.1 Show that:

(a) the $6n$-dimensional manipulator twist lies in the null space of the $6n \times 6n$ manipulator angular velocity matrix $\mathbf{W}$;

(b) the time-derivative of the $6n \times 6n$ manipulator mass matrix $\mathbf{M}$ is given by

$$\dot{\mathbf{M}} = \mathbf{WM} - \mathbf{MW}$$

(c)

$$\frac{d\boldsymbol{\mu}}{dt} = \mathbf{M}\dot{\mathbf{t}} + \mathbf{WMt}$$

thereby verifying Eq. (7.15).

7.2 In order to gain insight into the meaning of vector $\boldsymbol{\gamma}$, as defined in Example 7.3.1, we define a similar vector $\boldsymbol{\eta}$ as

$$\boldsymbol{\eta} = \frac{\partial(\mathbf{I}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}}$$

Compute $\boldsymbol{\eta}$ for that example and compare the result with $\boldsymbol{\gamma}$.

7.3 The decoupled robot of Fig. 4.23 is to undergo a maneuver, at the posture displayed in that figure, that involves the velocity and acceleration specifications given below, in base coordinates:

$$\dot{\mathbf{c}} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \text{ m/s}, \quad \boldsymbol{\omega} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ rad/s},$$

$$\ddot{\mathbf{c}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ m/s}^2, \quad \dot{\boldsymbol{\omega}} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \text{ rad/s}^2$$

Compute the joint torques required to drive the robot through the desired maneuver, if the robot is known to have the inertial parameters given below:

$$m_1 = 10.521,\ m_2 = 15.781,\ m_3 = 8.767,$$
$$m_4 = 1.052,\ m_5 = 1.052,\ m_6 = 0.351$$

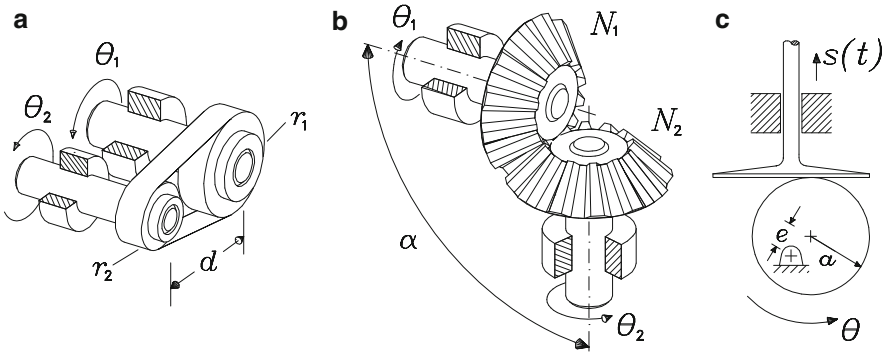

Fig. 7.9 Three different pairs of coupled bodies

$$\boldsymbol{\rho}_1 = \begin{bmatrix} 0 \\ -0.054 \\ 0 \end{bmatrix}, \quad \boldsymbol{\rho}_2 = \begin{bmatrix} 0.140 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\rho}_3 = \begin{bmatrix} 0 \\ -0.197 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\rho}_4 = \begin{bmatrix} 0 \\ 0 \\ -0.057 \end{bmatrix}, \quad \boldsymbol{\rho}_5 = \begin{bmatrix} 0 \\ -0.007 \\ 0 \end{bmatrix}, \quad \boldsymbol{\rho}_6 = \begin{bmatrix} 0 \\ 0 \\ -0.019 \end{bmatrix}$$

$$\mathbf{I}_1 = \text{diag}\begin{bmatrix} 1.6120 & 0.5091 & 1.6120 \end{bmatrix}$$

$$\mathbf{I}_2 = \text{diag}\begin{bmatrix} 0.4898 & 8.0783 & 8.2672 \end{bmatrix}$$

$$\mathbf{I}_3 = \text{diag}\begin{bmatrix} 3.3768 & 0.3009 & 3.3768 \end{bmatrix}$$

$$\mathbf{I}_4 = \text{diag}\begin{bmatrix} 0.1810 & 0.1810 & 0.1273 \end{bmatrix}$$

$$\mathbf{I}_5 = \text{diag}\begin{bmatrix} 0.0735 & 0.0735 & 0.1273 \end{bmatrix}$$

$$\mathbf{I}_6 = \text{diag}\begin{bmatrix} 0.0071 & 0.0071 & 0.0141 \end{bmatrix}$$

where m_i, $\boldsymbol{\rho}_i$, and $\mathbf{I}_i$ are given in units of kg, m and kg m^2, respectively, with the position vectors of the centers of mass and the moment-of-inertia matrices given in link-fixed coordinates. *Note: Assume that Z_7 is perpendicular to Z_5 and Z_6, with O_7 located at the OP of the EE.*

7.4 Derive *homogeneous, linear constraint equations* on the twists of the pairs of coupled bodies appearing in Fig. 7.9, namely,

(a) two rigid pulleys coupled by an inextensible belt, under no slip;
(b) the bevel pinion-and-gear train with axes intersecting at an arbitrary angle α;

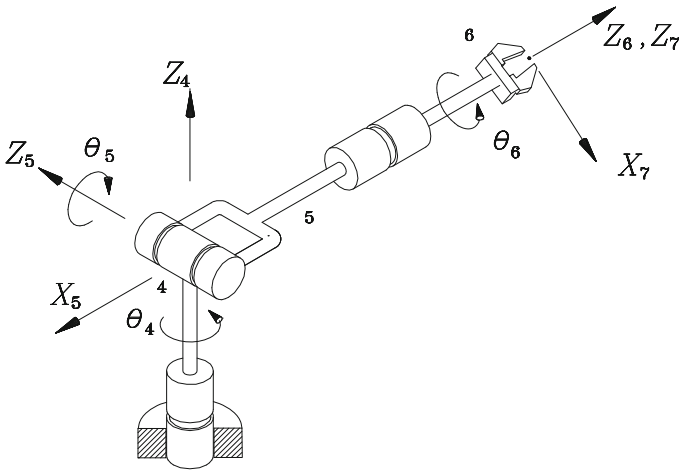

Fig. 7.10 A three-revolute spherical wrist

(c) the cam-and-follower mechanism whose cam disk is an eccentric circular disk.

Notice that the constraint equations sought should have the form:

$$\mathbf{A}\mathbf{t}_1 + \mathbf{B}\mathbf{t}_2 = \mathbf{0}$$

with $\mathbf{t}_1$ and $\mathbf{t}_2$ denoting the twists of bodies 1 and 2, respectively.

7.5 Use the expressions derived in Example 7.6.2 with the aid of the natural orthogonal complement, as pertaining to the planar manipulator of Fig. 7.1, to obtain an expression for the time-derivative of the inertia matrix of this manipulator. Compare the expression thus obtained with that derived in Example 7.3.1, and verify that the difference $\dot{\mathbf{I}} - 2\mathbf{C}$ is skew-symmetric—see Exercise 12.2—where $\mathbf{C}$ is the matrix coefficient of the Coriolis and centrifugal terms.

7.6 A three-revolute spherical wrist with an orthogonal architecture, i.e., with neighboring joint axes at right angles, is shown in Fig. 7.10. Assume that the moments of inertia of its three links with respect to O, the point of concurrency of the three axes, are given by constant diagonal matrices, in link-fixed coordinates, as

$$\mathbf{I}_4 = \text{diag}(J_1, J_2, J_3)$$

$$\mathbf{I}_5 = \text{diag}(K_1, K_2, K_3)$$

$$\mathbf{I}_6 = \text{diag}(L_1, L_2, L_3)$$

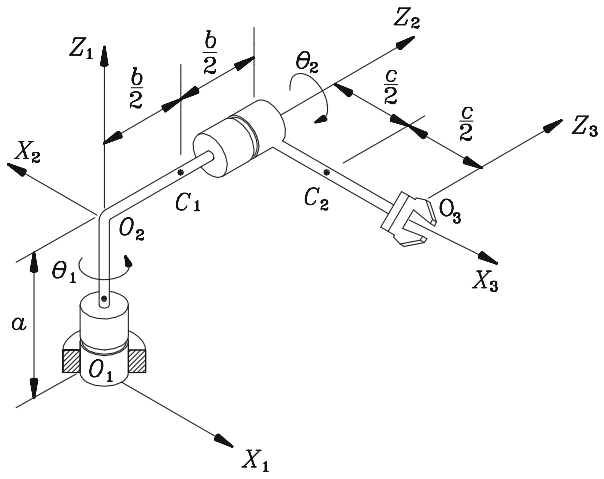

Fig. 7.11 A two-revolute pointing manipulator

while the potential energy of the wrist is

$$V = -m_6 g a \cos\theta_5$$

Moreover, the motors produce torques τ_4, τ_5, and τ_6, respectively, whereas the power losses can be accounted for via a dissipation function of the form

$$\Delta = \sum_4^6 \left(\frac{1}{2} b_i \dot{\theta}_i^2 + \tau_i^C |\dot{\theta}_i| \right)$$

where b_i and τ_i^C, for $i = 4, 5, 6$, are constants.

(a) Derive an expression for the matrix of generalized inertia of the wrist.
(b) Derive an expression for the term of Coriolis and centrifugal forces.
(c) Derive the dynamical model of the wrist. *Hint: The kinetic energy T of a rigid body rotating about a fixed point O with angular velocity $\boldsymbol{\omega}$ can be written as $T = \frac{1}{2}\boldsymbol{\omega}^T \mathbf{I}_O \boldsymbol{\omega}$, where $\mathbf{I}_O$ is the moment-of-inertia matrix of the body with respect to O.*

7.7 Shown in Fig. 7.11 is a two-revolute *pointing manipulator*. The centroidal inertia matrices of the links are denoted by $\mathbf{I}_1$ and $\mathbf{I}_2$. These are given, in link-fixed coordinates, by:

$$\mathbf{I}_1 = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{12} & I_{22} & I_{23} \\ I_{13} & I_{23} & I_{33} \end{bmatrix}, \quad \mathbf{I}_2 = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{12} & J_{22} & J_{23} \\ J_{13} & J_{23} & J_{33} \end{bmatrix}$$

Moreover, the centers of mass of the links are denoted by C_1 and C_2, respectively, and are shown in the same figure, the masses being denoted by m_1 and m_2.

(a) Determine the kinetic energy of the manipulator as a quadratic function of $\dot{\theta}_1$ and $\dot{\theta}_2$.
(b) Determine the 2×2 matrix of generalized inertia.
(c) Find an expression for the time-rate of change of the matrix of generalized inertia by straightforward differentiation of the expression found in item (b).
(d) Repeat item (c), but now by differentiation of the three factors of $\mathbf{I}$, as given in

$$\mathbf{I} = \mathbf{T}^T \mathbf{M} \mathbf{T}$$

7.8 The twist $\mathbf{t}_i$ of the ith link of an n-dof serial manipulator can be expressed as

$$\mathbf{t}_i = \mathbf{T}_i \dot{\boldsymbol{\theta}}$$

where $\mathbf{T}_i$ is a $6 \times n$ link-twist-shaping matrix and $\dot{\boldsymbol{\theta}}$ is the n-dimensional vector of actuated joint rates. Moreover, let $\mathbf{M}_i$ and $\mathbf{W}_i$ be the 6×6 matrices defined in Sect. 7.3. Show that if the link is constrained to undergo planar motion, then the product $\mathbf{T}_i^T \mathbf{W}_i \mathbf{M}_i \mathbf{T}_i$ vanishes.

7.9 Devise a recursive algorithm to compute the joint torques required to balance a wrench $\mathbf{w}$ acting at the EE of a six-revolute manipulator of arbitrary architecture. Then, derive the number of floating-point operations (multiplications and additions) required to compute these torques, and compare your result with the number of floating point operations required to compute the same by matrix–times–vector multiplications, using the transpose Jacobian.

7.10 Establish the computational cost incurred in computing the term of Coriolis and centrifugal forces of an n-revolute serial manipulator, when the Newton–Euler algorithm is used for this purpose.

7.11 Shown in Fig. 7.12 is an RRP manipulator, whose DH parameters are displayed in Table 7.5. The masses of its three moving links are denoted by m_1, m_2, and m_3, and the center of mass of each of links 1 and 2 coincides with O_1, while the center of mass of link 3 is located at P. Moreover, the centroidal moments of inertia of these links are, in link-fixed coordinates,

$$[\mathbf{I}_1]_2 = A\mathbf{1}, \quad [\mathbf{I}_2]_3 = B\mathbf{1}, \quad [\mathbf{I}_3]_4 = C\mathbf{1}$$

where $\mathbf{1}$ denotes the 3×3 identity matrix.

(a) Derive the Euler–Lagrange equations of the manipulator under the assumption that gravity acts in the direction of X_1.
(b) Find the generalized inertia matrix of the manipulator.

7.12 A link is said to be *inertially isotropic* if its three principal moments of inertia are identical.

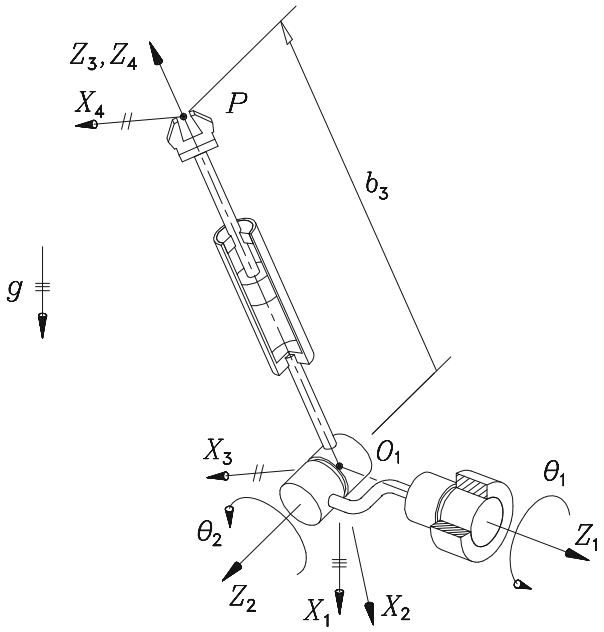

Fig. 7.12 An RRP spatial manipulator

Table 7.5 DH parameters of the RRP manipulator

i	a_i	b_i	α_i
1	0	0	90°
2	0	0	90°
3	0	b_3	0°

(a) Show that *any* direction is a principal axis of inertia of an inertially isotropic link.
(b) Explore the advantages of a manipulator with inertially isotropic links with regard to its real-time control, i.e., find the savings in floating-point operations required to compute the recursive Newton–Euler algorithm of such a manipulator.

7.13 Devise an algorithm similar to Algorithm 7.6.1, but applicable to planar manipulators, and determine the computational costs involved in its implementation.

7.14 Write a piece of code to evaluate numerically the inertia matrix of an n-axis manipulator and test it with the manipulator of Example 7.6.2. For this purpose, assume that $I = ma^2$.

7.15 With reference to the mathematical model of a n-dof serial manipulator of Eq. (7.60), show that the matrix difference $\mathbf{\Delta} \equiv \dot{\mathbf{I}} - 2\mathbf{C}$ is skew-symmetric. This result is important, because it leads to the stabilization of the manipulator with the aid of a simple proportional-derivative (PD) controller (Spong et al. 2006).