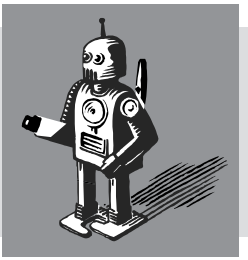
9

Recursive Robot Dynamics*



This chapter introduces one of the orthogonal-complement-based methodologies for the automatic generation of dynamic equations of motion and associated algorithms, namely, the inverse and forward dynamics. As mentioned in Chapter 8, inverse dynamics is essential for control of robot manipulators, whereas forward dynamics is required for computer simulation and real-time feedback control. To solve the inverse or forward dynamics problems of a complex robotic system, a set of dynamic equations of motion or the dynamic model of the robot under study is needed, which were derived in Chapter 8 using Euler–Lagrange (EL) and Newton–Euler (NE) formulations. The resultant equations of motion are expressed generally in the form of Ordinary Differential Equations (ODE). The same set can be obtained using alternate approaches, e.g., orthogonal complements of the velocity constraints (Huston and Passerello, 1974), Kane's equations (Kane and Levinson, 1983), and others. Amongst these, the former approach has been adopted by many researchers for the automatic generation of the equations of motion for complex mechanical systems like the robots studied in this book. One such complement is the *Natural Orthogonal Complement* (NOC), which was originally proposed by Angeles and Lee (1988) for serial robots, but generalized later by Saha and Angeles (1991) to take into account the nonholonomic constraints of wheeled mobile robots. The NOC was eventually decoupled by the author of this book (Saha 1997, 1999, 2003) which was named as the Decoupled NOC (DeNOC). The decoupling of the NOC or the use of DeNOC has the following advantages:

Why Recursive

Recursive algorithms, where the computations are done based on some previous calculations, provide computationally efficient algorithms.

- It allows one to obtain recursive order (n)—n being the number of links in the serial-chain robot—inverse and forward dynamics algorithms. The recursive forward dynamics algorithm was not possible with the original form of the NOC. Also, the recursive NE algorithm presented in Section 8.4 is meant only for the inverse dynamics.
- Each scalar element of the matrices and vectors associated with the equations of motion of the robot can be written analytically, which allows one to provide

* This chapter requires in-depth understanding of Chapter 8. Hence, it is advised to offer to final year undergraduate students, postgraduate students at MTech, MS, and PhD levels.

many physical interpretation, e.g., articulated body inertia, etc., and helps a programmer debug the computer algorithms.

- Since the methodology is built upon basic mechanics and linear algebra theories, the concept can be easily understood even by undergraduate students.

The DeNOC-based modeling has been successfully applied to (i) serial manipulators with fixed-base, as used in industrial robots (Saha, 1997; 1999; 2003); (ii) serial robots with free-base, i.e., a configuration for free-floating space robots (Saha, 1996); (iii) closed-loop parallel Stewart platform-type robots (Saha and Schiehlen, 2001; Khan et al., 2005), and Hexapod machine tools (Koteswara Rao et al., 2006); (iv) multi-closed-loop general mechanical systems (Chaudhary and Saha, 2006); (v) serial flexible-link systems (Mohan and Saha, 2007a), and (vi) general tree-type systems (Shah and Saha, 2013). As emphasized in Mohan and Saha (2007b), and Shah et al. (2013), the DeNOC based formulation provides efficient and numerically stable algorithms.

9.1 DYNAMIC MODELING

In this section, dynamic equations of motion of an *n*-link *n* degree-of-freedom (DOF) serial robot, as shown in Fig. 9.1(a), are derived using the Decoupled Natural Orthogonal Compliment (DeNOC) matrices. First, the uncoupled Newton-Euler (NE) equations of motion, as introduced in Section 8.3, are written for all *n* links in a compact form. Next, the constraints between the links due to the joints, e.g., revolute, prismatic, etc., are expressed mathematically, which brings the DeNOC matrix into picture. The DeNOC relates the Cartesian velocities of all the links with the joint rates or velocities. Finally, the pre-multiplication of the DeNOC matrices with the uncoupled NE equations yields a set of independent equations of motion, which are nothing but the Euler–Lagrange (EL) equations of motion presented in Section 8.2. Hence, the EL equations are derived via DeNOC matrices without resorting to the complex partial derivatives given by Eq. (8.24).

Modeling

Modeling here means a way to be able to understand the behavior of a robot even without having a real one.

9.1.1 Uncoupled Newton–Euler Equations

For the *n*-link *n*-DOF open-loop serial-chain robot manipulator, Fig. 9.1(a), if m_i is the mass of the i^{th} link and $\mathbf{I}_i$ denotes the 3×3 inertia tensor of the i^{th} link about its mass center C_i, as indicated in Fig. 9.1(b), the Newton–Euler (NE) equations of motion for the i^{th} link can be derived from its free-body diagram, and written as

$$\text{Euler's Equation:} \qquad \mathbf{I}_i\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i = \mathbf{n}_i \tag{9.1a}$$

$$\text{Newton's Equation:} \qquad m_i\ddot{\mathbf{c}}_i = \mathbf{f}_\mathbf{i} \tag{9.1b}$$

where $\boldsymbol{\omega}_i$ and $\dot{\boldsymbol{\omega}}_i$ are the 3-dimensional vectors of angular velocity and acceleration for the i^{th} link, respectively, whereas $\ddot{\mathbf{c}}_i$ is the 3-dimensional vector of acceleration of the mass center C_i. Moreover, $\mathbf{n}_i$ and $\mathbf{f}_\mathbf{i}$ are the 3-dimensional vectors of the resultant moment about C_i and resultant force at C_i, respectively. Note here that no reference to the coordinate frame is made to express the vectors and matrices, as they can be

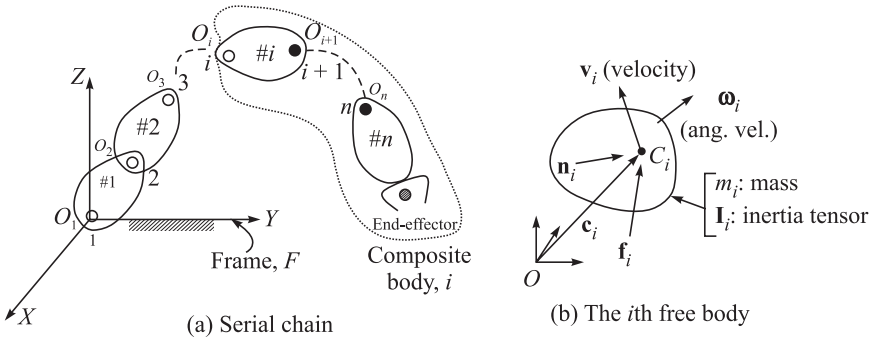


Fig. 9.1 A serial robot manipulator

represented in any frame of the analyst's choice. Typically, they are expressed in the frame attached to the i^{th} link, i.e., frame i+1. However, during the derivations of the equations of motion in the subsequent sections, they will be avoided. Combining Eqs. (9.1a-b), the six scalar uncoupled NE equations are written in a compact form as

$$\mathbf{M}_i\dot{\mathbf{t}}_i + \mathbf{W}_i\mathbf{M}_i\mathbf{t}_i = \mathbf{w}_i \tag{9.2a}$$

where the 6 × 6 mass matrix $\mathbf{M}_i$, and the the 6 × 6 angular velocity matrix $\mathbf{W}_i$, for the i^{th} link are given by

$$\mathbf{M}_i \equiv \begin{bmatrix} \mathbf{I}_i & \mathbf{O} \\ \mathbf{O} & m_i\mathbf{1} \end{bmatrix}; \text{ and } \mathbf{W}_i \equiv \begin{bmatrix} \boldsymbol{\omega}_i \times \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \tag{9.2b}$$

in which $\boldsymbol{\omega}_i \times \mathbf{1}$ is the 3 × 3 cross-product tensor associated with the angular velocity vector $\boldsymbol{\omega}_i$, which is defined as $(\boldsymbol{\omega}_i \times \mathbf{1})\mathbf{x} \equiv \boldsymbol{\omega}_i \times \mathbf{x}$, for any three dimensional Cartesian vector $\mathbf{x}$. Moreover, $\mathbf{1}$ and $\mathbf{O}$ are the 3 × 3 identity and zero matrices, respectively. Furthermore, the 6-dimensional vectors, twist $\mathbf{t}_i$ and wrench $\mathbf{w}_i$, are as follows:

$$\mathbf{t}_i \equiv \begin{bmatrix} \boldsymbol{\omega}_i \\ \dot{\mathbf{c}}_i \end{bmatrix} \text{ and } \mathbf{w}_i \equiv \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \tag{9.2c}$$

where, in contrast to the definition of twist given in Eq. (6.93), the linear velocity of the mass center of the i^{th} link C_i is considered. In Eq. (9.2a), vector $\dot{\mathbf{t}}_i$ is the time derivative of the twist vector $\mathbf{t}_i$ defined in Eq. (9.2c). Equation (9.2a) is now written for all n links, i.e., for i = 1, …, n, as

$$\mathbf{M}\dot{\mathbf{t}} + \mathbf{W}\mathbf{M}\mathbf{t} = \mathbf{w} \tag{9.3a}$$

where $\mathbf{M}$ and $\mathbf{W}$ are the 6n × 6n generalized mass matrix, and the generalized matrix of the angular velocities, respectively. They are given by

$$\mathbf{M} \equiv \text{diag.}[\mathbf{M}_1, \ldots, \mathbf{M}_n], \text{ and } \mathbf{W} \equiv \text{diag.}[\mathbf{W}_1, \ldots, \mathbf{W}_n] \tag{9.3b}$$

Also, the 6n-dimensional vectors of generalized twist and wrench are defined as

$$\mathbf{t} \equiv [\mathbf{t}_1{}^T, \ldots, \mathbf{t}_n{}^T]^T; \text{ and } \mathbf{w} \equiv [\mathbf{w}_1{}^T, \ldots, \mathbf{w}_n{}^T]^T \tag{9.3c}$$

Equations (9.3a-c) represent the 6n uncoupled NE equations of motion of the n-links in the serial robot manipulator under study.

9.1.2 Kinematic Constraints

Links of the robot manipulator, Fig. 9.1(a), are coupled by kinematic pairs or joints which are either revolute or prismatic. From the rigid-body motion of the two bodies or links, namely, #*i* and #*j*, as shown in Fig. 9.2, the angular and linear velocities of the i^{th} link, i.e., the twist of the i^{th} body defined in Eq. (9.2c), can be derived from the velocities of the j^{th} link or #*j*, and the joint motion of the i^{th} joint. They are derived as follows:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_j + \mathbf{e}_i\dot{\theta}_i \tag{9.4a}$$

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_j + \boldsymbol{\omega}_j \times \mathbf{r}_j + \boldsymbol{\omega}_i \times \mathbf{d}_i \tag{9.4b}$$

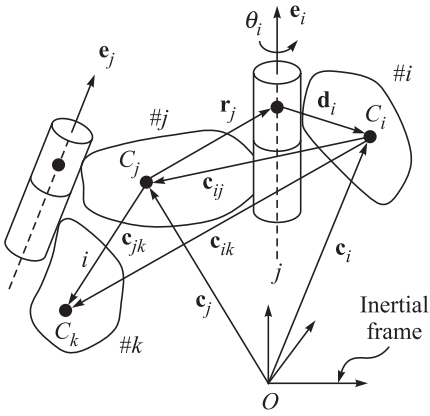


Fig. 9.2 Three coupled bodies

The above six scalar velocity constraint equations can be written in compact form as

$$\mathbf{t}_i = \mathbf{B}_{ij}\mathbf{t}_j + \mathbf{p}_i\dot{\theta}_i \tag{9.4c}$$

where θ_i is the joint displacement, angular for a revolute joint and linear for a prismatic joint. Accordingly $\dot{\theta}_i$ is the joint rate. Moreover, the 6×6 matrix $\mathbf{B}_{ij}$ and the 6-dimensional vector $\mathbf{p}_i$ are functions of the positions of the mass centres of the two successive bodies, i.e., C_i and C_j of Fig. 9.2, and the axis of the joint joining them, namely, $\mathbf{e}_i$. They are defined as

$$\mathbf{B}_{ij} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{c}_{ij} \times \mathbf{1} & \mathbf{1} \end{bmatrix} \text{ and } \mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{d}_i \end{bmatrix} \tag{9.4d}$$

$\mathbf{c}_{ij} \times \mathbf{1}$ being the cross-product tensor associated with vector $\mathbf{c}_{ij}$ shown in Fig. 9.2, which is defined similar to $\boldsymbol{\omega}_i \times \mathbf{1}$ of Eq. (9.2b), i.e., $(\mathbf{c}_{ij} \times \mathbf{1})\mathbf{x} = \mathbf{c}_{ij} \times \mathbf{x}$, for any arbitrary 3-dimensional Cartesian vector $\mathbf{x}$. The vector $\mathbf{c}_{ij}$ is given by $\mathbf{c}_{ij} = -\mathbf{d}_i - \mathbf{r}_j$. It is interesting to note here that the matrix $\mathbf{B}_{ij}$ and the vector $\mathbf{p}_i$ have the following interpretations:

- For two rigidly connected moving links, #*i* and #*j*, $\mathbf{B}_{ij}$, propagates the twist of #*j* to #*i*. Hence, matrix $\mathbf{B}_{ij}$ is termed here as the *twist propagation matrix*, which has the following properties:

$$\mathbf{B}_{ij}\mathbf{B}_{jk} = \mathbf{B}_{ik};\ \mathbf{B}_{ii} = \mathbf{1};\ \text{and } \mathbf{B}_{ij}^{-1} = \mathbf{B}_{ji} \tag{9.5a}$$

- Vector $\mathbf{p}_i$ on the other hand takes into account the motion of the i^{th} joint. Hence, $\mathbf{p}_i$ is termed as the *joint-motion propagation vector*, which is dependent on the type of joint. For example, the expression of $\mathbf{p}_i$ in Eq. (9.4d) is for a revolute joint shown in Fig. 9.2, whereas for a prismatic joint, vector $\mathbf{p}_i$ is given by

$$\mathbf{p}_i \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \end{bmatrix} \text{: For a prismatic joint} \tag{9.5b}$$

  where $\mathbf{e}_i$ is the unit vector parallel to the axis of linear motion. Correspondingly, $\dot{\theta}_i$ of Eq. (9.4c) would mean the linear joint-rate. Other joints are not treated here because any other joint, e.g., a spherical or a cylindrical, can be treated as

combination of three revolute, or revolute and prismatic pairs, respectively. For $i = 1, \ldots, n$, Eq.(9.4c) is put in a compact form for all the n joints as

$$\mathbf{t} = \mathbf{N}\dot{\boldsymbol{\theta}}, \text{ where } \mathbf{N} \equiv \mathbf{N}_l\, \mathbf{N}_d \tag{9.6a}$$

where $\mathbf{t}$ is the $6n$-dimensional generalized twist defined in Eq. (9.3c). In Eq. (9.6a), it is expressed as a linear transformation of the n-dimensional joint-rate vector $\dot{\boldsymbol{\theta}}$. The $6n \times 6n$ matrix $\mathbf{N}_l$, the $6n \times n$ matrix $\mathbf{N}_d$, and the n-dimensional vector $\dot{\boldsymbol{\theta}}$, are defined as follows:

$$\mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{B}_{21} & \mathbf{1} & \cdots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & \mathbf{1} \end{bmatrix}; \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{p}_n \end{bmatrix}; \text{ and } \dot{\boldsymbol{\theta}} \equiv \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} \tag{9.6b}$$

The $6n \times n$ matrix $\mathbf{N}$ in Eq. (9.6a) is nothing but the Natural Orthogonal Complement (NOC) matrix of the velocity constraints (Angeles and Lee, 1988), and its decoupled form $\mathbf{N}_l$ and $\mathbf{N}_d$, are referred as the Decoupled NOC (DeNOC) matrices (Saha, 1997; 1999). The expressions of the DeNOC matrices allow one to develop recursive inverse and forward dynamics algorithms required in control and simulation of robot manipulators, respectively.

Note however the difference in the expression of $\mathbf{N}_l$ and $\mathbf{N}_d$ in Eq. (9.6b) compared to those in Chapter 6 for the derivation of Jacobian, namely, Eq. (6.98b) and (6.101). This is mainly due to the choice of a point on the rigid link to specify its linear velocity. Whereas in Chapter 6, the point O_i at which the links #(i–1) and #i are coupled was chosen to define the linear velocity of #i, as explained after Eq. (6.93), the same is defined in this chapter with respect to C_i, i.e., the mass center of #i which is given after Eq. (9.4c). To point out these differences, two different notation for the twist-propagation matrices denoted with $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ defined in Eqs. (6.94b) and (9.4d), respectively, are introduced. Such differences will occur even for a third choice of a point to define the linear velocity of #i. In general, if $\mathbf{t}_i^O$ and $\mathbf{t}_i^C$ are used to define the twists with respect to point O_i (Chapter 6) and point C_i (this chapter), then one can draw the following correlations:

$$\mathbf{t}_i^C = (\mathbf{B}_{i,i-1} + \mathbf{P}_{i-1}^d)\mathbf{t}_{i-1}^O + (\mathbf{p}_i^O + \mathbf{p}_i^d)\dot{\theta}_i \tag{9.7a}$$

$$\mathbf{t}_i^O = (\mathbf{A}_{i,i-1} - \mathbf{P}_{i-1}^d)\mathbf{t}_{i-1}^C + (\mathbf{p}_i^C - \mathbf{p}_i^d)\dot{\theta}_i \tag{9.7b}$$

where $\mathbf{p}_i^O$ and $\mathbf{p}_i^C$ are the joint-motion propagation vectors, i.e., $\mathbf{p}_i$'s of Eqs. (6.94b) and (9.4d), respectively, whereas the matrix $\mathbf{P}^d{}_{i,i-1}$ and vector $\mathbf{p}_i^d$ are defined as

$$\mathbf{P}^d{}_{i-1} \equiv \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ -\mathbf{d}_{i-1} \times \mathbf{1} & \mathbf{O} \end{bmatrix} \text{ and } \mathbf{p}_i^d \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \times \mathbf{d}_i \end{bmatrix} \tag{9.7c}$$

Even though both the definitions of twists were used in the derivations of dynamics equations of motion, e.g., Saha and Schiehlen (2001), and Chaudhary and Saha (2003) used $\mathbf{t}_i^O$, and Saha (1997, 1999) used $\mathbf{t}_i^C$, the latter has been adopted in this book mainly due to simplicity in the Euler's equations of rotational motion given by Eq. (8.92) where no explicit term associated with the linear acceleration appear.

9.1.3 Coupled Equations of Motion

The uncoupled NE equations of motion given by Eq. (9.3a) are rewritten as

$$\mathbf{M}\dot{\mathbf{t}} + \mathbf{WMt} = \mathbf{w}^E + \mathbf{w}^C \tag{9.8}$$

where $\mathbf{w}$ of Eq. (9.3a) is substituted as $\mathbf{w} \equiv \mathbf{w}^E + \mathbf{w}^C$—$\mathbf{w}^E$ and $\mathbf{w}^C$ being the external and constraint wrenches, respectively. The external wrench $\mathbf{w}^E$ is contributed from the moments and forces due to the joint actuators, gravity, environmental effects, etc., whereas the constraint wrench $\mathbf{w}^C$ is due to the presence of the joints that contains the reaction moments and forces at the joint interfaces. Since the constraint wrench $\mathbf{w}^C$ does not do any useful work towards the motion of the robot links, the power consumed due to $\mathbf{w}^C$, i.e., $\mathbf{\Pi}^C \equiv \mathbf{t}^T \mathbf{w}^C$, vanishes. The sole purpose of $\mathbf{w}^C$ is to maintain the relative configuration of the links without getting separated. Using the expression for the generalized twist $\mathbf{t}$ from Eq. (9.6a) the vanishing power due to $\mathbf{w}^C$, $\mathbf{\Pi}^C$, is given by

$$\mathbf{\Pi}^C \equiv \mathbf{t}^T\mathbf{w}^C \equiv \dot{\boldsymbol{\theta}}^T \mathbf{N}^T\mathbf{w}^C \equiv \dot{\boldsymbol{\theta}}^T \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^C = 0 \tag{9.9a}$$

For the n-link, n degree-of-freedom (DOF) serial robot, the n-dimensional joint-rate vector $\dot{\boldsymbol{\theta}}$ is independent. Hence, to satisfy Eq. (9.9a), the following condition must hold good:

$$\mathbf{N}^T\mathbf{w}^C \equiv \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^C = \mathbf{0} \tag{9.9b}$$

Now, upon multiplication of the transpose of the NOC, $\mathbf{N}^T$, to the uncoupled NE equations of motion, Eq. (9.8), the following set of independent dynamic equations of motion is obtained:

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + \mathbf{h} = \boldsymbol{\tau}, \text{ where } \mathbf{h} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} \tag{9.10a}$$

where the result of Eq. (9.9b), and the time derivative of the generalized twist $\mathbf{t}$ from Eq. (9.6a), namely, $\dot{\mathbf{t}} = \mathbf{N}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{N}}\dot{\boldsymbol{\theta}}$, are used. Note that Eq. (9.10a), in comparison to the equations of motion derived in Chapter 8, namely, Eq. (8.44a), does not contain the term $\boldsymbol{\gamma}$ due to the gravity. In fact, $\boldsymbol{\tau}$ of Eq. (9.10a) contains the effect of $\boldsymbol{\gamma}$. In a way, $\boldsymbol{\tau}$ of Eq. (9.10a) is equal to $\boldsymbol{\tau}$ plus $\boldsymbol{\gamma}$ of Eq. (8.44a). Moreover,

$\mathbf{I} \equiv \mathbf{N}^T\mathbf{MN} \equiv \mathbf{N}_d^T\tilde{\mathbf{M}}\mathbf{N}_d$: the $n \times n$ generalized inertia matrix (GIM), which is symmetric and positive definite;

$\mathbf{C} \equiv \mathbf{N}^T(\mathbf{M}\dot{\mathbf{N}} + \mathbf{WMN}) \equiv \mathbf{N}_d^T(\tilde{\mathbf{M}}_l + \tilde{\mathbf{M}}_\omega + \tilde{\mathbf{M}}_e)\mathbf{N}_d$: the $n \times n$ matrix of convective inertia (MCI) terms;

$\mathbf{h} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{N}_d^T\,\tilde{\mathbf{w}}'$: the n-dimensional vector of convective inertia (VCI) terms;

$\boldsymbol{\tau} \equiv \mathbf{N}^T\mathbf{w}^E \equiv \mathbf{N}_d^T\tilde{\mathbf{w}}^E$: the n-dimensional vector of generalized forces due to driving torques/forces, and those resulting from the gravity, environment and dissipation.

Also, the $6n \times 6n$ matrices, $\tilde{\mathbf{M}}$, $\tilde{\mathbf{M}}_l$, $\tilde{\mathbf{M}}_\omega$, $\tilde{\mathbf{M}}_e$ and the $6n$-dimensional vectors, $\tilde{\mathbf{w}}^E$ and $\tilde{\mathbf{w}}'$, are given by

$$\tilde{\mathbf{M}} \equiv \mathbf{N}_l^T\,\mathbf{MN}_l;\ \tilde{\mathbf{M}}_l \equiv \mathbf{N}_l^T\,\mathbf{M}\dot{\mathbf{N}}_l,\ \tilde{\mathbf{M}}_\omega \equiv \tilde{\mathbf{M}}\Omega,\ \tilde{\mathbf{M}}_e \equiv \mathbf{N}_l^T\,\mathbf{WMN}_l$$

$$\tilde{\mathbf{w}}' \equiv \mathbf{N}_l^T\,(\mathbf{Mt}' + \mathbf{WMt});\ \mathbf{t}' \equiv (\dot{\mathbf{N}}_l + \mathbf{N}_l\boldsymbol{\Omega})\mathbf{N}_d\,\dot{\boldsymbol{\theta}};\ \text{and}\ \tilde{\mathbf{w}}^E \equiv \mathbf{N}_l^T\,\mathbf{w}^E \tag{9.10b}$$

where $\dot{\mathbf{N}}_d = \boldsymbol{\Omega}\mathbf{N}_d$ in which $\boldsymbol{\Omega} \equiv \text{diag.}[\boldsymbol{\Omega}_1, ..., \boldsymbol{\Omega}_n]$ was used. Note that the 6×6 skew-symmetric matrix $\boldsymbol{\Omega}_i$ associated with the 3-dimensional angular velocity vector $\boldsymbol{\omega}_i$ is different from $\mathbf{W}_i$. The former is defined as $\boldsymbol{\Omega}_i \equiv \text{diag.}\,[\boldsymbol{\omega}_i \times \mathbf{1},\ \boldsymbol{\omega}_i \times \mathbf{1}]$. Moreover, the matrices, $\mathbf{N}_l$, $\mathbf{M}$, $\mathbf{W}$, and the vector $\mathbf{w}^E$ are defined in the previous sections, whereas the vector $\mathbf{t}'$ in Eq. (9.10b) is nothing but the twist-rate vector while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$. It is pointed

out here that Eq. (9.10a) was also derived in Chapter 8, i.e., Eq. (8.44a), using the Euler–Lagrange equations of motion. In the latter case, one required complex partial differentiations, whereas the former is derived here from the Newton–Euler equations of motion which are simpler to visualize in the three-dimensional Cartesian space, and simple linear-algebra concepts.

9.2 ANALYTICAL EXPRESSIONS

It can be shown that using the concept of the DeNOC matrices each element of the GIM, MCI, VCI and the generalized forces can be expressed analytically, which allow one to extract many physical interpretations and suggest ways to simplify the expressions and computational complexity. The analytical expressions are obtained next.

Computation Complexity

In computer algorithms, computational complexity can be estimated in terms of the number of arithmetic operations performed in terms of multiplications, additions, etc.

9.2.1 Generalized Inertia Matrix (GIM)

The expressions for the $n \times n$ GIM $\mathbf{I}$ is given after Eq. (9.10a) as

$$\mathbf{I} \equiv \mathbf{N}^T\mathbf{M}\mathbf{N} \equiv \mathbf{N}_d{}^T\tilde{\mathbf{M}}\mathbf{N}_d \tag{9.11a}$$

where $\tilde{\mathbf{M}} \equiv \mathbf{N}_l{}^T\mathbf{M}\mathbf{N}_l$ is the $6n \times 6n$ symmetric *composite mass matrix*, which is obtained as

$$\tilde{\mathbf{M}} \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1 & & & & \\ \tilde{\mathbf{M}}_2\mathbf{B}_{21} & \tilde{\mathbf{M}}_2 & & sym & \\ \tilde{\mathbf{M}}_3\mathbf{B}_{31} & \tilde{\mathbf{M}}_3\mathbf{B}_{32} & \tilde{\mathbf{M}}_3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \tilde{\mathbf{M}}_n\mathbf{B}_{n1} & \tilde{\mathbf{M}}_n\mathbf{B}_{n2} & \tilde{\mathbf{M}}_n\mathbf{B}_{n3} & \cdots & \tilde{\mathbf{M}}_n \end{bmatrix} \tag{9.11b}$$

where *sym* denotes the symmetric elements of the matrix $\tilde{\mathbf{M}}$. The 6×6 matrix $\tilde{\mathbf{M}}_i$ can be evaluated from $i = n$ to 1 as

$$\tilde{\mathbf{M}}_i \equiv \mathbf{M}_i + \sum_{k=i+1}^{n} \mathbf{B}_{ki}^T\tilde{\mathbf{M}}_k\mathbf{B}_{ki} \tag{9.11c}$$

which requires order n^2 computations, as there is a summation over $k = i + 1, \ldots, n$. A close look into the equation, along with the first two properties of Eq. (9.5a), however, reveal that the summation expression can be evaluated recursively, for $i = n, \ldots, 1$, as

$$\tilde{\mathbf{M}}_l \equiv \mathbf{M}_i + \mathbf{B}^T_{i+1,i}\tilde{\mathbf{M}}_{i+1}\mathbf{B}_{i+1}, \text{ where } \tilde{\mathbf{M}}_n \equiv \mathbf{M}_n \tag{9.11d}$$

Equation (9.11d) has the following physical interpretations:

1. It is the mass matrix of a body composed of links, #*n*, …, #*i*, that are rigidly connected. This is referred as *composite body i*, as indicated in Fig. 9.1(a), whose name can be justified from the 3×3 block matrices of $\tilde{\mathbf{M}}_i$. In Eq. (9.11d), if $i = n$,

$$\tilde{\mathbf{M}}_n = \mathbf{M}_n \equiv \begin{bmatrix} \mathbf{I}_n & \mathbf{O} \\ \mathbf{O} & m_n\mathbf{1} \end{bmatrix} \tag{9.12a}$$

and, for $i = n–1$,

$$\tilde{\mathbf{M}}_{n-1} \equiv \mathbf{M}_{n-1} + \mathbf{B}^T_{n,n-1}\tilde{\mathbf{M}}_n\mathbf{B}_{n,n-1} \tag{9.12b}$$

which can be rewritten as

$$\tilde{\mathbf{M}}_{n-1} \equiv \begin{bmatrix} \tilde{\mathbf{I}}_{n-1} & -\tilde{\boldsymbol{\delta}}_{n-1}\times\mathbf{1} \\ \tilde{\boldsymbol{\delta}}_{n-1}\times 1 & \tilde{m}_{n-1}\mathbf{1} \end{bmatrix} \tag{9.12c}$$

where the scalar $\tilde{m}_{n-1}$, the 3-dimensional vector $\boldsymbol{\delta}_{n-1}$, and the 3 × 3 matrix $\tilde{\mathbf{I}}_{n-1}$ are given by

$$\tilde{m}_{n-1} \equiv m_{n-1} + \tilde{m}_n, \text{ where } \tilde{m}_n = m_n \tag{9.13a}$$

$$\tilde{\boldsymbol{\delta}}_{n-1} \equiv m_n\mathbf{c}_{n,n-1} + \tilde{\boldsymbol{\delta}}_n, \text{ where } \tilde{\boldsymbol{\delta}}_n = 0 \tag{9.13b}$$

$$\tilde{\mathbf{I}}_{n-1} \equiv \mathbf{I}_{n-1} + \tilde{\mathbf{I}}_n - \mathbf{c}_{n,n-1} \times (\tilde{\boldsymbol{\delta}}_{n-1}\times \mathbf{1}), \text{ where } \tilde{\mathbf{I}}_n = \mathbf{I}_n \tag{9.13c}$$

The matrix $\tilde{\mathbf{I}}_{n-1}$ is the inertia tensor of the body composed of rigidly connected links #(n–1) and #n with respect to the mass center of the $(i–1)^{st}$ link, i.e., C_{i-1}, in which the third term is nothing but the one associated with the transfer of the definition of $\mathbf{I}_n$ from C_n to C_{n-1}, similar to the parallel-axis theorem given in Section 8.1.3. Continuing with $i = n–2, \ldots, 1$, the scalar $\tilde{m}_i$, the 3-dimensional vector $\tilde{\boldsymbol{\delta}}_i$, and the 3 × 3 matrix $\tilde{\mathbf{I}}_i$ are calculated as follows:

$$\tilde{m}_i \equiv m_i + \tilde{m}_{i+1} \tag{9.14a}$$

$$\tilde{\boldsymbol{\delta}}_i \equiv \tilde{m}_{i+1}\mathbf{c}_{i+1,i} + \tilde{\boldsymbol{\delta}}_{i+1} \tag{9.14b}$$

$$\tilde{\mathbf{I}}_i \equiv \mathbf{I}_i + \tilde{\mathbf{I}}_{i+1} - \tilde{\boldsymbol{\delta}}_{i+1} \times (\mathbf{c}_{i+1,i}\times\mathbf{1}) - \mathbf{c}_{i+1,i} \times (\tilde{\boldsymbol{\delta}}_i\times\mathbf{1}) \tag{9.14c}$$

2. The inertia effect of the composite body $(i + 1)$, i.e., $\tilde{\mathbf{I}}_{i+1}$, is taken into account with respect to C_i, and added with the inertia tensor of link i with respect to C_i, i.e., $\mathbf{I}_i$, to give the inertia of the composite body i with respect to C_i, i.e., $\tilde{\mathbf{I}}_i$. Other terms, i.e., (2,1) and (2,2)—blocks of Eq. (9.12c) are similarly added to define the mass matrix of the composite body i or the composite mass matrix $\tilde{\mathbf{M}}_i$. Now, using the expression, $\mathbf{I} \equiv \mathbf{N}_d^T\tilde{\mathbf{M}}\mathbf{N}_d$, the symmetric GIM $\mathbf{I}$ is written as

$$\mathbf{I} \equiv \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}}\tilde{\mathbf{M}}_1\mathbf{p}_1 & \cdots & \cdots & \cdots & sym \\ \mathbf{p}_2^{\mathrm{T}}\tilde{\mathbf{M}}_2\mathbf{B}_{21}\mathbf{p}_1 & \mathbf{p}_2^{\mathrm{T}}\tilde{\mathbf{M}}_2\mathbf{p}_2 & \cdots & \cdots & \vdots \\ \mathbf{p}_3^{\mathrm{T}}\tilde{\mathbf{M}}_3\mathbf{B}_{31}\mathbf{p}_1 & \mathbf{p}_3^{\mathrm{T}}\tilde{\mathbf{M}}_3\mathbf{B}_{32}\mathbf{p}_2 & \mathbf{p}_3^{\mathrm{T}}\tilde{\mathbf{M}}_3\mathbf{p}_3 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_n^{\mathrm{T}}\tilde{\mathbf{M}}_n\mathbf{B}_{n1}\mathbf{p}_1 & \mathbf{p}_n^{\mathrm{T}}\tilde{\mathbf{M}}_n\mathbf{B}_{n2}\mathbf{p}_2 & \mathbf{p}_n^{\mathrm{T}}\tilde{\mathbf{M}}_n\mathbf{B}_{n3}\mathbf{p}_3 & \cdots & \mathbf{p}_n^{\mathrm{T}}\tilde{\mathbf{M}}_n\mathbf{p}_n \end{bmatrix} \tag{9.15a}$$

where "*sym*" denotes the symmetric elements, and the (i,j) scalar element of the GIM, denoted by i_{ij}, is given analytically as

$$i_{ij} \equiv \mathbf{p}_i^T\tilde{\mathbf{M}}_i\mathbf{B}_{ij}\mathbf{p}_j, \text{ for } i = n, \ldots, 1; j = i, \ldots, 1 \tag{9.15b}$$

Note that the symmetric elements above the diagonal elements of the GIM, Eq. (9.15a), can be expressed as

$$i_{ji} \equiv \mathbf{p}_j^T \mathbf{B}_{ij}^T \tilde{\mathbf{M}}_i \mathbf{p}_i \text{, for } i = n, \ldots, 1; j = i, \ldots, 1 \tag{9.15c}$$

In Eq. (9.15a), $i_{ij} = i_{ji}$, as obvious from the transpose of the right hand sides of Eqs. (9.15b) and (9.15c). Hence, one is required only to compute either (9.15b) or (9.15c) in a dynamics algorithm. In the examples of this chapter, the lower triangular elements of the GIM are computed using Eq. (9.15b), which require n^2 arithmetic operations—n being the number of links or joints in the serial-chain robot.

9.2.2 Matrix for Convective Inertia (MCI) Terms

Analytical expressions of the matrix of convective inertia (MCI) terms are derived from the MCI definition given after Eq. (9.10a), i.e.,

$$\mathbf{C} \equiv \mathbf{N}^T (\mathbf{M}\dot{\mathbf{N}} + \mathbf{W}\mathbf{M}\mathbf{N}) \equiv \mathbf{N}_d^T (\tilde{\mathbf{M}}_l + \tilde{\mathbf{M}}_\omega + \tilde{\mathbf{M}}_e)\mathbf{N}_d \tag{9.16a}$$

where the $6n \times 6n$ matrices $\tilde{\mathbf{M}}_l$, $\tilde{\mathbf{M}}_w$ and $\tilde{\mathbf{M}}_e$ are reproduced from Eq. (9.10b) as

$$\tilde{\mathbf{M}}_l \equiv \mathbf{N}_l^T \mathbf{M}\dot{\mathbf{N}}_l; \tilde{\mathbf{M}}_\omega \equiv \tilde{\mathbf{M}}\boldsymbol{\Omega} \equiv \mathbf{N}_l^T \mathbf{M}\mathbf{N}_l\boldsymbol{\Omega}; \text{ and } \tilde{\mathbf{M}}_e \equiv \mathbf{N}_l^T \mathbf{W}\mathbf{M}\mathbf{N}_l \tag{9.16b}$$

Matrix $\tilde{\mathbf{M}}_l$ is then obtained as

$$\tilde{\mathbf{M}}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{O} & \mathbf{1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{M}_1 & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \mathbf{M}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{M}_n \end{bmatrix} \begin{bmatrix} \mathbf{O} & \mathbf{O} & \cdots & \mathbf{O} \\ \dot{\mathbf{B}}_{21} & \mathbf{O} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{O} \\ \dot{\mathbf{B}}_{n1} & \cdots & \dot{\mathbf{B}}_{n,n-1} & \mathbf{O} \end{bmatrix} \tag{9.16c}$$

where the 6×6 matrix $\dot{\mathbf{B}}_{i+1,i}$ is given by

$$\dot{\mathbf{B}}_{i+1,i} \equiv \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ -(\dot{\mathbf{r}}_i + \dot{\mathbf{d}}_{i+1}) \times \mathbf{1} & \mathbf{O} \end{bmatrix} \tag{9.16d}$$

in which $\dot{\mathbf{r}}_i = \boldsymbol{\omega}_i \times \mathbf{r}_i$, and $\dot{\mathbf{d}}_{i+1} = \boldsymbol{\omega}_{i+1} \times \mathbf{d}_{i+1}$. The expression of $\tilde{\mathbf{M}}_l$ is then rewritten as

$$\tilde{\mathbf{M}}_l \equiv \begin{bmatrix} \mathbf{B}_{21}^T \tilde{\mathbf{H}}_{21} & \mathbf{B}_{31}^T \tilde{\mathbf{H}}_{32} & \cdots & \mathbf{B}_{n1}^T \tilde{\mathbf{H}}_{n,n-1} & \mathbf{O} \\ \tilde{\mathbf{H}}_{21} & \mathbf{B}_{32}^T \tilde{\mathbf{H}}_{32} & \cdots & \vdots & \vdots \\ \tilde{\mathbf{H}}_{31} & \tilde{\mathbf{H}}_{32} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \mathbf{B}_{n,n-1}^T \tilde{\mathbf{H}}_{n,n-1} & \vdots \\ \tilde{\mathbf{H}}_{n1} & \tilde{\mathbf{H}}_{n2} & \cdots & \tilde{\mathbf{H}}_{n,n-1} & \mathbf{O} \end{bmatrix} \tag{9.16e}$$

In Eq. (9.16e), $\tilde{\mathbf{H}}_{ij} \equiv \tilde{\mathbf{M}}_i \dot{\mathbf{B}}_{ij} + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{H}}_{i+1,i}$, and for, $i = n$, $\tilde{\mathbf{H}}_{n+1,n} = \mathbf{O}$. Next, the $6n \times 6n$ matrix $\tilde{\mathbf{M}}_\omega$, as defined in Eq. (9.16b), is formed as

$$\tilde{\mathbf{M}}_\omega \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1 & \cdots & \cdots & sym \\ \tilde{\mathbf{M}}_2 \mathbf{B}_{21} & \tilde{\mathbf{M}}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}_n \mathbf{B}_{n1} & \tilde{\mathbf{M}}_n \mathbf{B}_{n2} & \cdots & \tilde{\mathbf{M}}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_1 & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\Omega}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O} & \cdots & \cdots & \boldsymbol{\Omega}_n \end{bmatrix} \tag{9.16f}$$

which yields

$$\tilde{\mathbf{M}}_{\omega} \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1\boldsymbol{\Omega}_1 & \mathbf{B}_{21}^T\tilde{\mathbf{M}}_2\boldsymbol{\Omega}_2 & \cdots & \mathbf{B}_{n1}^T\tilde{\mathbf{M}}_n\boldsymbol{\Omega}_n \\ \tilde{\mathbf{M}}_2\mathbf{B}_{21}\boldsymbol{\Omega}_1 & \tilde{\mathbf{M}}_2\boldsymbol{\Omega}_2 & \cdots & \mathbf{B}_{n2}^T\tilde{\mathbf{M}}_n\boldsymbol{\Omega}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}_n\mathbf{B}_{n1}\boldsymbol{\Omega}_1 & \tilde{\mathbf{M}}_n\mathbf{B}_{n2}\boldsymbol{\Omega}_2 & \cdots & \tilde{\mathbf{M}}_n\boldsymbol{\Omega}_n \end{bmatrix} \tag{9.16g}$$

Finally, the matrix $\tilde{\mathbf{M}}_e$ is obtained as

$$\tilde{\mathbf{M}}_e \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{O} & \mathbf{1} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{W}_1\mathbf{M}_1 & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \mathbf{W}_2\mathbf{M}_2 & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{W}_n\mathbf{M}_n \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{B}_{21} & \mathbf{1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & \mathbf{1} \end{bmatrix} \tag{9.16h}$$

which is written in compact form as

$$\tilde{\mathbf{M}}_e \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1' & \cdots & \cdots & sym \\ \tilde{\mathbf{M}}_2'\mathbf{B}_{21} & \tilde{\mathbf{M}}_2' & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}_n'\mathbf{B}_{n1} & \tilde{\mathbf{M}}_n'\mathbf{B}_{n2} & \cdots & \tilde{\mathbf{M}}_n' \end{bmatrix} \tag{9.16i}$$

where $\tilde{\mathbf{M}}_i'$ for $i = n, \ldots, 1$, is calculated similar to matrix $\tilde{\mathbf{M}}_i$, Eq. (9.11d), as

$$\tilde{\mathbf{M}}_i' = \mathbf{M}_i' + \mathbf{B}_{i+1,i}^T\tilde{\mathbf{M}}_{i+1}'\mathbf{B}_{i+1,i} \tag{9.16j}$$

in which $\mathbf{M}_i' = \mathbf{W}_i\mathbf{M}_i$, for $i = n, \ldots, 1$. The scalar elements of the MCI $\mathbf{C}$, i.e., c_{ij}, for $i,j = n, \ldots, 1$, are then obtained explicitly from Eqs. (9.16a-j) as

$$c_{i,j} \equiv \mathbf{p}_i^T[\mathbf{B}_{j+1,i}^T\tilde{\mathbf{H}}_{j+1,i} + \mathbf{B}_{ji}^T(\tilde{\mathbf{M}}_j\boldsymbol{\Omega}_j + \tilde{\mathbf{M}}_j')]\mathbf{p}_j \qquad \text{if } i \le j \tag{9.16k}$$

$$c_{i,j} \equiv \mathbf{p}_i^T(\tilde{\mathbf{H}}_{ij} + \tilde{\mathbf{M}}_i\mathbf{B}_{ij}\boldsymbol{\Omega}_j + \tilde{\mathbf{M}}_i'\mathbf{B}_{ij})\mathbf{p}_j \qquad \text{otherwise.}$$

Comparing Eqs. (9.15b) and (9.16k), it is observed that the elements of the GIM and MCI are expressed in a uniform manner, i.e., $\mathbf{p}^T{}_i(.)\mathbf{p}_j$, where (.) denotes the matrix argument. In Eq. (9.16k), the explicit analytical expression for each element of the MCI is available, which is not advisable to be used for the calculation of vector $\mathbf{h}(\equiv \mathbf{C}\dot{\boldsymbol{\theta}})$ given by Eq. (9.10a) but suitable for the physical interpretations and debugging. Whereas the explicit evaluation of the MCI $\mathbf{C}$ requires order (n^2)—n being the degree of freedom of the robot—calculations, the VCI $\mathbf{h}$ can be computed recursively needing order n operations. The order n algorithm for the VCI is given next. Now, as per as the physical interpretation of the MCI is concerned, note that for a planar robot with all revolute joints, the axes are all parallel and do not change their direction while the robot is in motion. Hence, their time derivative, $\dot{\mathbf{N}}_d = \boldsymbol{\Omega}\mathbf{N}_d = \mathbf{O}$. As a result, the term in Eq.(9.16a) associated with $\tilde{\mathbf{M}}_{\omega}$ vanishes to provide simpler expressions of the MCI $\mathbf{C}$.

9.2.3 Vector of Convective Inertia (VCI) Terms

The vector of convective inertia (VCI) terms $\mathbf{h}$ is given after Eq. (9.10a) is reproduced below:

$$\mathbf{h} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{N}_d^T \tilde{\mathbf{w}}', \text{ where } \tilde{\mathbf{w}}' = \mathbf{N}_l^T(\mathbf{M}\mathbf{t}' + \mathbf{W}\mathbf{M}\mathbf{t}) \text{ and } \mathbf{t}' = (\dot{\mathbf{N}}_l + \mathbf{N}_l \boldsymbol{\Omega})\mathbf{N}_d \dot{\boldsymbol{\theta}} \tag{9.17}$$

Note that $\mathbf{t}'$ is the generalized twist-rate vector while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$, i.e., it contains the centrifugal and Coriolis acceleration terms. Introducing the following notations

$$\mathbf{M}' = \mathbf{W}\mathbf{M} \text{ and } \mathbf{w}' = \mathbf{M}\mathbf{t}' + \mathbf{M}'\mathbf{t}$$

and substituting the expression for $\mathbf{N}_l$ from Eq. (9.6b) into Eq. (9.17) yields

$$\tilde{\mathbf{w}}' \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{O} & \mathbf{1} & & \vdots \\ \vdots & & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1' \\ \mathbf{w}_2' \\ \vdots \\ \mathbf{w}_n' \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}}_1' \\ \tilde{\mathbf{w}}_2' \\ \vdots \\ \tilde{\mathbf{w}}_n' \end{bmatrix} \tag{9.18a}$$

where $\mathbf{w}_i' = \mathbf{M}_i \mathbf{t}_i' + \mathbf{M}_i' \mathbf{t}_i$, for $i = n, \ldots, 1$. Note that the $6n$-dimensional vector $\mathbf{w}'$ can be interpreted as the generalized wrench due to the convective inertia terms. Moreover, the elements of the vector $\tilde{\mathbf{w}}_i'$, Eq. (9.18a), can be obtained recursively as

$$\tilde{\mathbf{w}}_i' = \mathbf{w}_i' + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{w}}_{i+1}', \text{ where } \tilde{\mathbf{w}}_n' = \mathbf{w}_n' \tag{9.18b}$$

Furthermore, using the expression for $\mathbf{N}_d$, the VCI of Eq. (9.17) $\mathbf{h}$ is given by

$$\mathbf{h} \equiv \begin{bmatrix} \mathbf{p}_1^T \tilde{\mathbf{w}}_1' \\ \mathbf{p}_2^T \tilde{\mathbf{w}}_2' \\ \vdots \\ \mathbf{p}_n^T \tilde{\mathbf{w}}_n' \end{bmatrix} \tag{9.19a}$$

in which each element of $\mathbf{h}_i$, denoted as h_i, is written as

$$h_i = \mathbf{p}_i^T \tilde{\mathbf{w}}_i', \text{ for } i = n, \ldots, 1 \tag{9.19b}$$

The expressions in Eqs. (9.19a-b) together provide an order n algorithm for the calculation of the VCI.

9.2.4 Generalized Force

The expressions for the elements of the generalized forces $\boldsymbol{\tau}$, given after Eq. (9.10a) are as follows:

$$\boldsymbol{\tau} = \mathbf{N}_d^T \tilde{\mathbf{w}}^E, \text{ where } \tilde{\mathbf{w}}^E = \mathbf{N}_l^T \mathbf{w}^E \tag{9.20a}$$

Upon substitution of the expression for $\mathbf{N}_l$ from Eq. (9.6b) and noting that, $\mathbf{w}^E \equiv [(\mathbf{w}_1{}^E)^T \ldots (\mathbf{w}_n{}^E)^T]^T$, the $6n$-dimensional vector $\tilde{\mathbf{w}}^E$ is written as

$$\tilde{\mathbf{w}}^E \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n,1}^T \\ \mathbf{O} & \mathbf{1} & & \vdots \\ \vdots & & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1^E \\ \mathbf{w}_2^E \\ \vdots \\ \mathbf{w}_n^E \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}}_1^E \\ \tilde{\mathbf{w}}_2^E \\ \vdots \\ \tilde{\mathbf{w}}_n^E \end{bmatrix} \tag{9.20b}$$

where $\tilde{\mathbf{w}}_i^E \equiv \mathbf{w}_i^E + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{w}}_{i+1}^E$ and $\tilde{\mathbf{w}}_n^E \equiv \mathbf{w}_n^E$. The generalized force $\boldsymbol{\tau}$ is then found as

$$\boldsymbol{\tau} \equiv \begin{bmatrix} \mathbf{p}_1^T \tilde{\mathbf{w}}_1^E \\ \mathbf{p}_2^T \tilde{\mathbf{w}}_2^E \\ \vdots \\ \mathbf{p}_n^T \tilde{\mathbf{w}}_n^E \end{bmatrix} \tag{9.21a}$$

From Eq. (9.21a), each element of the n-dimensional vector $\boldsymbol{\tau}$, i.e., τ_i is then obtained from

$$\tau_i = \mathbf{p}_i^T \tilde{\mathbf{w}}_i^E, \text{ for } i = n, \ldots, 1 \tag{9.21b}$$

Equations (9.10a), (9.15b), (9.16k) or (9.19b), and (9.21b) together provide the explicit expressions for the dynamic equations of motion for the n-link, n-DOF serial robot.

Example 9.1 One-link Planar Arm

For the one-link arm shown in Fig. 9.3, the only equation of motion is derived here using the concept of the DeNOC matrices. Using Eqs. (9.6a-b), the DeNOC matrices for the one-link arm are given by

$$\mathbf{N} = \mathbf{N}_l \mathbf{N}_d = \mathbf{p}, \text{ where } \mathbf{N}_l = \mathbf{1}, \text{ and } \mathbf{N}_d = \mathbf{p} \tag{9.22}$$

The inertia term, which is a scalar for the one-link arm, is then obtained from Eq. (9.15b) as

$$I(\equiv i_{11}) = \mathbf{p}^T \tilde{\mathbf{M}} \mathbf{p} = \mathbf{e}^T \mathbf{I} \mathbf{e} + m(\mathbf{e} \times \mathbf{d})^T (\mathbf{e} \times \mathbf{d});$$

$$\text{where } \mathbf{p} \equiv \begin{bmatrix} \mathbf{e} \\ \mathbf{e} \times \mathbf{d} \end{bmatrix}; \tilde{\mathbf{M}} \equiv \mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & m\mathbf{1} \end{bmatrix} \tag{9.23a}$$

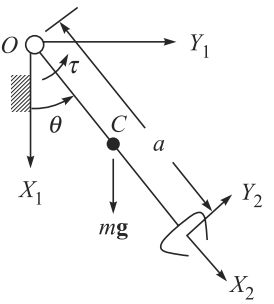


Fig. 9.3 One-link arm

In Eq. (9.23a), no subscript is used to the vector and matrix notations, as there is only one link and one joint. Now, the 3-dimensional vectors **e** and **d** in the fixed frame, i.e., frame 1, are obtained as

$$[\mathbf{e}]_1 \equiv [0 \quad 0 \quad 1]^T; [\mathbf{d}]_1 \equiv \begin{bmatrix} \frac{1}{2}ac & \frac{1}{2}as & 0 \end{bmatrix}^T \tag{9.23b}$$

where $s \equiv \sin\theta$ and $c \equiv \cos\theta$. Moreover, the 3×3 inertia matrix **I** for the one-link arm about its mass center C represented in the link-fixed frame, i.e., frame 2, is written using Eq. (8.22b) as

$$[\mathbf{I}]_2 = \frac{ma^2}{12} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.23c}$$

Using Eq. (8.82), one can write

$$[\mathbf{I}]_1 = \mathbf{Q}[\mathbf{I}]_2\,\mathbf{Q}^T = \frac{ma^2}{12}\begin{bmatrix} c^2 & sc & 0 \\ sc & s^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.23d}$$

where $\mathbf{Q}$ is the 3×3 rotation matrix given by

$$\mathbf{Q} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.23e}$$

Substituting Eqs. (9.23d) in Eq. (9.23a), one gets

$$I = \frac{1}{3}ma^2 \tag{9.23f}$$

Next, the VCI is obtained using Eq. (9.16k), where for the one-link case, $\mathbf{B}_{11} = \mathbf{1}$, $\mathbf{B}_{21}^T\tilde{\mathbf{H}}_{21} = \mathbf{O}$, as it does not arise, and $\tilde{\mathbf{M}}' \equiv \mathbf{M}' = \mathbf{WM}$. Hence,

$$C(\equiv c_{11}) = \mathbf{p}^T(\mathbf{M\Omega} + \mathbf{WM})\mathbf{p} = \dot{\theta}\mathbf{e}^T[(\mathbf{Ie}\times\mathbf{e}) + (\mathbf{e}\times\mathbf{Ie}) - m\mathbf{d}\times\mathbf{d}] = 0 \tag{9.24a}$$

where the expression of the 6×6 matrices $\mathbf{\Omega}$ and $\mathbf{W}$ used above are given by

$$\mathbf{\Omega} = \begin{bmatrix} \dot{\theta}\mathbf{e}\times\mathbf{1} & \mathbf{O} \\ \mathbf{O} & \dot{\theta}\mathbf{e}\times\mathbf{1} \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} \dot{\theta}\,\mathbf{e}\times 1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \tag{9.24b}$$

Moreover, the 6-dimensional vector $\mathbf{p}$ and the 6×6 matrix $\mathbf{M}$ are given in Eq. (9.23a). Furthermore, the properties of vector products were used in obtaining the result of Eq. (9.24a), i.e., $\mathbf{d}\times\mathbf{d} = \mathbf{0}$, and $\mathbf{e}^T(\mathbf{e}\times\mathbf{Ie}) = 0$ or $\mathbf{e}^T(\mathbf{eI}\times\mathbf{e}) = 0$. The latter results are due to the fact that vector $\mathbf{e}$ is orthogonal to the one that is obtained from the cross-product $\mathbf{e}\times\mathbf{Ie}$ or $\mathbf{Ie}\times\mathbf{e}$. Finally, the generalized force is obtained from Eq. (9.21b) as

$$\tau_1 = \mathbf{N}^T\mathbf{w}^E = [\mathbf{e}^T \quad (\mathbf{e}\times\mathbf{d})^T]\begin{bmatrix}\mathbf{n}\\ \mathbf{f}\end{bmatrix} = \tau - \frac{1}{2}mgas \tag{9.24c}$$

where the external moments due to the actuator and external force due to gravity represented in frame 1 are given as

$$[\mathbf{n}]_1 \equiv [0 \quad 0 \quad \tau]^T,\ [\mathbf{f}]_1 \equiv [mg \quad 0 \quad 0 \quad]^T \tag{9.24d}$$

It is pointed out here that for the simple systems like the one in this example, it may be easier to obtain the dynamic equation of motion given by Eqs. (9.23f) and (9.24c) from the direct application of the expressions in terms of the NOC matrix that appear after Eq. (9.10a).

Example 9.2 One-link Arm using MuPAD

Figure 9.4 shows how to develop the equation of motion of the one-link arm of Example 9.1 in the MuPAD environment.

Equation of moition of one-link system using DeNOC matrices

```
[th:=`&theta;`:thd:=Symbol::accentDot(`&theta;`):
[thdd:=Symbol::accentDoubleDot(`&theta;`):tau:=`&tau;`:
[e:=matrix([0,0,1]):eT:=matrix([[0,0,1]]):
[d:=matrix([a/2*cos(th),a/2*sin(th),0]):
[exd1:=e[2]*d[3]-e[3]*d[2]:
[exd2:=e[3]*d[1]-e[1]*d[3]:
[exd3:=e[1]*d[2]-e[2]*d[1]:
[exd:=matrix([exd1,exd2,exd3]):
[exdT:=linalg::transpose(exd):
[p:=matrix([e,exd]):
[pT:=matrix([[eT,exdT]]):
[izz:=iyy:=m*a^2/12:
[Ip:=matrix([[0,0,0],[0,iyy,0],[0,0,izz]]):
[cth:=cos(th):sth:=sin(th):
[Q:=matrix([[cth,-sth,0],[sth,cth,0],[0,0,1]]):
[Qt:=linalg::transpose(Q):
[Ipp:=Q*Ip*Qt:
[m1:=matrix([[m,0,0],[0,m,0],[0,0,m]]):
[Om:=matrix([[0,0,0],[0,0,0],[0,0,0]]):
[M:=matrix([[Ipp,Om],[Om,m1]]):
[iner:=simplify(pT*M*p):
[omv:=matrix([0,0,thd]):
[omx1:=matrix([[0,-omv[3],0],[omv[3],0,0],[0,0,0]]):
[W:=matrix([[omx1,Om],[Om,Om]]):
[h:=pT*(M*W+W*M)*p:
[n:=matrix([0,0,ta]):
[f:=matrix([m*g,0,0]):
[nT:=linalg::transpose(n):
[w:=matrix([n,f]):
[fT:=linalg::transpose(f):
[wT:=matrix([[nT,fT]]):
[tau:=pT*w
```

$$\left(\left(\tau - \frac{a\,g\,m\sin(\theta)}{2}\right)\right)$$

Final equation of motion

```
[eq:=iner*thdd+h=tau
```

$$\left(\left(\frac{\ddot{\theta}\,a^2\,m}{3}\right)\right) = \left(\left(\tau - \frac{a\,g\,m\sin(\theta)}{2}\right)\right)$$

Fig. 9.4 MuPAD commands for the dynamics of one-link arm

Example 9.3 Two-link Planar Arm

A two-link planar arm with two revolute joints is shown in Fig. 9.5. The manipulator has two degrees of freedom, whose independent coordinates are the joint angles θ_1 and θ_2. Correspondingly, the joint rates are $\dot{\theta}_1$ and $\dot{\theta}_2$. The 2-dimensional joint-rate vector $\dot{\boldsymbol{\theta}}$ is then defined as

$$\dot{\boldsymbol{\theta}} \equiv [\dot{\theta}_1, \dot{\theta}_2]^T \tag{9.25}$$

The link lengths are a_1 and a_2, and the masses are m_1 and m_2. The elements of the generalized inertia matrix (GIM) are then calculated using Eq. (9.15b) as follows: For $i, j = 2$, the calculation of i_{22} is shown below

$$i_{22} \equiv \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{B}_{22} \mathbf{p}_2 \tag{9.26a}$$

where

$$\mathbf{p}_2 \equiv \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_2 \times \mathbf{d}_2 \end{bmatrix}; \mathbf{B}_{22} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}; \text{ and } \tilde{\mathbf{M}}_2 = \mathbf{M}_2 \equiv \begin{bmatrix} \mathbf{I}_2 & \mathbf{O} \\ \mathbf{O} & m_2 \mathbf{1} \end{bmatrix} \tag{9.26b}$$

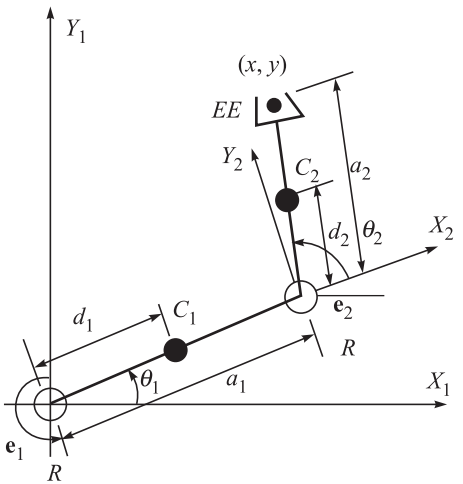


Fig. 9.5 Two-link planar arm

where $[\mathbf{e}_2]_2 \equiv [\mathbf{e}_2]_1 \equiv [0, 0, 1]^T$ is the unit vector along the axis of the second revolute joint, as indicated in Fig. 9.5. Vector $\mathbf{d}_2$ is the position of C_2, whereas $\mathbf{I}_2$ is the inertia tensor about C_2. Using Eq. (9.26b) and assuming the mass center lies at the center of the link length, i.e., $d_2 = a_2/2$, the (2,2)-element of the GIM i_{22} is then explicitly obtained, similar to $I(\equiv i_{11})$ of Example 9.1 as

$$\begin{aligned} i_{22} &= [\mathbf{e}_2]_1^T [\mathbf{I}_2]_1 [\mathbf{e}_2]_1 + m_2 [\mathbf{d}_2]_1^T [\mathbf{d}_2]_1 \\ &= \frac{1}{12} m_2 a_2^2 + \frac{1}{4} m_2 a_2^2 = \frac{1}{3} m_2 a_2^2 \end{aligned} \tag{9.27a}$$

Next, for $i = 2, j = 1$, the element i_{21} is given next as

$$i_{21} = \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \mathbf{p}_1, \text{ where } \mathbf{p}_1 \equiv \begin{bmatrix} \mathbf{e}_1^T & (\mathbf{e}_1 \times \mathbf{d}_1)^T \end{bmatrix}^T \tag{9.27b}$$

and the matrix $\mathbf{B}_{21}$ is given by

$$\mathbf{B}_{21} = \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ -(\mathbf{r}_1 + \mathbf{d}_2)\times\mathbf{1} & \mathbf{1} \end{bmatrix} \tag{9.27c}$$

where $\mathbf{r}_1 = \mathbf{a}_1 - \mathbf{d}_1$. Hence, the (2,1)-element of the GIM, i.e., i_{21} is calculated, similar to i_{22}, as

$$\begin{aligned} i_{21} &= \mathbf{e}_2^T \mathbf{I}_2 \mathbf{e}_1 + m_2(\mathbf{e}_2 \times \mathbf{d}_2)^T [\mathbf{e}_1 \times (\mathbf{a}_1 + \mathbf{d}_2)] = \mathbf{e}_2^T \mathbf{I}_2 \mathbf{e}_1 + m_2 \mathbf{d}_2^T \mathbf{a}_1 + m_2 \mathbf{d}_2^T \mathbf{d}_2 \\ &= \frac{1}{12} m_2 a_2^2 + \frac{1}{2} m_2 a_1 a_2 c_2 + \frac{1}{4} m_2 a_2^2 = \frac{1}{3} m_2 a_2^2 + \frac{1}{2} m_2 a_1 a_2 c_2 \end{aligned} \tag{9.27d}$$

In Eq. (9.27d), the vector product rule, namely, $(\mathbf{a} \times \mathbf{b})^T(\mathbf{c} \times \mathbf{d}) = (\mathbf{a}^T\mathbf{c})(\mathbf{b}^T\mathbf{d}) - (\mathbf{a}^T\mathbf{d})(\mathbf{b}^T\mathbf{c})$ was applied. Moreover, $\mathbf{e}_1 = \mathbf{e}_2$, $\mathbf{e}_2^T\mathbf{e}_2 = 1$ because $\mathbf{e}_2$ is a unit vector, and $\mathbf{e}_2^T\mathbf{d}_1 = 0$ because $\mathbf{e}_2$ is orthogonal to $\mathbf{d}_1$. Furthermore, $\mathbf{d}_i = \mathbf{a}_i/2$, $\mathbf{r}_i + \mathbf{d}_i = \mathbf{a}_i$, for $i = 1$, 2, and $\mathbf{d}_2^T \mathbf{a}_1 = \frac{1}{2} a_1 a_2 c_2$. Next, matrix $\tilde{\mathbf{M}}_1$ is calculated as

$$\tilde{\mathbf{M}}_1 = \mathbf{M}_1 + \mathbf{B}_{21}^T \tilde{\mathbf{M}}_2 \mathbf{B}_{21} = \begin{bmatrix} \tilde{\mathbf{I}}_1 & -\tilde{\boldsymbol{\delta}}_1 \times \mathbf{1} \\ \tilde{\boldsymbol{\delta}}_1 \times \mathbf{1} & \tilde{m}_1 \mathbf{1} \end{bmatrix} \tag{9.28a}$$

where $\tilde{\mathbf{I}}_1$, $\tilde{\boldsymbol{\delta}}_1$, and $\tilde{m}_1$ are given below.

$$\tilde{\mathbf{I}}_1 = \mathbf{I}_1 + \mathbf{I}_2 - m_2 \mathbf{c}_{21} \times (\tilde{\boldsymbol{\delta}}_1 \times \mathbf{1}); \; \tilde{\boldsymbol{\delta}}_1 = m_2 \mathbf{c}_{21}; \text{ and } \tilde{m}_1 = m_1 + m_2 \tag{9.28b}$$

in which $\tilde{\mathbf{I}}_2 \equiv \mathbf{I}_2$, $\tilde{\boldsymbol{\delta}}_2 = \mathbf{0}$, and $\tilde{m}_2 = m_2$ are used. Now, for $i, j = 1$, the (1, 1)-element of the GIM i_{11} is now found as

$$i_{11} = \mathbf{e}_1^T \tilde{\mathbf{I}}_1 \mathbf{e}_1 + \tilde{m}_1 \mathbf{d}_1^T \mathbf{d}_1 = \frac{1}{3}(m_1 a_1^2 + m_2 a_2^2) + m_2 a_1^2 + m_2 a_1 a_2 c_2 \tag{9.29}$$

For $i,j = 2, 1$, the MCI elements are given from Eq. (9.16k) as

$$c_{22} = \mathbf{p}_2^T[(\mathbf{B}_{32}^T \tilde{\mathbf{H}}_{32} + \mathbf{B}_{22}^T(\tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 + \tilde{\mathbf{M}}_2')]\mathbf{p}_2 = 0 \tag{9.30a}$$

$$c_{21} = \mathbf{p}_2^T[\tilde{\mathbf{H}}_{21} + \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \boldsymbol{\Omega}_1 + \tilde{\mathbf{M}}_2' \mathbf{B}_{21}]\,\mathbf{p}_1 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1 \tag{9.30b}$$

$$c_{12} = \mathbf{p}_1^T[\mathbf{B}_{31}^T \tilde{\mathbf{H}}_{32} + \mathbf{B}_{21}^T(\tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 + \tilde{\mathbf{M}}_2')]\,\mathbf{p}_2 = -\frac{1}{2} m_2 a_1 a_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \tag{9.30c}$$

$$c_{11} = \mathbf{p}_1^T[\mathbf{B}_{21}^T \tilde{\mathbf{H}}_{21} + \mathbf{B}_{11}^T(\tilde{\mathbf{M}}_1 \boldsymbol{\Omega}_1 + \tilde{\mathbf{M}}_1')]\,\mathbf{p}_1 = -\frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_2 \tag{9.30d}$$

In Eqs. (9.30a-d), $\tilde{\mathbf{H}}_{32} \equiv \mathbf{O}$ and $\tilde{\mathbf{M}}_2' \mathbf{p}_2 = \mathbf{0}$ were used. Moreover, for planar robots, it can be shown that the term $\mathbf{p}_i^T \mathbf{B}_{ji}^T \tilde{\mathbf{M}}_j \boldsymbol{\Omega}_j \mathbf{p}_j$, for $i,j = 1, 2$, vanishes. Finally, the elements of vector $\mathbf{h}$ can be shown to have the following expressions:

$$h_2 = \mathbf{p}_2^T \tilde{\mathbf{w}}_2' = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 \tag{9.31a}$$

$$h_1 = \mathbf{p}_1^T \tilde{\mathbf{w}}_1' = -m_2 a_1 a_2 s_2 \left(\frac{1}{2}\dot{\theta}_2 + \dot{\theta}_1\right)\dot{\theta}_2 \tag{9.31b}$$

which are nothing but the result of the multiplication of the elements of the MCI, Eqs. (9.30a-d), with those of the joint rate vector $\dot{\boldsymbol{\theta}}$ of Eq. (9.25).

9.3 RECURSIVE INVERSE DYNAMICS OF ROBOANALYZER†

Inverse dynamics is defined as "given a robot's geometrical and inertial parameters, along with its joint motions, find the joint torques and forces." In this section, it is emphasized that for the purpose of inverse dynamics analysis it is not required to explicitly evaluate the matrices and vectors appearing in Eq. (9.10a), as the required right-hand side can be evaluated using a recursive order (n) algorithm (Saha, 1999), where n is the number of links in the robot manipulator. A recursive algorithm in the C++ program was initially developed for the dynamic analysis of serial robots, which was given an acronym RIDIM (Recursive Inverse Dynamics for Industrial Manipulators) (Marothiya and Saha, 2003). Later, it was superseded with RoboAnalzyer (RA) which was written in C# and based on a similar algorithm called ReDySim (Recursive Dynamics Simulator) (Shah et al., 2013) using the DeNOC matrices. The RA in comparison to ReDySim has many additional features like 3-dimensional models of robots with animation, kinematic analyses, trajectory planning, etc. See Appendix C for more details.

The recursive algorithm (Saha, 1999) implemented in RoboAnalyzer is presented next. It has two recursions, namely, forward and backward. They are given below:

1. Forward Recursion (Kinematic Equations)

$$\mathbf{t}_1 = \mathbf{p}_1\dot{\theta}_1; \qquad \dot{\mathbf{t}}_1 = \mathbf{p}_1\ddot{\theta}_1 + \boldsymbol{\Omega}_1\mathbf{p}_1\dot{\theta}_1 \tag{9.32a}$$

$$\mathbf{t}_2 = \mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2; \qquad \dot{\mathbf{t}}_2 = \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{p}_2\ddot{\theta}_2 + \boldsymbol{\Omega}_2\mathbf{p}_2\dot{\theta}_2 \tag{9.32b}$$

$$\vdots \qquad\qquad \vdots$$

$$\mathbf{t}_n = \mathbf{B}_{n,n-1}\mathbf{t}_{n-1} + \mathbf{p}_n\dot{\theta}_n; \quad \dot{\mathbf{t}}_n = \mathbf{B}_{n,n-1}\dot{\mathbf{t}}_{n-1} + \dot{\mathbf{B}}_{n,n-1}\mathbf{t}_{n-1} + \mathbf{p}_n\ddot{\theta}_n + \boldsymbol{\Omega}_n\mathbf{p}_n\dot{\theta}_n \tag{9.32c}$$

where the 6×6 matrix $\boldsymbol{\Omega}_i$ was defined as after Eq. (9.10b).

2. Backward Recursion (Dynamic Equations)

$$\mathbf{w}_n = \mathbf{M}_n\dot{\mathbf{t}}_n + \mathbf{W}_n\mathbf{M}_n\mathbf{t}_n; \ \tilde{\mathbf{w}}_n = \mathbf{w}_n; \qquad \tau_n = \mathbf{p}_n^T\tilde{\mathbf{w}}_n \tag{9.33a}$$

$$\mathbf{w}_{n-1} = \mathbf{M}_{n-1}\dot{\mathbf{t}}_{n-1} + \mathbf{W}_{n-1}\mathbf{M}_{n-1}\mathbf{t}_{n-1}; \ \tilde{\mathbf{w}}_{n-1} = \mathbf{w}_{n-1} + \mathbf{B}_{n,n-1}^T\tilde{\mathbf{w}}_n; \ \tau_{n-1} = \mathbf{p}_{n-1}^T\tilde{\mathbf{w}}_{n-1} \tag{9.33b}$$

$$\vdots$$

$$\mathbf{w}_1 = \mathbf{M}_1\dot{\mathbf{t}}_1 + \mathbf{W}_1\mathbf{M}_1\mathbf{t}_1; \ \tilde{\mathbf{w}}_1 = \mathbf{w}_1 + \mathbf{B}_{21}^T\tilde{\mathbf{w}}_2; \qquad \tau_1 = \mathbf{p}_1^T\tilde{\mathbf{w}}_1 \tag{9.33c}$$

where $\mathbf{t}_i$, $\dot{\mathbf{t}}_i$ and $\mathbf{w}_i$ are the 6-dimensional vectors of twist, twist-rate, and the resultant wrench on the uncoupled i^{th} link. In the above algorithm, no moment and force, i.e., wrench, acting on the end-effector was considered. In case of their presence they should be appropriately included, i.e., Eq. (9.33a) is modified as

$$\mathbf{w}_n = \mathbf{M}_n\dot{\mathbf{t}}_n + \mathbf{W}_n\mathbf{M}_n\mathbf{t}_n - \mathbf{w}_n^N \tag{9.34}$$

where $\mathbf{w}_n^N$ is the wrench acting on the end-effector by the environment. Rest of

Multiplication vs. Division

In computer calculations, a multiplication is treated similar to a division due to the internal algorithms used by a computer. Hence, (*a***b*)/*c* needs two multiplications (M).

† The dynamics algorithm of RoboAnalyzer reported here is basically same as that of Recursive Inverse Dynamics for Industrial Manipulators (RIDIM) appeared in the 1st edition of this book.

the step remains same. Moreover, if gravity is present it is taken into account by providing negative acceleration due to the gravity, denoted by **g**, to the first body, i.e., link #1 (Luh, Walker and Paul, 1980). Accordingly, $\dot{\mathbf{t}}_1$ of Eq. (9.32a) is modified as

$$\dot{\mathbf{t}}_1 = \mathbf{p}_1\ddot{\theta}_1 + \boldsymbol{\Omega}_1\mathbf{p}_1\dot{\theta}_1 + \boldsymbol{\rho}\text{, where } \boldsymbol{\rho} \equiv [\mathbf{0}, -\mathbf{g}^T]^T \tag{9.35}$$

Now, for an all revolute-jointed serial robotic manipulator, the computational complexity of the above algorithm is shown in Table 9.1, which is compared with some other existing inverse dynamics algorithms. The details of how to calculate the computational complexity are appeared in Appendix A of Shah et al. (2013). The above algorithm is one of the best. Besides, it is very simple, as evident from the two-step algorithm where six-dimensional vectors are treated similar to those of three dimensional vectors of recursive Newton-Euler algorithm as presented in Chapter 8. A close look into the algorithms actually exposes that both are fundamentally same. The latter one is, however, more elegant and concise. That is, if one knows $\mathbf{t}_1$ finding out $\mathbf{t}_2$ is very similar to the evaluation of the linear velocity of link #2 from the known linear velocity of link #1.

Addition vs. Subtraction

In computer calculations, an addition is similar to a subtraction due to the internal computer algorithms. Hence, computation count to find $a + b - c$ is two additions (A).

Table 9.1 Computational complexities for inverse dynamics

Algorithm	Multiplications/ Divisions (M)	Additions/ Subtractions (A)	$n = 6$	
Hollerbach (1980)	$412n - 277$	$320n - 201$	2195M	1719A
Luh et al. (1980)	$150n - 48$	$131n + 48$	852M	834A
Walker and Orin (1982)	$137n - 22$	$101n - 11$	800M	595A
RIDIM (Saha, 1999)	$120n - 44$	$97n - 55$	676M	527A
Khalil et al. (1986)	$105n - 92$	$94n - 86$	538M	478A
Angeles et al. (1989)	$105n - 109$	$90n - 105$	521M	435A
ReDySim (Shah et al., 2013)	$\mathbf{94n - 81}$	$\mathbf{82n - 75}$	**483M**	**417A**
Balafoutis et al. (1991)	$93n - 69$	$81n - 65$	489M	421M

Example 9.4 Inverse Dynamics of Three-DOF Planar Arm

The three-link three-DOF manipulator under study, is shown in Fig. 9.6, whose DH and inertia parameters are shown in Table 9.2. It is assumed that the manipulator moves on the *X*-*Y* plane, where the gravity is working in the negative *Y* direction. Let **i** and **j** be the two unit vectors parallel to the axes *X* and *Y*, respectively, and **k** = **i** × **j** is the one parallel to Z-axis. The three joint torques, namely, τ_1, τ_2, and τ_3, were evaluated using RoboAnalyzer software. The joint angles used to generate the joint torques are as follows: For i = 1, 2 and 3,

$$\theta_i = \theta_i(0) + \frac{\theta_i(T) - \theta_i(0)}{T}\left[t - \frac{T}{2\pi}\sin\left(\frac{2\pi}{T}t\right)\right] \tag{9.36a}$$

$$\dot{\theta}_i = \frac{\theta_i(T) - \theta_i(0)}{T}\left[1 - \cos\left(\frac{2\pi}{T}t\right)\right] \tag{9.36b}$$

$$\ddot{\theta}_i = \frac{\theta_i(T) - \theta_i(0)}{T}\left[\frac{2\pi}{T}\sin\left(\frac{2\pi}{T}t\right)\right] \tag{9.36c}$$

where T = 10 seconds. The initial and final joint values were taken as $\theta_i(0) = 0$ and $\theta_i(T) = \pi$. Moreover, Eqs. (9.36a-c) guarantee that $\dot{\theta}_i(0) = \ddot{\theta}_i(0) = \dot{\theta}_i(T) = \ddot{\theta}_i(T) = 0$, for i = 1, 2, 3. The plots for the joint angle and their time derivatives are shown in Fig. 9.7, the joint torques obtained from RoboAnalyzer are then plotted in Fig. 9.8, where the terms, τ_1, τ_2, and τ_3, are the joint torques. The plots were also verified with those obtained from the explicit expressions available in many textbooks on robotics, e.g., Angeles (2003).

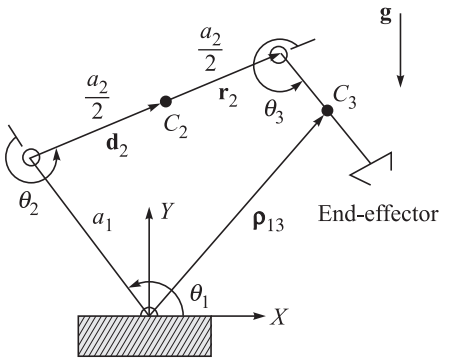


Fig. 9.6 Three-DOF planar robot arm

Table 9.2 The DH and inertia parameters of the three-DOF robot arm

(a) DH parameters

Link	Joint	a_i (m)	b_i (m)	α_i (rad)	θ_i (rad)
1	r	0.3	0	0	JV [0]
2	r	0.25	0	0	JV [0]
3	r	0.22	0	0	JV [0]

JV: Joint variable with initial values inside brackets [and]; r: Revolute joint

(b) Mass and inertia parameters

Link	m_i	$r_{i,x}$	$r_{i,y}$	$r_{i,z}$	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
	(kg)	(m)			(kg-m^2)					
1	0.5	0.15	0	0	0	0	0	0.00375	0	0.00375
2	0.4	0.125	0	0	0	0	0	0.00208	0	0.00208
3	0.3	0.11	0	0	0	0	0	0.00121	0	0.00121

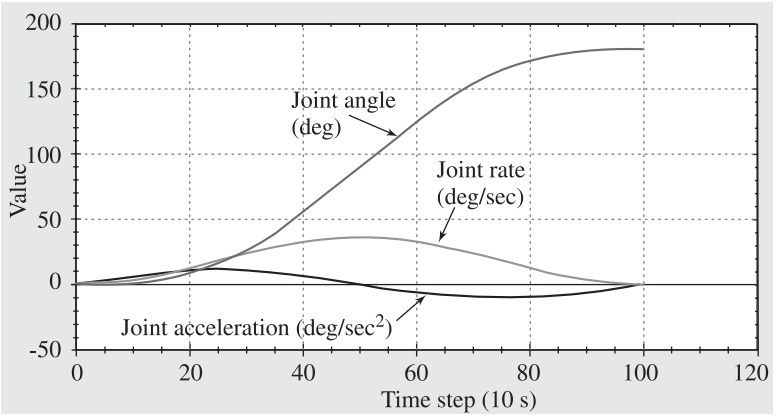


Fig. 9.7 Joint trajectory

A screenshot of the 3-dimensional view of the 3-DOF robot arm in RoboAnalyzer with its kinematic parameters visible at the bottom of the screen are shown in Fig. 9.9.

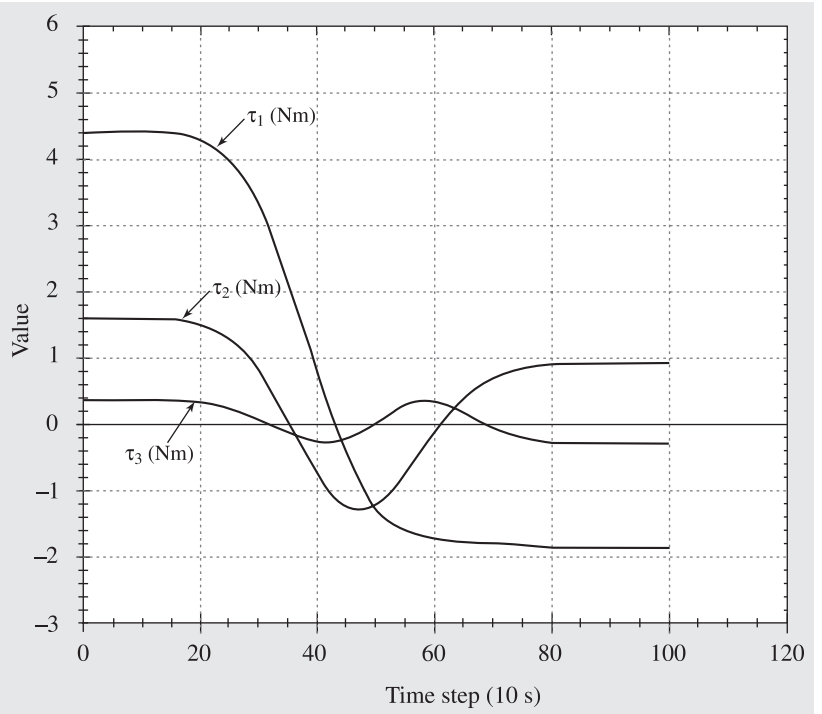


Fig. 9.8 Joint torques for the three-DOF arm

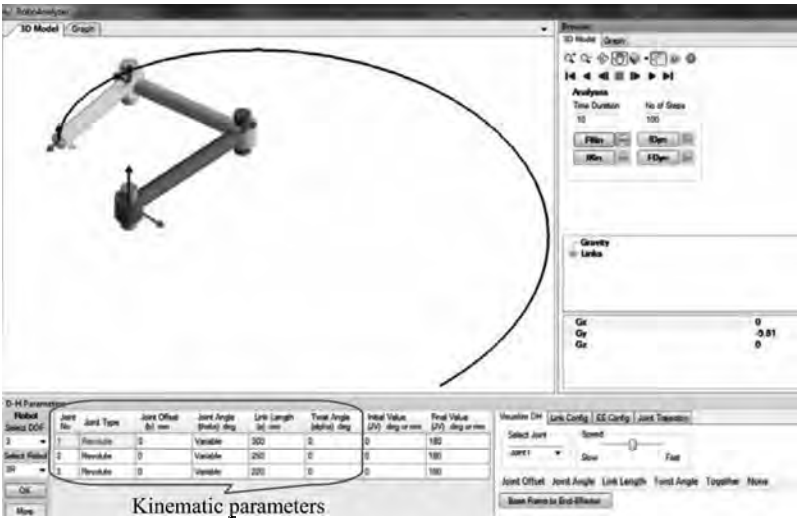


Fig. 9.9 RoboAnalyzer screenshot of three-link robot arm

Example 9.5 Inverse Dynamics of PUMA Architecture

For the six-DOF robot with PUMA architecture shown in Fig. 9.10, the inverse dynamics results were obtained for its DH and inertial parameters shown in Table 9.3.

Table 9.3 The DH and inertia parameters of PUMA robot

(a) DH parameters

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0	JV [0]	0	$-\pi/2$
2	r	0.149	JV [0]	0.432	0
3	r	0	JV [0]	0.02	$-\pi/2$
4	r	0.432	JV [0]	0	$-\pi/2$
5	r	0	JV [0]	0	$-\pi/2$
6	r	0.05	JV [0]	0	0

JV: Joint variable with initial values inside [and]; r: Revolute joint

(b) Mass and inertia parameters

Link	m_i	$r_{i,x}$	$r_{i,y}$	$r_{i,z}$	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
	(kg)	(m)			(kg-m^2)					
1	10.521	0	0	0.054	1.612	0	0	1.612	0	0.5091
2	15.761	0.292	0	0	0.4898	0	0	8.0783	0	8.2672
3	8.767	0.02	0	–0.197	3.3768	0	0	3.3768	0	0.3009
4	1.052	0	–0.057	0	0.181	0	0	0.1273	0	0.181
5	1.052	0	0	–0.007	0.0735	0	0	0.1273	0	0.0735
6	0.351	0	0	0.019	0.0071	0	0	0.0071	0	0.0141

Note that the DH frames in Fig. 9.10 are assigned little differently, namely, frames 5 and 6, compared to those in Fig. 6.6 for the same robot. Hence, there are some changes in the DH parameters in Table 9.3 compared to those in Table 6.2. This was done intentionally to illustrate the variation in the assignment of DH frames for the same architecture of a robot. The trajectory functions for each joint is taken same as for the three-link robot arm, as defined in Eq. (9.36a-c) for the same T, and initial and final joint values. Corresponding joint torque plots generated from RoboAnalyzer software are given in Figs. 9.11(a-f).

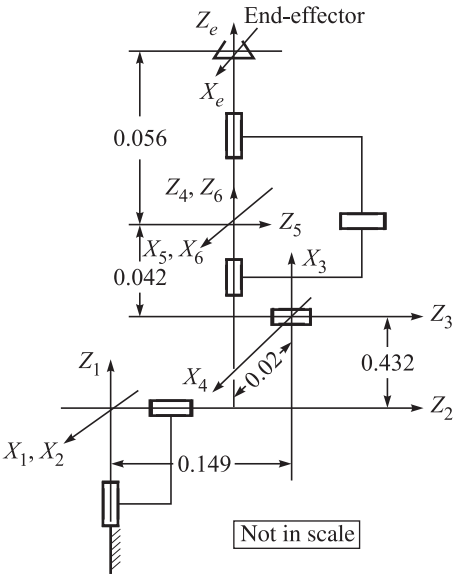
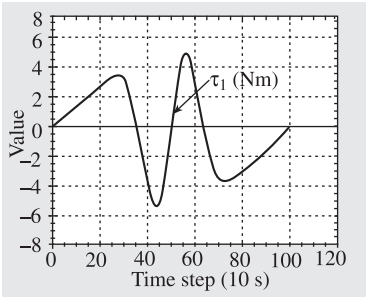
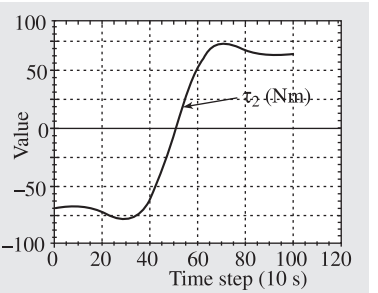


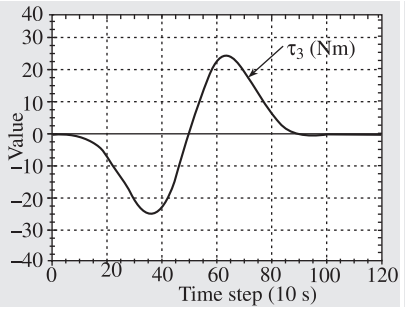
Fig. 9.10 A PUMA architecture



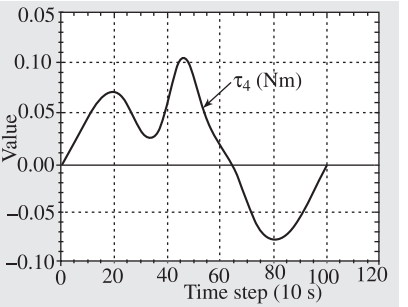
(a) Joint 1



(b) Joint 2

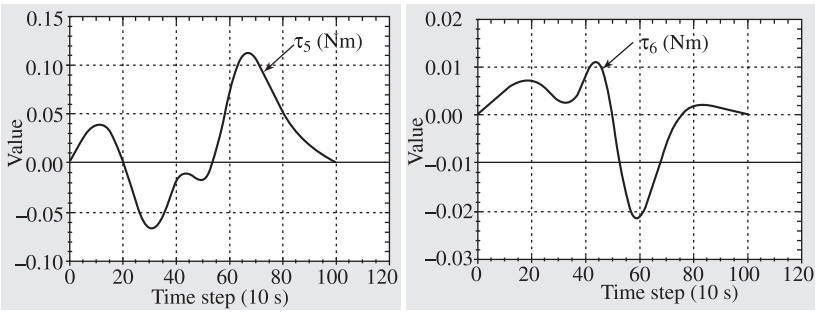


(b) Joint 3



(d) Joint 4

(*Contd.*)



(e) Joint 5 (f) Joint 6

Fig. 9.11 Joint torques for PUMA architecture

Example 9.6 Inverse Dynamics of Stanford Arm

For the Stanford arm shown in Fig. 9.12, the DH and other parameters are shown in Table 9.4. Note that it differentiates from the PUMA robot in a way that it has a prismatic joint in joint location 3. Moreover, the DH frames are assigned differently than in Fig. 6.8(a) due to the reasons cited for PUMA robot of Example 9.5. The functions for the revolute joints were taken same as in Example 9.5, whereas the function of the prismatic joint variable, namely b_3 is taken as

$$b_i = b_i(0) + \frac{b_i(T) - b_i(0)}{T}\left[t - \frac{T}{2\pi}\sin\left(\frac{2\pi}{T}t\right)\right] \tag{9.37}$$

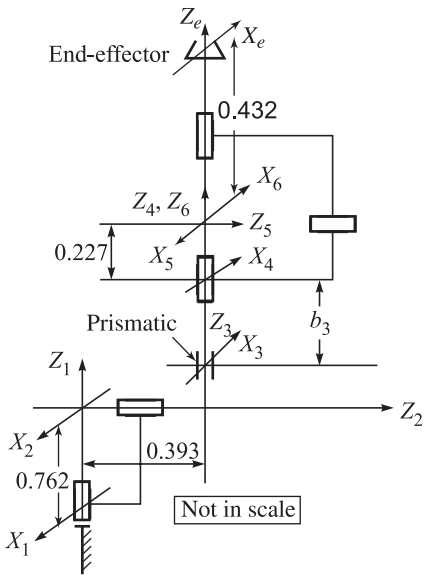


Fig. 9.12 The Stanford arm

Table 9.4 The DH and inertia parameters of the Stanford arm

(a) DH parameters

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0.762	JV [0]	0	$-\pi/2$
2	r	0.393	JV [$-\pi/2$]	0	$\pi/2$
3	p	JV [0.635]	0	0	0
4	r	0.227	JV [0]	0	$-\pi/2$
5	r	0	JV [π]	0	$-\pi/2$
6	r	0.432	JV [π]	0	0

JV: Joint variable with initial values within [and]; r: Revolute joint; p: Prismatic joint

(b) Mass and inertia parameters

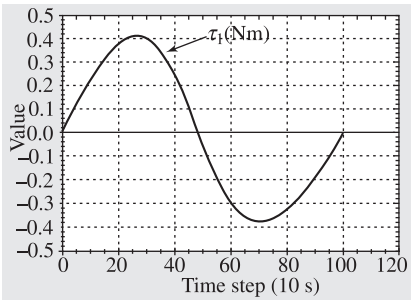
Link	m_i	$r_{i,x}$	$r_{i,y}$	$r_{i,z}$	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
	(kg)		(m)				(kg-m^2)			
1	9	0	–0.1	0	0.01	0	0	0.02	0	0.01
2	6	0	0	0	0.05	0	0	0.06	0	0.01
3	4	0	0	0	0.4	0	0	0.4	0	0.01
4	1	0	–0.1	0	0.001	0	0	0.001	0	0.0005
5	0.6	0	0	–0.06	0.0005	0	0	0.0005	0	0.0002
6	0.5	0	0	–0.2	0.003	0	0	0.001	0	0.002

For T = 10 seconds, initial joint values were taken as follows:

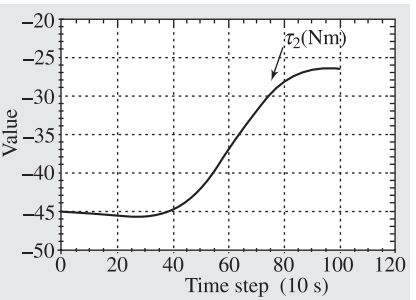
$\theta_1(0) = 0$, $\theta_2(0) = -\pi/2$, $b_3(0) = 0.635$ m, $\theta_4(0) = 0$, $\theta_5(0) = \theta_6(0) = \pi$

$\theta_1(T) = \pi/3$, $\theta_2(T) = -5\pi/6$, $b_3(T) = 0.735$ m, $\theta_4(T) = \pi/3$, $\theta_5(T) = \theta_6(T) = 5\pi/6$

The above values were taken in a way so that the robot links do not interfere with each other. The joint torques and force were obtained using RoboAnalzyer are plotted in Figs. 9.13(a-f).

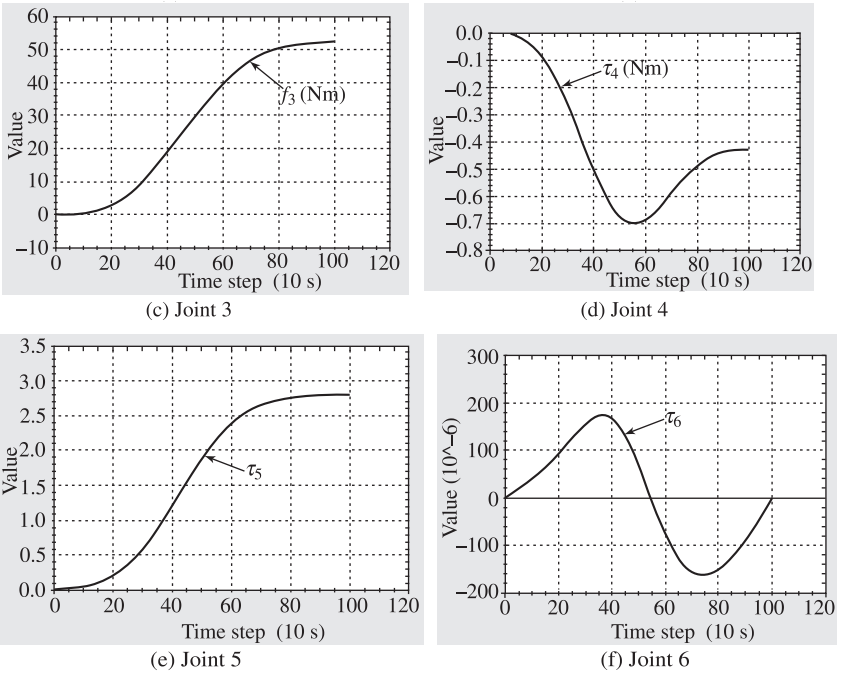


(a) Joint 1



(b) Joint 2

(*Contd.*)



(c) Joint 3 (d) Joint 4

(e) Joint 5 (f) Joint 6

Fig. 9.13 Joint torques and force for Stanford arm

Example 9.7 Inverse Dynamics of KUKA KR-5 Robot

In this example, a practically used industrial robot, namely, KUKA KR-5 (5 stands for the 5 kg payload capacity of the robot), is taken up. The CAD model available from the company's website (WR: Kuka) has been processed in Autodesk Inventor software to extract the DH parameters of the robot using an in-house developed add-in for the Autodesk Inventor software (Rajeevlochana, 2012). They are shown in Table 9.5(a), whereas the mass and inertia properties were taken from the CAD model, as shown in Table 9.5(b). The function of the trajectory for each joint was taken same as in Eq. (9.36a-c) with $\theta_i(0) = 0$ and $\theta_i(T) = \pi/3$, and $T = 10$ seconds. Here, the joint trajectories were taken in a way that the robot links do not interfere with each other. The Cartesian trajectory followed by the robot is shown in Fig. 9.14, whereas the required torque plots are given in Fig. 9.15.

Fig. 9.14 KUKA KR-5 during its motion

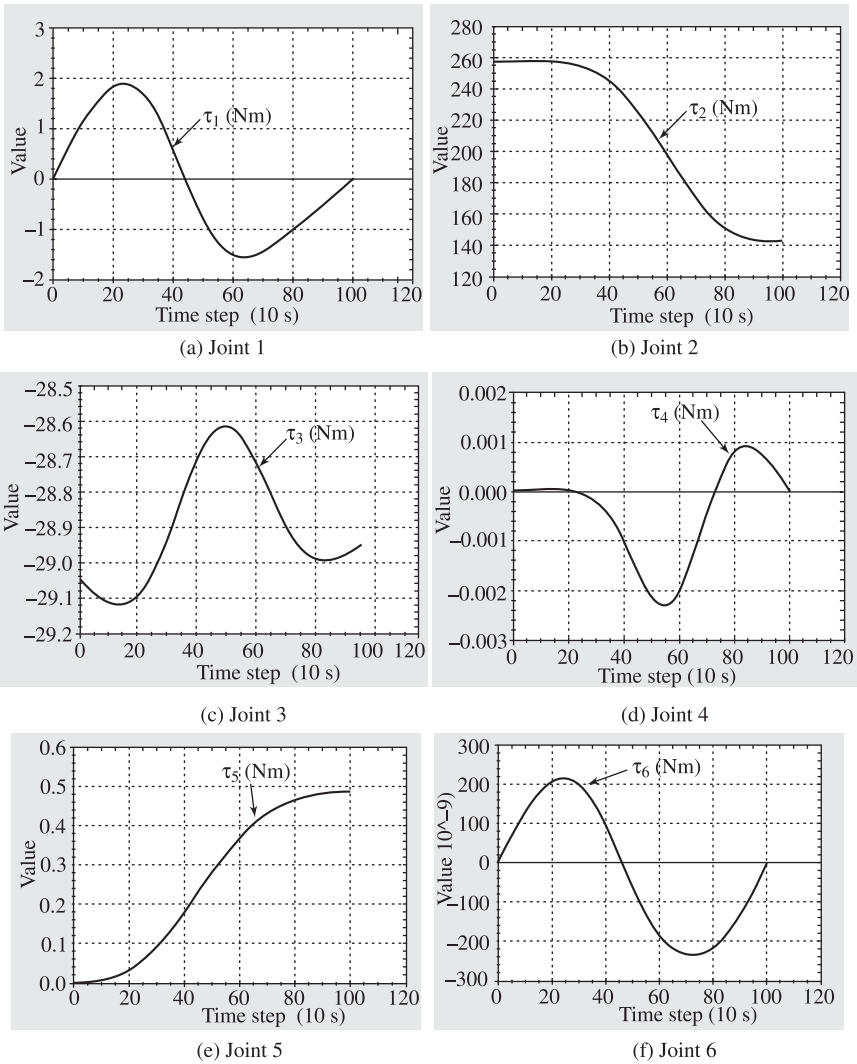


Fig. 9.15 Joint torques for KUKA KR-5

Table 9.5 The DH and inertia parameters of KUKA KR-5 robot

(a) DH parameters

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0.4	JV [0]	0.180	$\pi/2$
2	r	0.135	JV [0]	0.600	π
3	r	0.135	JV [0]	0.120	$-\pi/2$
4	r	0.620	JV [0]	0	$\pi/2$
5	r	0	JV [0]	0	$-\pi/2$
6	r	0.115	JV [0]	0	0

JV: Joint variable with initial values within [and]; r: Revolute joint

(*Contd.*)

(b) Mass and inertia parameters

m_i	$r_{i,x}$	$r_{i,y}$	$r_{i,z}$	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
(kg)	(m)			(kg-m^2)					
26.98	0.091	0.067	0.006	0.322	–0.018	–0.145	0.467	–0.014	0.478
15.92	0.333	0.002	0.039	0.541	0.000435	–0.005	0.552	0.017	0.044
25.852	0.032	–0.008	–0.034	0.775	–0.009	0.025	0.75	0.007	0.208
4.008	0	0.109	–0.008	0.01	0.002	0	0.02	0	0.024
1.615	0	–0.01	–0.033	0.002	0	0	0.004	0	0.004
0.016	0	0	–0.111	0	0	0	0	0	0

9.4 RECURSIVE FORWARD DYNAMICS AND SIMULATION

Forward dynamics is defined as "given the joint torques and forces, along with the robot's physical parameters, find the joint accelerations, i.e., solve for the joint accelerations $\ddot{\boldsymbol{\theta}}$ from the dynamic equations of motion Eq. (9.10a)." Simulation, on the other hand, involves forward dynamics followed by the solution of the differential equations in joint accelerations to obtain the joint velocities and positions, i.e., $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$, respectively, for a given set of initial joint rates and positions of the manipulator under study, i.e., $\dot{\boldsymbol{\theta}}(0)$ and $\boldsymbol{\theta}(0)$.

Why "Forward?"

This step of simulation leads to the behavior of a robot as if a the robot exists and one moves it. Hence, the word *forward* is used.

9.4.1 Recursive Forward Dynamics Algorithm

Conventionally, joint accelerations are solved from Eq. (9.10a) using the Cholesky decomposition of the GIM $\mathbf{I}$, as done by Walker and Orin (1982), Angeles (2003), and others, or using the MATLAB command "$\mathbf{I}\backslash\boldsymbol{\phi}$" (WR-Matlab), where $\boldsymbol{\phi}$ represents the vector of generalized forces due to external moments and forces, gravity and Coriolis terms, etc. The above approach requires order (n^3)—n being the number of links or joints in the robot—computations, and produces nonsmooth joint accelerations (Ascher et al., 1997). On the contrary, the dynamic formulation based on the Decoupled Natural Orthogonal Complement (DeNOC) matrices presented in this chapter allows one to solve $\ddot{\boldsymbol{\theta}}$ from Eq. (9.10a) recursively with order (n) computational complexity (Saha, 1999; 2003). Such recursive algorithms are known to provide smooth profiles for $\ddot{\boldsymbol{\theta}}$, as reported in Ascher at al. (1997), Mohan and Saha (2007), Shah et al. (2013), and others.

In this section, a recursive order (n) forward dynamics algorithm is presented, which requires the following inputs: For i =1, …, n,

1. DH parameters, and the mass and inertia properties of all links, as they are shown in Tables 9.2–9.4.
2. Initial values for the variable DH parameters, i.e.,θ_i, for a revolute joint, and b_i, for a prismatic joint, and their first time derivatives, i.e., $\dot{\theta}_i$ s and $\dot{b}_i$s.
3. Time history of the input joint forces/torques, i.e., τ_i.

4. Each component of the vector, $\boldsymbol{\phi} \equiv \boldsymbol{\tau} - \mathbf{h}$, obtained from Eq. (9.10a), i.e., ϕ_i, which is to be calculated recursively using an inverse dynamics algorithm, e.g., the one given in Section 9.3 while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$.

The recursive forward dynamics algorithm presented here is based on the $\mathbf{UDU}^T$ decomposition of the generalized inertia matrix, $\mathbf{I}$ of Eq. (9.10a), i.e., $\mathbf{I} = \mathbf{UDU}^T$, where $\mathbf{U}$ and $\mathbf{D}$ are the upper triangular and diagonal matrices, respectively. Moreover, substituting $\mathbf{I} = \mathbf{UDU}^T$ and $\boldsymbol{\varphi} \equiv \boldsymbol{\tau} - \mathbf{h}$ in Eq. (9.10a), one obtains

$$\mathbf{UDU}^T\ddot{\boldsymbol{\theta}} \equiv \boldsymbol{\phi} \tag{9.38}$$

A three step recursive scheme is then used to calculate the joint accelerations from the above equations, i.e.,

1. Solution for $\hat{\boldsymbol{\phi}}$: The solution $\hat{\boldsymbol{\phi}} = \mathbf{U}^{-1}\boldsymbol{\phi}$ is evaluated in terms of the scalar terms as

$$\hat{\phi}_i = \phi_i - \mathbf{p}_i^T \boldsymbol{\eta}_{i,i+1}, \text{ for } i = n, \ldots, 1 \tag{9.39a}$$

Note $\hat{\phi}_n \equiv \phi_n$, and the 6-dimensional vector $\boldsymbol{\eta}_{i,i+1}$ is obtained as

$$\boldsymbol{\eta}_{i,i+1} \equiv \mathbf{B}^T_{i+1,i}\,\boldsymbol{\eta}_{i+1} \text{ and } \boldsymbol{\eta}_{i+1} \equiv \boldsymbol{\psi}_{i+1}\hat{\phi}_{i+1} + \boldsymbol{\eta}_{i+1,i+2} \tag{9.39b}$$

in which $\boldsymbol{\eta}_{n,n+1} = \mathbf{0}$. The new variable $\boldsymbol{\psi}_{i+1}$ is the 6×6 matrix which is evaluated using

$$\boldsymbol{\psi}_i \equiv \frac{\hat{\boldsymbol{\psi}}_i}{\hat{m}_i}, \text{ where } \hat{\boldsymbol{\psi}}_i \equiv \hat{\mathbf{M}}_i\mathbf{p}_i \text{ and } \hat{m}_i \equiv \mathbf{p}_i^T\hat{\boldsymbol{\psi}}_i \tag{9.39c}$$

In Eq. (9.39c), the 6×6 matrix $\hat{\mathbf{M}}_i$ called the *articulated body inertia* (Saha, 1997; 1999) that can be obtained recursively, similar to $\tilde{\mathbf{M}}_i$ in Eq. (9.11d), as

$$\hat{\mathbf{M}}_i \equiv \mathbf{M}_i + \mathbf{B}^T_{i+1,i}\,\bar{\mathbf{M}}_{i+1}\mathbf{B}_{i+1,i}, \text{ where } \bar{\mathbf{M}}_{i+1} \equiv \hat{\mathbf{M}}_{i+1} - \hat{\boldsymbol{\psi}}_{i+1}\boldsymbol{\psi}^T_{i+1} \tag{9.39d}$$

for $i = n–1, \ldots, 1$, and $\hat{\mathbf{M}}_n = \mathbf{M}_n$.

2. Solution for $\tilde{\boldsymbol{\phi}}$: The solution $\tilde{\boldsymbol{\phi}} = \mathbf{D}^{-1}\hat{\boldsymbol{\phi}}$ involves the inverse of the diagonal matrix, $\mathbf{D}$ of Eq. (9.38), which is simple. The inverse $\mathbf{D}^{-1}$ has only nonzero diagonal elements that are the reciprocal of the corresponding diagonal elements of $\mathbf{D}$. Vector $\tilde{\boldsymbol{\phi}}$ is obtained as follows: For $i = 1, \ldots, n$,

$$\tilde{\varphi}_i = \hat{\phi}_i / \hat{m}_i \tag{9.40}$$

The scalar $\hat{m}_i$ is defined in Eq.(9.39c).

3. Solution for $\ddot{\boldsymbol{\theta}}$: In this step, $\ddot{\boldsymbol{\theta}} = \mathbf{U}^{-T}\tilde{\boldsymbol{\phi}}$ is calculated for $i = 2, \ldots, n$ as

$$\ddot{\theta}_i = \tilde{\varphi}_i - \hat{\boldsymbol{\psi}}_i^T\tilde{\boldsymbol{\mu}}_{i,i-1} \tag{9.41a}$$

where $\ddot{\theta}_1 \equiv \tilde{\phi}_1$, and the 6-dimensional vector $\tilde{\boldsymbol{\mu}}_{i,i-1}$ is obtained as

$$\tilde{\boldsymbol{\mu}}_{i,i-1} \equiv \mathbf{B}_{i,i-1}\tilde{\boldsymbol{\mu}}_{i-1} \text{ and } \tilde{\boldsymbol{\mu}}_{i-1} \equiv \mathbf{p}_{i-1}\ddot{\theta}_{i-1} + \tilde{\boldsymbol{\mu}}_{i-1,i-2} \tag{9.41b}$$

in which $\tilde{\boldsymbol{\mu}}_{10} \equiv 0$. The above forward dynamics algorithm (Saha, 1997; 2003) is the basis for ReDySim (Shah et al., 2013) which can also take care of the tree-type systems like biped and walking robots with one-, two-, and three-DOF joints, i.e., revolute, universal and spherical joints, respectively. Table 9.6 shows the comparison of various forward dynamics algorithms, along with the one for ReDySim which was

adopted in RoboAnalyzer software. Typically, recursive dynamics algorithms are known to be computationally efficient when $n \geq 10$ or 12. However, the ReDySim-based algorithm adopted in RoboAnalyzer is efficient even for $n \geq 7$.

It is pointed out here that the use of the DeNOC matrices for forward dynamics is more advantageous in comparison to inverse dynamics algorithm, as evident from Tables 9.1 and 9.6. Besides, as shown in Agrawal (2013), the ReDySim-based forward dynamics algorithm is extremely stable numerically for the reason explained in the paper.

9.4.2 Simulation

Simulation consists of forward dynamics to find the joint acceleration $\ddot{\boldsymbol{\theta}}$ followed by its integration to obtain $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ for a given set of initial conditions, i.e., $\dot{\boldsymbol{\theta}}(0)$ and $\boldsymbol{\theta}(0)$. As pointed out in Chapter 8 that except in extremely simple cases the integration needs to be done numerically either using well-established methods like Runge–Kutta–Fehlberg, Adams–Bashforth–Moulton and others (Shampine, 1994), or MATLAB commands like "ODE45", etc. (WR-Matlab). In this section, simulation results were generated using RoboAnalyzer where a computer program in C# was written for the numerical integration using Runge–Kutta–Fehlberg formula.

Table 9.6 Computational complexities for forward dynamics

Algorithm	Multiplications/ Divisions (M)	Additions/ Subtractions (A)	$n = 6$	$n = 7$	$n = 10$
ReDySim (Shah et al., 2013)	**$135n - 116$**	**$131n - 123$**	694M 663A	829M 794A	1234M 1187A
Saha (2003)	$191n - 284$	$187n - 325$	862M 797A	1053M 984A	1626M 1545A
Featherstone (1983)	$199n - 198$	$174n - 173$	996M 871A	1195M 1045A	1792M 1527A
Stejskal and Valasek (1996)	$226n - 343$	$206n - 345$	1013M 891A	1239M 1097A	1917M 1715A
Brandl et al. (1988)	$250n - 222$	$220n - 198$	1278M 1122A	1528M 1342A	2278M 2002A
Walker and Orin (1982)	$n^3/6 + 23\,n^2/2 + 115n/3 - 47$	$n^3/6 + 7n^2 + 233n/3 - 46$	633M 480A	842M 898A	1653M 1209A

An important aspect of smooth joint acceleration profiles available from a recursive forward dynamics algorithm, as mentioned in Section 9.4.1, is elaborated here. When the joint acceleration profile for $\ddot{\boldsymbol{\theta}}$ is not smooth convergence of its numerical integration to obtain the joint velocity and position, i.e., $\dot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$, is slow. Alternatively, with smooth $\ddot{\boldsymbol{\theta}}$ profile obtained from a recursive forward dynamics algorithm convergence of the numerical integration results is must faster. Hence, the overall CPU time required for the forward dynamics and numerical integration together using the recursive algorithm may be smaller even for $n = 6$ compared to an order (n^3) algorithm which requires less forward dynamics computations when $n = 6$ (see Table 9.6) but may require more time to perform the numerical integration. This aspect was proven by Mohan and Saha (2007b).

Example 9.8 Simulation of the Three-DOF Planar Arm

The three-DOF arm shown in Fig. 9.6 is considered here to let fall freely under gravity from the horizontal initial configuration with no initial motion, i.e., $\theta_i(0) = \dot{\theta}_i(0) = 0$, for $i = 1, 2, 3$. Time step ΔT for the numerical integration was taken as $\Delta T = 0.01$ second. The results for the joint positions, namely, the variations of θ_1, θ_2, and θ_3 with time are shown in Fig. 9.16. Note from Fig. 9.6 that due to the gravity the first joint angle θ_1 will increase initially in the negative direction. This is evident from Fig. 9.16. Moreover, the system under gravity behaves as a three-link pendulum, which is clear from all the joint-angle variations of Fig. 9.16.

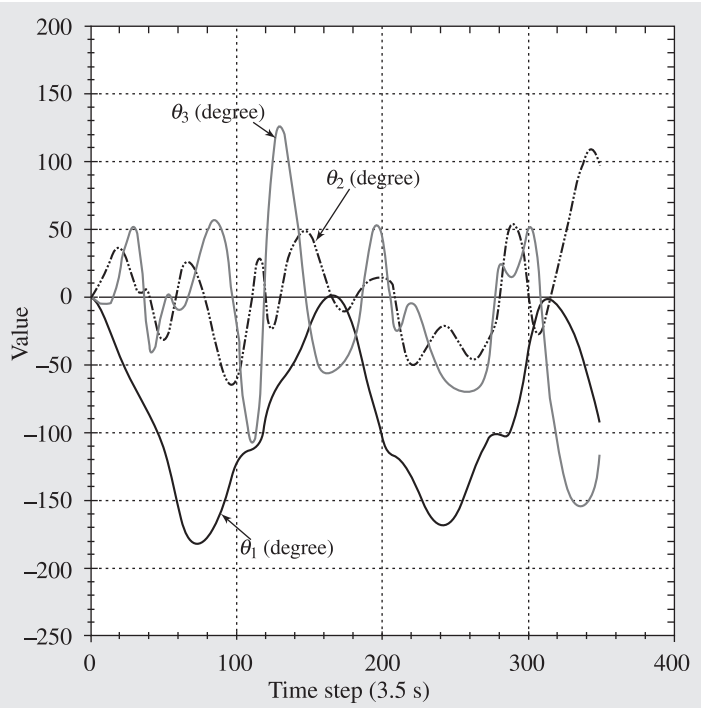


Fig. 9.16 Joint angles for the three-DOF arm

Example 9.9 Simulation of the PUMA Architecture

The robot with PUMA architecture is shown in Fig. 9.10. Its free-fall simulation, i.e., the robot was let fall freely under gravity, was carried out using RoboAnalyzer with the initial conditions of $\theta_i(0) = \dot{\theta}_i(0) = 0$, for $i = 1, \ldots, 6$. The time step $\Delta T = 0.01$ second was taken for the numerical integration. Variations of the joint angles versus time are shown in Figs. 9.17(a–f). It is clear from Fig. 9.10 that due to the length $a_3 = 0.02$, joint 2 will rotate in the positive direction, which is evident from Fig. 9.17(b).

What is "Free?"

The joints are free without any applied torques.

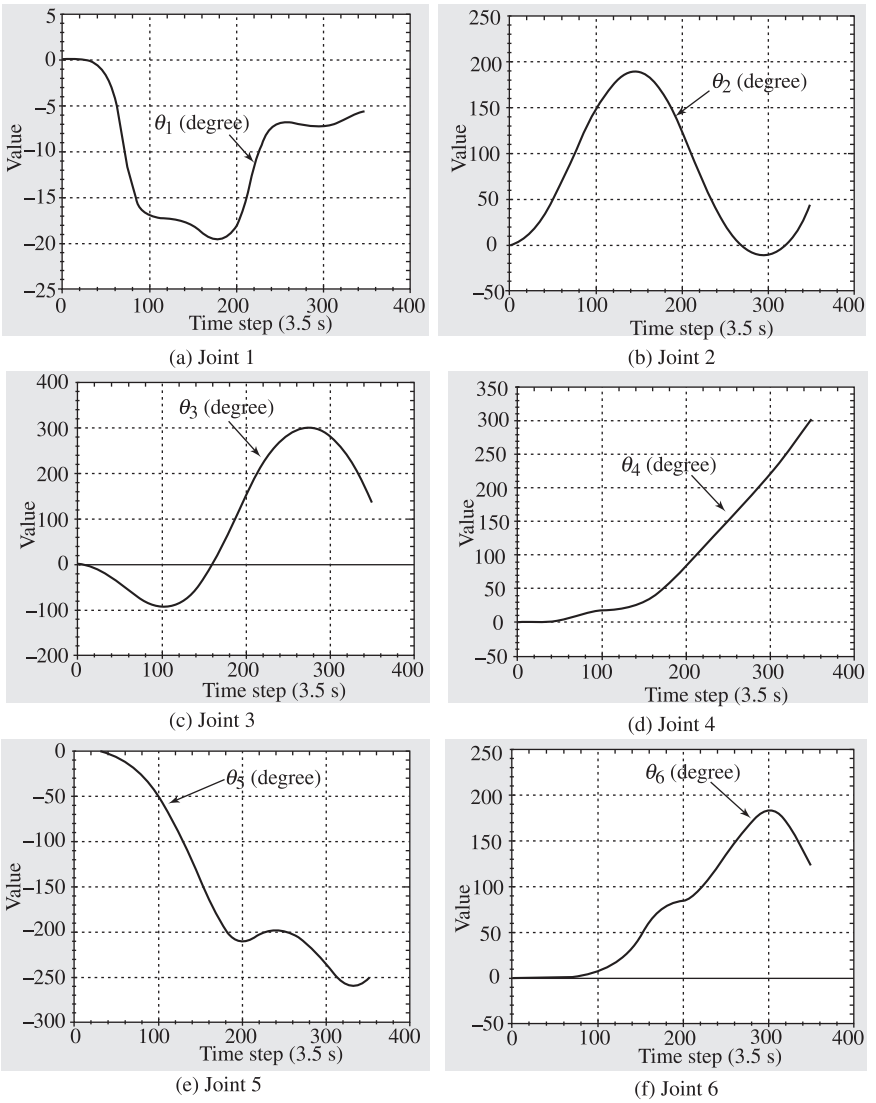


Fig. 9.17 Joint angles for the PUMA architecture

Example 9.10 Simulation of the Stanford Arm

Free-fall simulation of the Stanford arm shown in Fig. 9.12 was carried out with the following initial values of the joint positions and rates:

$$\theta_1(0) = 0,\ \theta_2(0) = -\pi/2,\ b_3\ (0) = 0.635 \text{ m},\ \theta_4(0) = 0,\ \theta_5(0) = \theta_6(0) = \pi$$

$$\dot{\theta}_i\,(0) = 0, \text{ for } i = 1, 2, 4, 5, 6, \text{ and } \dot{b}_3(0) = 0$$

Time step ΔT for numerical integration was taken same as before, i.e., $\Delta T = 0.01$ second. The joint position results obtained from RoboAnalyzer are then shown in Figs. 9.18(a–f). Since the initial configuration of the Stanford arm given in

Table 9.4(a), the motion of Joint 3 should increase sharply after one second when Joint 2 turns more than 90°. This is evident from Figs. 9.18(b) and (c), respectively. Once can visualize the same in the animation of RoboAnalyzer software.

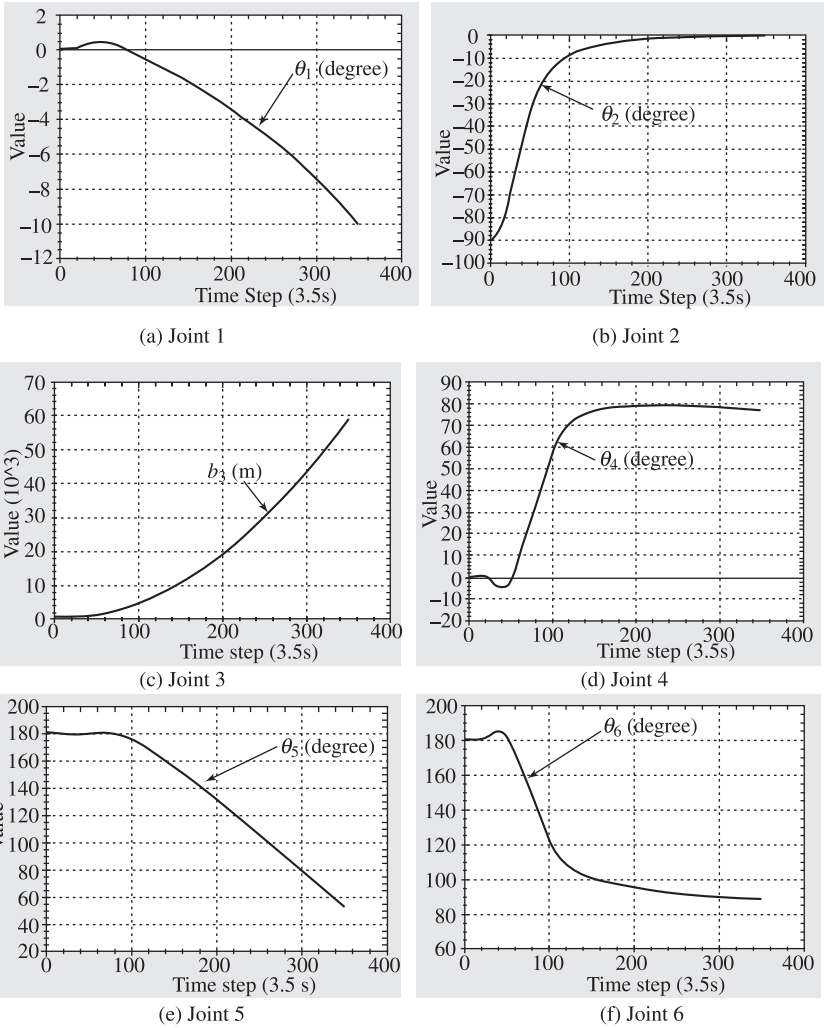


Fig. 9.18 Joint angles and displacement for the Stanford arm

Example 9.11 Simulation of the KUKA KR-5

The industrial robot considered for inverse dynamics analysis in Example 9.7 is considered here for free-fall simulation. The initial joint configuration is shown in Table 9.5. While variation of the joint angles are shown in Figs. 9.19(a-f), an intermediate configuration during the animation in RoboAnalyzer software is shown in Fig. 9.20. If one looks at the animation screenshot of Fig. 9.20 carefully, one can notice that the robot links interfere during the motion which is actually not

permissible. Hence, such simulation tool allows a motion planner to decide the range of motion be allowed during actual motion of the robot.

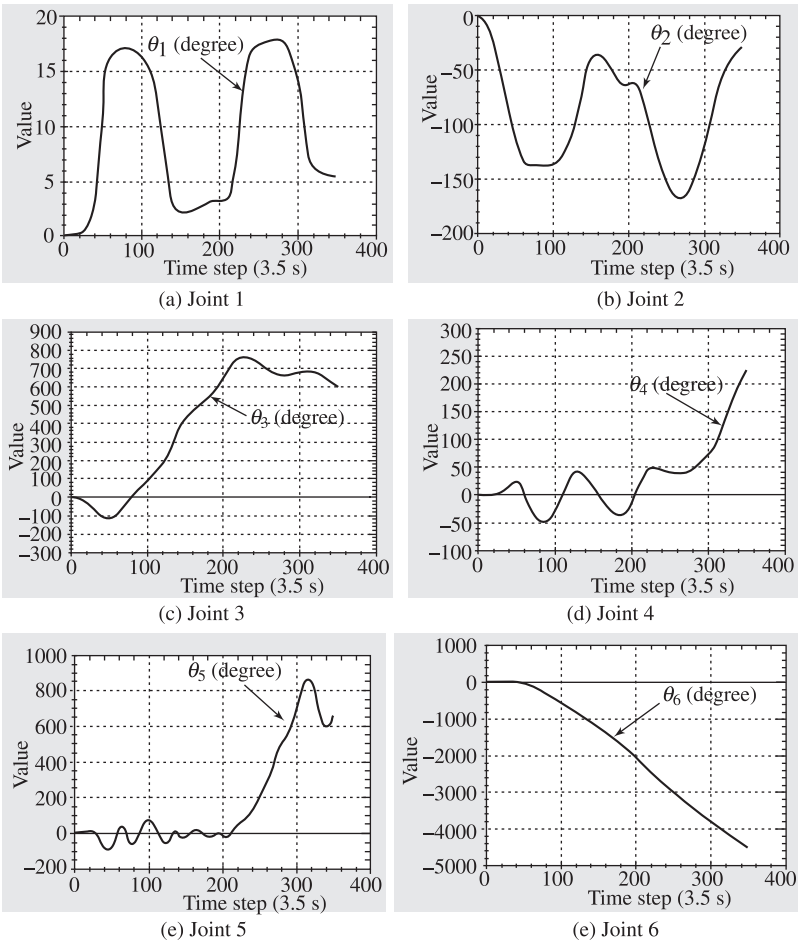


Fig. 9.19 Joint angles for KUKA KR-5 robot

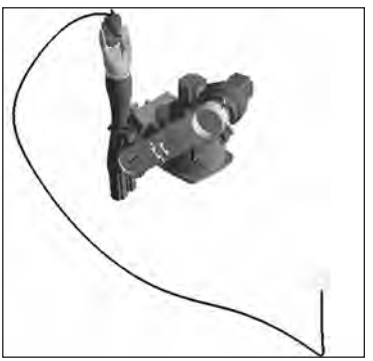


Fig. 9.20 RoboAnalyzer screenshot during free-fall simulation of KUKA KR-5

SUMMARY

In this chapter, dynamic modeling of serial robots using the Decoupled Natural Orthogonal Complements (DeNOC) is presented. Recursive inverse and forward dynamics algorithms are presented for robot control and simulation, respectively. Computational complexities of both the algorithms are reported with illustrative results for planar and spatial robotic systems using an in-house developed software called RoboAnalyzer.

EXERCISES

9.1 What is inverse dynamics?

9.2 Define forward dynamics and its difference in context with simulation.

9.3 Why the concept of the DeNOC matrices is preferable over other dynamic modeling approaches?

9.4 What is the meaning of *orthogonal* in DeNOC?

9.5 For forward dynamics using the DeNOC matrices, for how many links in a robot is the algorithm more efficient?

9.6 Find the expression of kinetic energy of one-link arm, Eq. (8.56a), using the DeNOC matrices.

9.7 Derive the equation of motion of one-link arm, Fig. 9.4, using the matrix and vector expressions of the NOC matrix **N** appearing after Eq. (9.10a).

9.8 Write the expression of the total kinetic energy for an *n*-link robot manipulator using the definition of the generalized twist **t** given in Eq. (9.3c).

9.9 Redo Exercise 9.7 using the definition of the generalized joint-rate $\dot{\boldsymbol{\theta}}$ of Eq. (9.3c).

9.10 What is $\mathbf{UDU}^T$ and what does it perform?

9.11 Generate joint torque plots for the Example 9.4 using the following explicit expressions:

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + \mathbf{C}\dot{\boldsymbol{\theta}} = \boldsymbol{\tau}^g + \boldsymbol{\tau}, \text{ where } \boldsymbol{\theta} \equiv [\theta_1 \quad \theta_2 \quad \theta_3]^T \tag{9.42a}$$

where the 3 × 3 GIM and MCI matrices **I** and **C**, respectively, and the 3-dimensional vector $\boldsymbol{\tau}^g$ due to gravity are given by

$$\begin{aligned}
i_{11} &= \frac{1}{3}(m_1a_1^2 + m_2a_2^2 + m_3a_3^2) + m_2a_1^2 + m_3(a_1^2 + a_2^2) + (m_2 + 2m_3)a_1a_2c_2 \\
&\quad + m_3a_3(a_2c_3 + a_1c_{23}) \\
i_{12} &= i_{21} = \frac{1}{3}(m_2a_2^2 + m_3a_3^2) + m_3a_2^2 + \left(\frac{1}{2}m_2a_2 + m_3a_2\right)a_1c_2 \\
&\quad + \frac{1}{2}m_3a_3(2a_2c_3 + a_1c_{23}) \\
i_{13} &= i_{31} = \frac{1}{3}m_3a_3^2 + \frac{1}{2}m_3a_3(a_2c_3 + a_1c_{23}) \\
i_{22} &= \frac{1}{3}(m_2a_2^2 + m_3a_3^2) + m_3(a_2^2 + a_2a_3c_3) \\
i_{23} &= i_{32} = \frac{1}{3}m_3a_3^2 + \frac{1}{2}m_3a_2a_3c_3; \qquad i_{33} = \frac{1}{3}m_3a_3^2
\end{aligned} \tag{9.42b}$$

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

$$c_{12} = -\frac{1}{2}[\{m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})\}\dot{\theta}_{12} + m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_3]$$

$$c_{13} = -\frac{1}{2} m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_{123}$$

$$c_{21} = \frac{1}{2}[\{m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})\}\dot{\theta}_1 - m_3 a_2 a_3 s_3 \dot{\theta}_3]$$

$$c_{22} = -\frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_3; \qquad c_{23} = -\frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_{123}$$

$$c_{31} = \frac{1}{2} m_3[(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1 + a_2 a_3 s_3 \dot{\theta}_2]; \; c_{32} = \frac{1}{2} m_3 a_2 a_3 s_3 \dot{\theta}_{12}$$

$$c_{33} = 0 \tag{9.42c}$$

and

$$\tau_1^g = \frac{1}{2} g[-m_1 a_1 c_1 - 2m_2\left(a_1 c_1 + \frac{1}{2} a_2 c_{12}\right) - 2m_3\left(a_1 c_1 + a_2 c_{12} + \frac{1}{2} a_3 c_{123}\right)$$

$$\tau_2^g = \frac{1}{2} g\left[-m_2 a_2 c_{12} - 2m_3\left(a_2 c_{12} + \frac{1}{2} a_3 c_{123}\right)\right]$$

$$\tau_3^g = \frac{1}{2} g[-m_3 a_3 c_{123}] \tag{9.42d}$$

where $\theta_{12} \equiv \theta_1 + \theta_2$; $\theta_{123} \equiv \theta_{12} + \theta_3$; $\dot{\theta}_{12} \equiv \dot{\theta}_1 + \dot{\theta}_2; \dot{\theta}_{123} \equiv \dot{\theta}_{12} + \dot{\theta}_3$; and $s(.) \equiv \sin(.)$, $c(.) \equiv \cos(.)$. Find the joint actuator torques, τ, for plotting.

9.12 Verify the simulation results of Example 9.8 using the expressions of Eq. (9.42) and "ODE45" function of MATLAB.

9.13 What are the features of RoboAnalyzer (RA)?

9.14 Using the RA software find the joint torques for the two-DOF planar manipulator shown in Fig. 9.5.

9.15 Generate joint torques using RA for the three-DOF manipulator of Fig. 9.6 for the joint trajectory given by Eqs. (9.36a-c) with the following inputs:

$$\theta_i(0) = \dot{\theta}_i(0) = \ddot{\theta}_i(0) = 0; \; \theta_i(T) = \pi/2, \; \dot{\theta}_i(0) = \ddot{\theta}_i(0) = 0, \text{ for } i = 1,2,3 \tag{9.43}$$