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 Chapter 4, we indicate the units of multiplications and additions with M and A, respectively.

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 & 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 very simple components in $\mathcal{F}_i$, namely, $[0,\ 0,\ 1]^T$, this will be regarded as a vector of the $(i-1)$st link. 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 Section 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 kinematic relations above, 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

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

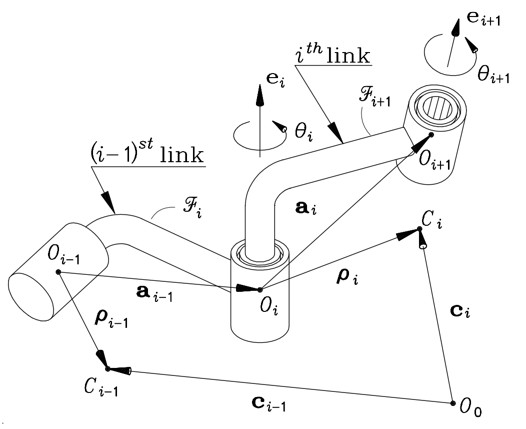

Figure 7.2: A revolute joint

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,

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

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 mass center 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 mass centers thus observe the relationships

(i) if the ith joint is R,

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

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

(ii) if the ith joint is P,

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

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

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 & d) with respect to time, we derive the corresponding relations between the velocities and accelerations of the mass centers of links $i-1$ and i, namely,

(i) if the ith joint is R,

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

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

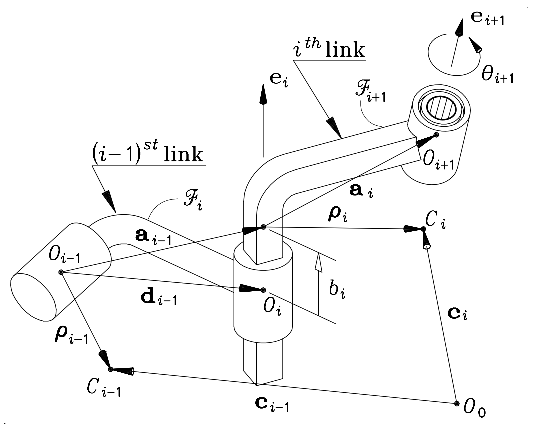

Figure 7.3: A prismatic joint

(*ii*) if the ith joint is P,

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

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

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

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

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

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

for $i = 1, 2, \ldots, n$, where $\dot{\mathbf{c}}_0$ and $\ddot{\mathbf{c}}_0$ are the velocity and acceleration of the mass center of 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) to (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 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 on 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 $\{\mathbf{Q}_i\}_0^{n-1}$, $\mathbf{c}_0$, $\boldsymbol{\omega}_0$, $\dot{\mathbf{c}}_0$, $\dot{\boldsymbol{\omega}}_0$, $\ddot{\mathbf{c}}_0$, $\{\boldsymbol{\rho}_i\}_1^n$, $\{\boldsymbol{\delta}_i\}_0^{n-1}$
For i = 1 to n step 1 do
      update $\mathbf{Q}_i$
      if R then
        $\mathbf{c}_i \leftarrow \mathbf{Q}_i^T(\mathbf{c}_{i-1} + \boldsymbol{\delta}_{i-1}) + \boldsymbol{\rho}_i$
        $\boldsymbol{\omega}_i \leftarrow \mathbf{Q}_i^T(\boldsymbol{\omega}_{i-1} + \dot{\theta}_i\mathbf{e}_i)$
        $\mathbf{u}_{i-1} \leftarrow \boldsymbol{\omega}_{i-1} \times \boldsymbol{\delta}_{i-1}$
        $\mathbf{v}_i \leftarrow \boldsymbol{\omega}_i \times \boldsymbol{\rho}_i$
        $\dot{\mathbf{c}}_i \leftarrow \mathbf{Q}_i^T(\dot{\mathbf{c}}_{i-1} + \mathbf{u}_{i-1}) + \mathbf{v}_i$
        $\dot{\boldsymbol{\omega}}_i \leftarrow \mathbf{Q}_i^T(\dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \dot{\theta}_i\mathbf{e}_i + \ddot{\theta}_i\mathbf{e}_i)$
        $\ddot{\mathbf{c}}_i \leftarrow \mathbf{Q}_i^T(\ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{\delta}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{u}_{i-1})$
                 $+\dot{\boldsymbol{\omega}}_i \times \boldsymbol{\rho}_i + \boldsymbol{\omega}_i \times \mathbf{v}_i$
      else
        $\mathbf{u}_i \leftarrow \mathbf{Q}_i^T\boldsymbol{\delta}_{i-1} + \boldsymbol{\rho}_i + b_i\mathbf{e}_i$
        $\mathbf{c}_i \leftarrow \mathbf{Q}_i^T\mathbf{c}_{i-1} + \mathbf{u}_i$
        $\boldsymbol{\omega}_i \leftarrow \mathbf{Q}_i^T\boldsymbol{\omega}_{i-1}$
        $\mathbf{v}_i \leftarrow \boldsymbol{\omega}_i \times \mathbf{u}_i$
        $\mathbf{w}_i \leftarrow \dot{b}_i\mathbf{e}_i$
        $\dot{\mathbf{c}}_i \leftarrow \mathbf{Q}_i^T\dot{\mathbf{c}}_{i-1} + \mathbf{v}_i + \mathbf{w}_i$
        $\dot{\boldsymbol{\omega}}_i \leftarrow \mathbf{Q}_i^T\dot{\boldsymbol{\omega}}_{i-1}$
        $\ddot{\mathbf{c}}_i \leftarrow \mathbf{Q}_i^T\ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{u}_i + \boldsymbol{\omega}_i \times (\mathbf{v}_i + \mathbf{w}_i + \mathbf{w}_i) + \ddot{b}_i\mathbf{e}_i$
      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:

(i) 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 $5A$
$\dot{\mathbf{c}}_i$ requires $20M$ and $16A$
$\dot{\boldsymbol{\omega}}_i$ requires $10M$ and $7A$
$\ddot{\mathbf{c}}_i$ requires $32M$ and $28A$

(ii) 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 $11A$

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	$82n$	$66n$

$\dot{\boldsymbol{\omega}}_i$ 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

Moreover, a free-body diagram of the end-effector, or nth link, appears in Fig. 7.5. Note that this link is acted upon by a nonworking 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 Section 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

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

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.

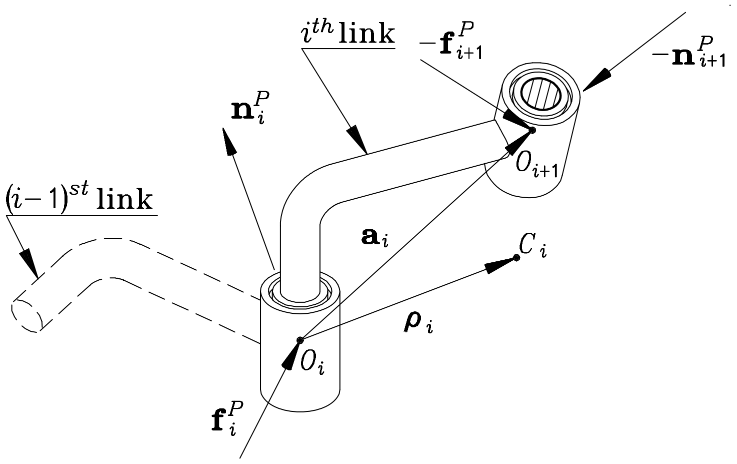

Figure 7.4: Free-body diagram of the ith link

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 mass center of 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 & c).

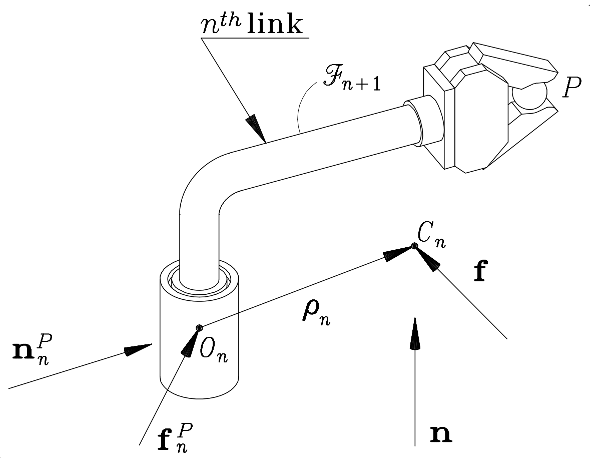

Figure 7.5: Free-body diagram of the end-effector

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

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

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 & 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 & Patel (1991)	tensor	$93n - 69$	$81n - 65$
Li & 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 & 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 & Orin (1982)	N-E	800	595
Hollerbach and Sahar (1983)	N-E	688	558
Kane & Levinson (1983)	Kane	646	394
Khalil et al. (1986)	N-E	538	478
Angeles et al. (1989)	Kane	521	435
Balafoutis & Patel (1991)	tensor	489	420
Li & Sankar (1992)	E-L	459	390

7.5 The Natural Orthogonal Complement in Robot Dynamics

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

$$\mathbf{x} \equiv \begin{bmatrix} \boldsymbol{\theta}^T & \dot{\boldsymbol{\theta}}^T \end{bmatrix}^T \tag{7.46}$$

The n generalized coordinates, then, define the configuration of the system, while their time-derivatives determine its generalized momentum, an item defined in eq.(7.19d). Hence, knowing $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$, we can predict the future values of these variables 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, mass-center 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 Subsection 7.5.2.

We now recall the Newton-Euler equations of the ith body in 6-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 & 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 on 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 mass center 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 *uncoupled* Newton-Euler equations of the overall manipulator. The following step of this derivation consists in representing the coupling between every two consecutive links 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 Subsection 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 singular. In fact, the dimension of the nullspace of $\mathbf{K}$, termed its *nullity*, is exactly equal to n, the degree of freedom of the manipulator. 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 possible 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 inner product of $\mathbf{w}^C$ and $\mathbf{t}$, as stated by eq.(7.51), vanishes, and hence, $\mathbf{t}$ lies in the nullspace 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 Subsection 7.5.1 below. Now, upon substitution of eq.(7.54) into eq.(7.50), we obtain

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

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.,

$$\mathbf{K}\mathbf{T} = \mathbf{O} \tag{7.55b}$$

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,

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

Furthermore, the uncoupled equations, eqs.(7.48), are multiplied from the left by $\mathbf{T}^T$, thereby eliminating $\mathbf{w}^C$ from those equations and reducing these to a system of only n independent equations, free of nonworking constraint wrenches. These are nothing but the Euler-Lagrange equations of the manipulator, namely,

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

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

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