8
Improved Formulations for Real-Time Dynamics

The general purpose dynamic formulations described in Chapter 5 are simple and efficient, but they are not suitable for real time dynamic simulation. Real time performance requires faster formulations. These can be developed by taking into account the system's kinematic configuration or topology. In the last two decades, a big effort has been dedicated to developing very efficient dynamic formulations for serial robots or manipulators. These formulations have been extended later on to general open and closed chain configurations.

In the first section of this chapter, a survey of two of the most efficient available formulations that need $O(N^3)$ and $O(N)$ arithmetic operations is presented. It follows, a detailed description of a formulation based on *velocity transformations* that can be parallelized at body level. In the next section, a description of how the penalty formulation can be used for improved performance is also included. Finally, two complex examples: a model of the human body and a heavy truck, with theoretical count of arithmetic floating-point operations, and some numerical results are presented.

The methods and results presented in this chapter are contributions coming mainly from Jiménez (1993), and Avello et al. (1993).

8.1 Survey of Improved Dynamic Formulations

Most of the improvements in multibody dynamic formulations that have been developed in the last 25 years come from the robotics field. This field was very active in the 60's and 70's with scientists trying to solve very hard problems of simulation and control with the limited computational resources available at that time. Anyone familiar with robot control algorithms based on dynamic models knows the very competitive race that took place in order to decrease the number of arithmetic operations required for the inverse dynamics in serial robots. This research was quite important and led to the solution of the inverse dynamic problem two or three hundred times per second with a DEC-PDP 11 processor. The recursive Newton-Euler formulation seems to have been the winner of the race

(Luh, Walker, and Paul (1980)), at least for general geometries of serial manipulators with about six degrees of freedom.

Inverse dynamics, which consists in computing the motor torques and/or forces required to produce a desired motion, was directly interrelated to control purposes. However, it was realized that the high efficiency reached in solving the inverse dynamics could also benefit the formulation of the simulation problem, That is, the computation of the accelerations from the state variables (position and velocities) and the external and driving forces. As it may be seen in Walker and Orin (1982), the solution of the inverse dynamics allows for a formulation of the equations of motion much more efficient than conventional or even recursive Euler-Lagrange formulations.

Other authors, such as Armstrong (1979) and Featherstone (1983, 1987) have developed fully recursive *O(N)* algorithms for open-chain systems, that is, algorithms whose number of floating-point arithmetic operations grows linearly with the number of degrees of freedom or bodies in the open-chain. The algorithms of Walker and Orin (1982) conclude with the solution of a system of N linear equations. They are of order $O(N^3)$, if Gaussian elimination is used.

Although it has been demonstrated (Featherstone (1987)) that the best $O(N^3)$ algorithms are better and faster than the best *O(N)* algorithms for $N<10$ (that is, for most of the practical cases), the elegance and attractiveness of the Featherstone's *O(N)* formulation has exerted a strong influence on later developments that have generalized these ideas for non-serial (tree-configuration) systems (Bae and Haug (1987)) and closed-loop systems (Bae and Haug (1987-88), Rodriguez et al. (1991)). A limitation arises when closed-chain multibodies are analyzed, since for these cases special provisions must be made to account for the reaction forces between the different loops. More interest has been placed recently in looking for improved efficiency using $O(N^3)$ methods in cases with small values of N per chain (Bae, Hwang and Haug (1988), Bae and Won (1990)).

The foundations of the $O(N^3)$ formulation of Walker and Orin (1982), the *O(N)* formulation of Featherstone (1987), and the generalizations of Bae and Haug (1987-88) will be described in the following subsections . Since it is not possible to reproduce these formulations in detail, the fundamental ideas on which each formulation is based will be outlined. Those interested in further details are referred to the original references given at the end of the chapter.

The number of floating-point arithmetic operations has been the criterion followed to compare the efficiency of different dynamic algorithms. From the *60's* to the late *80's*, this seemed to be the most reasonable criterion, since the average computers required over 20 times more *CPU* time to carry out a *SP* or a *DP* floating-point operation than an integer operation. Thus, it made full sense to neglect integer or logical operations and to draw comparative conclusions only from the theoretical account of floating-point operations. This procedure does not make sense any longer, since the RISC computers of the 90's have similar *CPU* cost for integer and floating-point operations. The new RISC processors can carry out two double precision floating-point operations (a product and an accu-

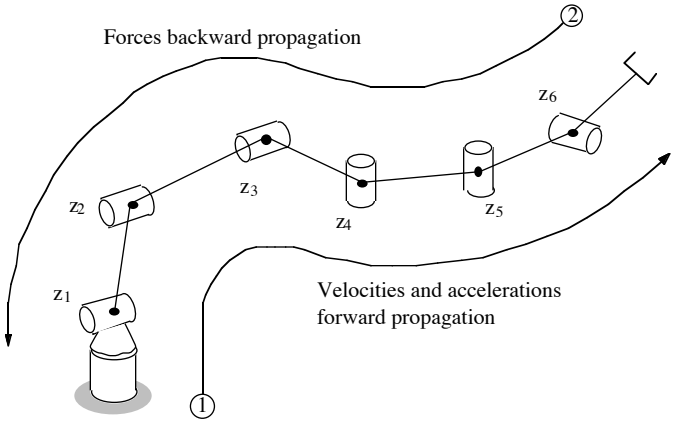


Figure 8.1. Recursive Newton-Euler method.

mulative addition) for each clock cycle. In these new machines, a clear and neat logic of the algorithm and an efficient use of the registers and of the cache memory can be more important than the number of floating-point operations. It will be seen that the methods presented in Sections 8.2 and 8.3 are simpler than other methods found in the literature. Moreover, these methods offer excellent opportunities to get very efficient computer implementations on modern computers using high level languages.

8.1.1 Formulations O(N³): Composite Inertia

One of the most efficient dynamic formulation for serial robots with $N<10$ is the one based on the recursive Newton-Euler solution of the inverse dynamic problem (Walker and Orin (1982)). This formulation proceeds as indicated below.

The inverse dynamics consists in determining the vector $\boldsymbol{\tau}$ of the N motor torques and/or forces from the external forces $\mathbf{Q}_{ex}$ and the joint positions $\mathbf{z}$, velocities $\dot{\mathbf{z}}$, and accelerations $\ddot{\mathbf{z}}$. Symbolically, it can be written

$$\boldsymbol{\tau} = \mathbf{tau}(\mathbf{Q}_{ex}, \mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}) \tag{8.1}$$

where **tau**(**–**) represents a function that solves the inverse dynamics for the values of the input parameters.

The recursive Newton-Euler method proceeds as indicated in Figure 8.1. In the first step, the joint positions, velocities, and accelerations are recursively propagated forward to compute the absolute link position, velocity, and acceleration and from them the inertia forces. In the second step, inertia and external forces are recursively propagated backwards so as to compute the equilibrating actuator forces and torques. If local reference frames are properly chosen on each body and used for the recursive equations, one can further reduce the computational cost.

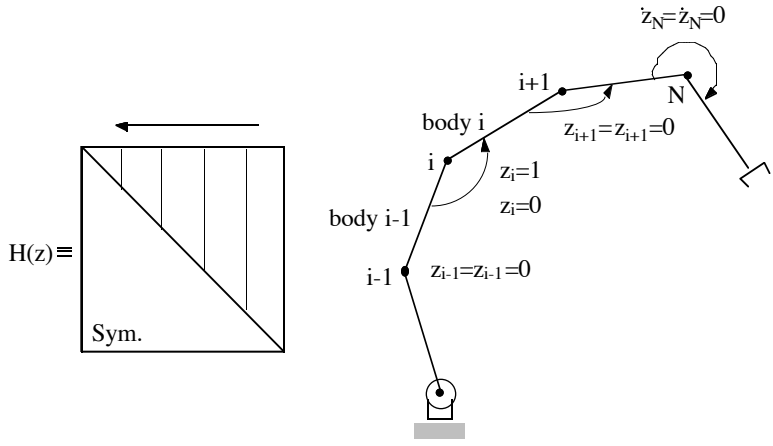


Figure 8.2. Composite-Inertia method.

Now it will be described how the inverse dynamic solution given by equation (8.1) can be used for direct dynamic simulation following Walker and Orin (1982). The equations of motion can be written in the form,

$$\mathbf{M}(\mathbf{z})\,\ddot{\mathbf{z}} + \mathbf{C}(\mathbf{z}, \dot{\mathbf{z}})\,\dot{\mathbf{z}} + \mathbf{G}(\mathbf{z}) + \mathbf{Q}(\mathbf{z}, \mathbf{Q}_{ex}) = \boldsymbol{\tau} \tag{8.2}$$

where $\mathbf{M}(\mathbf{z})$ is the position-dependent inertia matrix, $\mathbf{C}(\mathbf{z}, \dot{\mathbf{z}})\,\dot{\mathbf{z}}$ is a term representing velocity-dependent (centrifugal and Coriolis) inertia forces, $\mathbf{G}(\mathbf{z})$ represents the effect of gravitational forces, and $\mathbf{Q}(\mathbf{z}, \mathbf{Q}_{ex})$ represents the vector of generalized external forces. Remember that in the direct dynamic problem, everything is known except the accelerations $\ddot{\mathbf{z}}$.

If a vector $\mathbf{b}$ is defined as

$$\mathbf{b} \equiv \mathbf{C}(\mathbf{z}, \dot{\mathbf{z}})\,\dot{\mathbf{z}} + \mathbf{G}(\mathbf{z}) + \mathbf{Q}(\mathbf{z}, \mathbf{Q}_{ex}) \tag{8.3}$$

equation (8.2) can be written as

$$\mathbf{M}(\mathbf{z})\,\ddot{\mathbf{z}} = \boldsymbol{\tau} - \mathbf{b} \tag{8.4}$$

Now, all the terms in equation (8.4) can be computed with the inverse dynamics function **tau**(**–**) defined in (8.1):

– From equations (8.2) and (8.3), $\mathbf{b}$ can be computed by introducing a null acceleration in equation (8.1)

$$\mathbf{b} = \mathbf{tau}(\mathbf{Q}_{ex}, \mathbf{z}, \dot{\mathbf{z}}, 0) \tag{8.5}$$

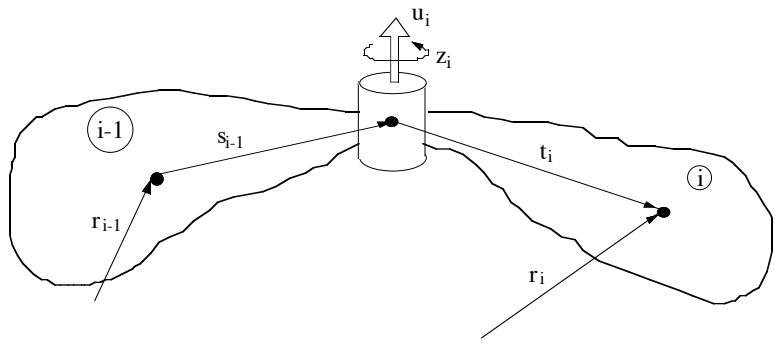


Figure 8.3. Revolute joint between contiguous bodies.

– From equation (8.4) it can be seen that the column *i* of **M**(**z**) can be obtained with an acceleration vector $\mathbf{e}^i$ (all elements zeroes, except for a unit value on component *i*),

$$\mathbf{m}_i(\mathbf{z}) = \mathbf{tau}(0, \mathbf{z}, 0, \mathbf{e}^i) \qquad i = 1, 2, ..., N \tag{8.6}$$

– Once all the columns of **M**(**z**) and vector **b** have been computed, the actual acceleration vector $\ddot{\mathbf{z}}$ can be obtained by solving the linear system of equations (8.4).

This algorithm can be improved if one takes into account that the matrix **M**(**z**) is symmetric and only the upper (or the lower) part needs to be calculated. The most efficient algorithm is the so-called *composite inertia* method (Walker and Orin (1982)). This method computes only the terms of **M**(**z**) located over the diagonal, starting from column *N* and proceeding to column 1 (See Figure 8.2a). In order to compute column *i* use equation (8.6). This equation computes the torques and/or forces in joints (*1, 2, ..., i*) with null velocities and null accelerations in all the joints, except in joint *i*, where $\ddot{z}_i = 1$ (See Figure 8.2b).

At the time of computing column *i,* the joints from *(i+1)* to *N* have zero velocity and zero acceleration. Furthermore, for all $j<i$, that is, in all subsequent calls to function **tau**(**–**), both the velocity and acceleration of joints *(i+1), (i+2), ... , N* remain zero. Therefore, bodies *i* to *N* do not exhibit relative velocities and accelerations and move as a single rigid body. Walker and Orin's idea (1982) makes use of this property and computes the *composite inertia*, or inertia of a fictitious rigid body, with the same center of gravity and inertia tensor as the set of rigid bodies *i, (i+1), ... , N*. The composite inertia is updated before each call to function **tau**(**–**). For instance, before computing column *(i–1)*, the inertia of body *i* is added to the composite inertia. The complexity of expressions in this method can be considerably reduced by choosing appropriate local (body-fixed) reference frames. So far, the composite inertia method is the most efficient general purpose algorithm for serial manipulators with $N<10$ which includes most practical cases.

8.1.2 Formulations O(N): Articulated Inertia

This method has been fully developed by Featherstone (1983, 1987) and has had a major influence on later works, in spite of being less efficient than the composite inertia method for $N<9$. Featherstone described this method using *spatial vector* notation for both the kinematic and dynamic equations. This notation yields simpler, more compact and more efficient expressions, but many engineers and students are not familiar with it. The main ideas behind Featherstone's formulation will be explained employing a more conventional notation used by Bae and Haug (1987).

Two contiguous elements, $(i-1)$ and i, linked by joint i can be seen in Figure 8.3. It can be assumed that joint i is of revolute type. The following kinematic relationships can be written for the rotation matrices:

$$\mathbf{A}_i = \mathbf{A}_{i,i-1}(\mathbf{u}_i, \mathbf{z}_i) \cdot \mathbf{A}_{i-1} \tag{8.7}$$

for angular velocities:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{z}_i \, \mathbf{u}_i \tag{8.8}$$

and for the velocities of the centers of gravity:

$$\begin{aligned} \dot{\mathbf{r}}_i &= \dot{\mathbf{r}}_{i-1} + \tilde{\boldsymbol{\omega}}_{i-1} \mathbf{s}_{i-1} + \tilde{\boldsymbol{\omega}}_i \mathbf{t}_i = \\ &= \dot{\mathbf{r}}_{i-1} + \tilde{\boldsymbol{\omega}}_{i-1} \mathbf{s}_{i-1} + \left(\tilde{\boldsymbol{\omega}}_{i-1} + \dot{z}_i \tilde{\mathbf{u}}_i\right) \mathbf{t}_i = \\ &= \dot{\mathbf{r}}_{i-1} + \tilde{\boldsymbol{\omega}}_{i-1} \left(\mathbf{s}_{i-1} + \mathbf{t}_i\right) + \dot{z}_i \tilde{\mathbf{u}}_i \mathbf{t}_i = \\ &= \dot{\mathbf{r}}_{i-1} + \tilde{\boldsymbol{\omega}}_{i-1} \left(\mathbf{r}_i - \mathbf{r}_{i-1}\right) + \dot{z}_i \tilde{\mathbf{u}}_i \mathbf{t}_i \end{aligned} \tag{8.9}$$

Equations (8.8) and (8.9) can be written together in the following matrix form:

$$\{\mathbf{Y}_i\} = \begin{Bmatrix} \dot{\mathbf{r}}_i \\ \boldsymbol{\omega}_i \end{Bmatrix} = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{r}}_{i-1} - \tilde{\mathbf{r}}_i \\ 0 & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{r}}_{i-1} \\ \boldsymbol{\omega}_{i-1} \end{Bmatrix} + \begin{Bmatrix} \tilde{\mathbf{u}}_i \mathbf{t}_i \\ \mathbf{u}_i \end{Bmatrix} \dot{z}_i \tag{8.10}$$

or, in compact form:

$$\mathbf{Y}_i = \mathbf{B}_i \mathbf{Y}_{i-1} + \mathbf{b}_i \dot{z}_i \tag{8.11}$$

Equation (8.11) is a recursive relation between the velocities of two consecutive bodies, in terms of the relative joint velocity. Matrix $\mathbf{B}_i$ and vector $\mathbf{b}_i$ depend on the position variables. Differentiating this equation one obtains

$$\dot{\mathbf{Y}}_i = \mathbf{B}_i \dot{\mathbf{Y}}_{i-1} + \mathbf{b}_i \ddot{z}_i + \mathbf{d}_i \tag{8.12}$$

where $\mathbf{d}_i$ is a vector that groups the velocity-dependent terms

$$\mathbf{d}_i = \dot{\mathbf{B}}_i \mathbf{Y}_{i-1} + \dot{\mathbf{b}}_i \dot{z}_i \tag{8.13}$$

Now, for an *N*-link serial manipulator, it is possible to formulate the principle of virtual power in the form

$$\sum_{i=1}^{N} \mathbf{Y}_i^{*T} \left(\mathbf{M}_i \dot{\mathbf{Y}}_i - \mathbf{Q}_i \right) = 0 \tag{8.14}$$

where

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{m}_i \mathbf{I} & 0 \\ 0 & \mathbf{J}_i \end{bmatrix} \tag{8.15}$$

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{n}_i - \tilde{\boldsymbol{\omega}}_i \mathbf{J}_i \boldsymbol{\omega}_i \end{bmatrix} \tag{8.16}$$

and where $\mathbf{n}_i$ and $\mathbf{f}_i$ are the external torques and forces acting at the center of gravity of link *i*. Vector $\mathbf{Y}_i^*$ represents the virtual velocities of link *i*. In equation (8.14), the virtual velocities cannot be eliminated, because they are not independent. Only the relative velocities $\dot{\mathbf{z}}$ are independent.

One can transform expression (8.14) and write explicitly the last two terms,

$$\begin{aligned} &\sum_{i=1}^{N-2} \mathbf{Y}_i^{*T} \left(\mathbf{M}_i \dot{\mathbf{Y}}_i - \mathbf{Q}_i \right) + \\ &+ \mathbf{Y}_{N-1}^{*T} \left(\mathbf{M}_{N-1} \dot{\mathbf{Y}}_{N-1} - \mathbf{Q}_{N-1} \right) + \mathbf{Y}_N^{*T} \left(\mathbf{M}_N \dot{\mathbf{Y}}_N - \mathbf{Q}_N \right) = 0 \end{aligned} \tag{8.17}$$

The virtual velocities must satisfy the compatibility equation (8.11). Substituting $\mathbf{Y}_N^*$ and $\dot{\mathbf{Y}}_N$ in equation (8.17)

$$\begin{aligned} &\sum_{i=1}^{N-2} \mathbf{Y}_i^{*T} \left(\mathbf{M}_i \dot{\mathbf{Y}}_i - \mathbf{Q}_i \right) + \mathbf{Y}_{N-1}^{*T} \left(\mathbf{M}_{N-1} \dot{\mathbf{Y}}_{N-1} - \mathbf{Q}_{N-1} \right) + \\ &+ \left(\mathbf{Y}_{N-1}^{*T} \mathbf{B}_N^T + \dot{z}_N^* \mathbf{b}_N^T \right) \left[\mathbf{M}_N \left(\mathbf{B}_N \dot{\mathbf{Y}}_{N-1} + \mathbf{b}_N \ddot{z}_N + \mathbf{d}_N \right) - \mathbf{Q}_N \right] = 0 \end{aligned} \tag{8.18}$$

Reordering and grouping the terms yields

$$\begin{aligned} &\sum_{i=1}^{N-2} \mathbf{Y}_i^{*T} \left(\mathbf{M}_i \dot{\mathbf{Y}}_i - \mathbf{Q}_i \right) + \mathbf{Y}_{N-1}^{*T} \Big[\left(\mathbf{M}_{N-1} + \mathbf{B}_N^T \mathbf{M}_N \mathbf{B}_N \right) \dot{\mathbf{Y}}_{N-1} - \\ &\quad - \left(\mathbf{Q}_{N-1} + \mathbf{B}_N^T \mathbf{Q}_N - \mathbf{B}_N^T \mathbf{M}_N \mathbf{d}_N \right) + \mathbf{B}_N^T \mathbf{M}_N \mathbf{b}_N \ddot{z}_N \Big] + \\ &+ \dot{z}_N^* \left(\mathbf{b}_N^T \mathbf{M}_N \mathbf{B}_N \dot{\mathbf{Y}}_{N-1} + \mathbf{b}_N^T \mathbf{M}_N \mathbf{b}_N \ddot{z}_N + \mathbf{b}_N^T \mathbf{M}_N \mathbf{d}_N - \mathbf{b}_N^T \mathbf{Q}_N \right) = 0 \end{aligned} \tag{8.19}$$

The joint virtual velocity $\dot{z}_N^*$ appears in this equation. This virtual velocity is independent of the remaining virtual velocities; thus the parenthesis multiplying $\dot{z}_N^*$ in equation (8.19) must be zero,

$$\mathbf{b}_N^T \mathbf{M}_N \mathbf{B}_N \dot{\mathbf{Y}}_{N-1} + \mathbf{b}_N^T \mathbf{M}_N \mathbf{b}_N \ddot{z}_N + \mathbf{b}_N^T \mathbf{M}_N \mathbf{d}_N - \mathbf{b}_N^T \mathbf{Q}_N = 0 \tag{8.20}$$

From this equation the independent acceleration $\ddot{z}_N$ can be computed as

$$\mathbf{b}_N^T \mathbf{Q}_N - \mathbf{b}_N^T \mathbf{M}_N \mathbf{B}_N \dot{\mathbf{Y}}_{N-1} - \mathbf{b}_N^T \mathbf{M}_N \mathbf{d}_N \tag{8.21}$$

On the other hand, defining

$$\overline{\mathbf{M}}_{N-1} \equiv \mathbf{M}_{N-1} + \mathbf{B}_N^T \left(\mathbf{M}_N - \mathbf{M}_N \mathbf{b}_N \left(\mathbf{b}_N^T \mathbf{M}_N \mathbf{b}_N\right)^{-1} \mathbf{b}_N^T \mathbf{M}_N\right) \mathbf{B}_N \tag{8.22}$$

$$\overline{\mathbf{Q}}_{N-1} \equiv \mathbf{Q}_{N-1} + \left(\mathbf{B}_N^T - \mathbf{B}_N^T \mathbf{M}_N \mathbf{b}_N \left(\mathbf{b}_N^T \mathbf{M}_N \mathbf{b}_N\right)^{-1} \mathbf{b}_N^T\right) \left(\mathbf{Q}_N - \mathbf{M}_N \mathbf{d}_N\right) \tag{8.23}$$

Equation (8.19) can be written as

$$\sum_{i=1}^{N-2} \mathbf{Y}_i^{*T} \left(\mathbf{M}_i \dot{\mathbf{Y}}_i - \mathbf{Q}_i\right) + \mathbf{Y}_{N-1}^{*T} \left(\overline{\mathbf{M}}_{N-1} \dot{\mathbf{Y}}_{N-1} - \overline{\mathbf{Q}}_N\right) = 0 \tag{8.24}$$

Equation (8.24) is similar to equation (8.17) but with two important differences:

- There are only (*N–1*) terms.
- The inertia matrix and the vector of forces corresponding to link (*N–1*) have been modified according to equations (8.22) and (8.23) in order to incorporate the effects of link *N*. $\overline{\mathbf{M}}_{N-1}$ is the *articulated inertia* of links (*N–1*) and *N*.

The process of eliminating the last element in the virtual power expression can continue in the same manner using the recursive relations,

$$\overline{\mathbf{M}}_{i-1} \equiv \mathbf{M}_{i-1} + \mathbf{B}_i^T \left(\overline{\mathbf{M}}_i - \overline{\mathbf{M}}_i \mathbf{b}_i \left(\mathbf{b}_i^T \overline{\mathbf{M}}_i \mathbf{b}_i\right)^{-1} \mathbf{b}_i^T \overline{\mathbf{M}}_i\right) \mathbf{B}_i \tag{8.25}$$

$$\overline{\mathbf{Q}}_{i-1} \equiv \mathbf{Q}_{i-1} + \left(\mathbf{B}_i^T - \mathbf{B}_i^T \overline{\mathbf{M}}_i \mathbf{b}_i \left(\mathbf{b}_i^T \overline{\mathbf{M}}_i \mathbf{b}_i\right)^{-1} \mathbf{b}_i^T\right) \left(\overline{\mathbf{Q}}_i - \overline{\mathbf{M}}_i \mathbf{d}_i\right) \tag{8.26}$$

$$\ddot{z}_i = \left(\mathbf{b}_i^T \overline{\mathbf{M}}_i \mathbf{b}_i\right)^{-1} \left(\mathbf{b}_i^T \overline{\mathbf{Q}}_i - \mathbf{b}_i^T \overline{\mathbf{M}}_i \mathbf{B}_i \dot{\mathbf{Y}}_{i-1} - \mathbf{b}_i^T \overline{\mathbf{M}}_i \mathbf{d}_i\right) \tag{8.27}$$

$$i = N, N-1, ..., 2, 1$$

Finally, one arrives at the equation of the first link. If it is a floating link, one can write,

$$\mathbf{Y}_1^{*T} \left(\overline{\mathbf{M}}_1 \dot{\mathbf{Y}}_1 - \overline{\mathbf{Q}}_1\right) = 0 \tag{8.28}$$

But now the virtual velocities are independent, so $\dot{\mathbf{Y}}_1$ can be computed as,

$$\dot{\mathbf{Y}}_1 = \overline{\mathbf{M}}_1^{-1} \overline{\mathbf{Q}}_1 \tag{8.29}$$

Summarizing: the method of articulated inertia proceeds with a triple recursion in the following way:

1. Knowing the position and velocity of the base body and the joint relative positions $\mathbf{z}$ and velocities $\dot{\mathbf{z}}$, one can compute recursively forward the Cartesian position and velocity of all the links from *i=1* to *i=N*.

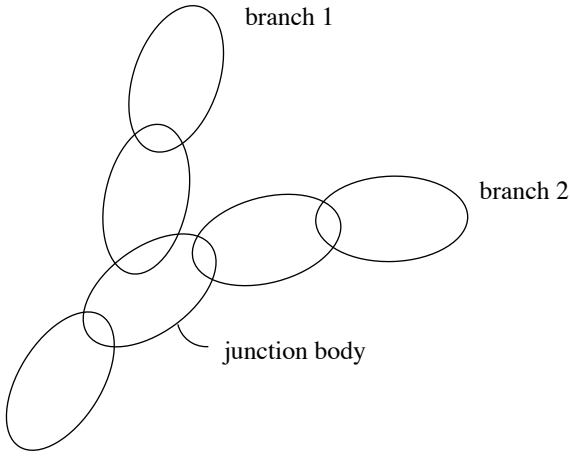


Figure 8.4. Kinematic chain with two branches.

2. The articulated inertias $\overline{\mathbf{M}}$, the forces $\overline{\mathbf{Q}}$, and the coefficients of equation (8.27) are then computed recursively backwards from $i=N$ to $i=1$, using equations (8.25)-(8.27).
3. Finally, the acceleration of the base body is computed from equation (8.29) and then the relative accelerations $\ddot{z}_i$ are computed recursively forward from $i=1$ to $i=N$, using equation (8.27).

The Featherstone version of this algorithm is probably much more efficient from the computational point of view. Bae, Hwang, and Haug (1988) presented an improved version of this algorithm, using as reference point the point of the moving body that instantaneously coincides with the origin of the inertial reference frame. Looking at the triple recursive procedure, one can see that the number of arithmetic operations grows proportionally with the number of degrees of freedom of the open-chain; thus it is an *O(N)* method.

8.1.3 Extension to Branched and Closed-Chain Configurations

The ideas explained in the two previous sections can be extended to multibody systems with any kinematic configuration. There are three main directions in which this formulation can be generalized: a) include any kind of joints, b) extend it to multibody systems with many branches on a tree-configuration, and c) generalize it to systems with closed loops. The extension of this formulation to multibody systems with joints of any type is a straightforward task. It can be found in the original references.

The consideration of branches in the kinematic chain (See Figure 8.4) is also a simple task. Both the composite inertia and the articulated inertia methods can be easily accommodated to include junction bodies, that is, links with more than

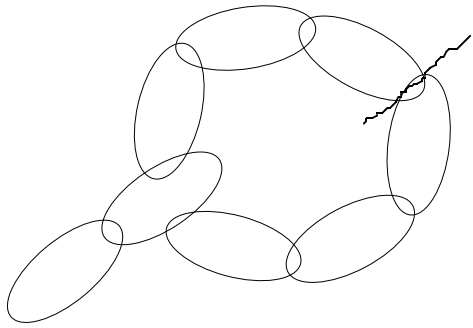


Figure 8.5. Cut-joint method to open a closed-chain multibody system.

two joints. In the junction bodies, remember that the forward recursive computations must be split into two separate procedures that move independently along each branch. In the backward recursive computations, the two separate procedures along each branch meet at the junction body and yield a single procedure. From the theoretical point of view, the formulation can be extended to include tree-structured multibody systems without difficulty. The main difficulties arise in the practical implementation, when one tries to compute in parallel terms corresponding to different branches, since it is necessary to set up synchronization points at the junction bodies.

More difficulties can be found when the above formulations are modified to tackle closed-chain multibody systems. These can be transformed into open-chain systems through the *cut-joint* method which eliminates or *cuts* a joint in each loop, as seen in Figure 8.5. This method is described by Bae et al. (1987-88, 1988).

Where a joint is removed, the corresponding constraint forces, formulated through the Lagrange multipliers method as ($\mathbf{\Phi}_\mathbf{q}^T \boldsymbol{\lambda}$) and ($-\mathbf{\Phi}_\mathbf{q}^T \boldsymbol{\lambda}$) on both links, can be propagated backwards on both branches just as equation (8.23) indicates. All joint accelerations in the loop, given by equation (8.27), depend on the Lagrange multipliers vector $\boldsymbol{\lambda}$.

In order to have enough equations to compute the relative accelerations $\ddot{\mathbf{z}}$ and the Lagrange multipliers $\boldsymbol{\lambda}$, it is necessary to differentiate twice the constraint equations corresponding to the cut joint.

Initially, these constraints are formulated using the Cartesian coordinates of the adjacent bodies. One must to substitute backwards on both branches the Cartesian velocities and accelerations by the corresponding relative variables using equations (8.11) and (8.12). Finally, as many equations as unknowns are available, and the details of this fully recursive formulation are described in Bae and Haug (1987-88). In a later work Bae, Hwang, and Haug (1988) introduced a modification addressed to compute all the relative accelerations $\ddot{\mathbf{z}}$ at once by solving a system of linear equations; thus becoming an $O(N^3)$ method. More recently, García de Jalón et al. (1989) and Bae and Won (1990) presented other

formulations better suited for real time analysis, which are based on *velocity transformations* (Jerkovsky (1978), Kim and Vanderploeg (1986)).

The recursive methods, so eagerly accepted at the beginning, have been steadily losing ground and interest in favor of the methods based on velocity transformations which seem simpler to formulate and easier to parallelize.

8.2 Velocity Transformations for Open-Chain Systems

Some of the most efficient formulations for the forward dynamic analysis that have been developed in the last decade have been dealt with in the preceding sections. These methods were classified, according to the number of floating-point operations that they require, as methods of order $O(N)$ or $O(N^3)$. Although there are important theoretical and practical differences among them, there are also common aspects that are worth pointing out.

The first point in common is their *origin*. These formulations arose from the study of the dynamics of serial robots, were posteriorly extended to other more general open-chain and tree-type multibody systems, and to those with a closed chain configuration. Therefore, the *topology* of the multibody system has played a key role in the development history of these recursive formulations. This is a point to be contrasted with kinematic and dynamic methods exposed in Chapters 3 and 5 which can be really considered as *configuration independent*.

The second point in common between the $O(N)$ and $O(N^3)$ families is the extensive use, with a different emphasis, that those methods make of the concept of *recursion* along the kinematic chain. The recursive formulations have meant an important contribution to the dynamics of robots in particular, and to the dynamics of multibody systems in general, since they have substantially reduced the number of required floating-point operations. This does not mean that they do not suffer from some limitations that make the other non-recursive formulations compete advantageously against them.

From a computational viewpoint, vectorization has an enormous importance in the solution of large computational problems with thousands of degrees of freedom, such as those arising in fluid and solid mechanics. It may not influence the analysis of multibody systems very much, because these problems lead to systems of equations of small or moderate size. Parallelization, however, may have a very important effect in the dynamics of multibody systems, provided the dynamic method is such that different parts of the computational process may be executed simultaneously.

A general and simple method that formulates the dynamic equations of any open- or closed-chain multibody system, and which can be parallelized even to the body (or element) level will be studied in this and the next sections. This formulation is based upon the *velocity transformations* considered in Section 5.3.

The general purpose dynamic formulations described in Sections 5.1 and 5.2, are simple and may be applied to any multibody system regardless of their con-

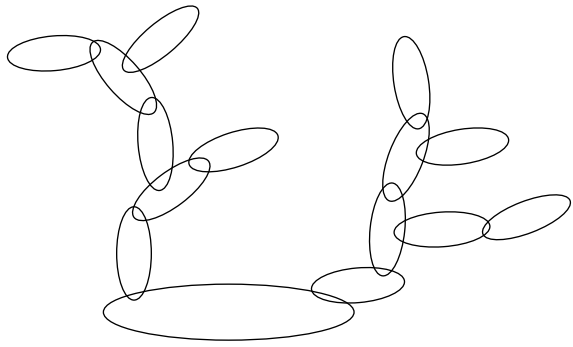


Figure 8.6. Open-chain multibody system with tree structure.

figuration. They treat all systems in the same way, regardless of their topology and particular characteristics which make those methods insufficiently efficient for real time applications. A way to improve the efficiency of these formulations is to take advantage of the open-chain configurations that the multibody systems may have, or in which they may be transformed.

Open-chain multibody systems are less constrained than the closed-chain ones. This is a fact that when conveniently considered can be used to improve the efficiency of the dynamic formulation. Open-chain multibody systems are not only frequently found in robotics, aerospace, and biomechanics, but in *any closed-chain multibody system that may also be transformed into an open-chain configuration by simply opening its loops* (removing certain constraint equations, which can be considered separately). Consequently, all the benefits that may be drawn from the formulation of open-chain systems can be posteriorly extended to closed kinematic chains as well. The study of the open-chain multibody systems is next.

8.2.1 Dependent and Independent Coordinates

An open-chain multibody system will be considered that consists of one or several *branches*, forming a *tree structure* and connected to a *base element*, as shown in Figure 8.6. The total number of degrees of freedom f (or independent coordinates) will be equal to the six degrees of freedom of the base body plus all the relative degrees of freedom allowed by the kinematic joints along the branches of the multibody system. In order to be consistent with the formulations previously written in this book, the vector of independent coordinates will be denoted as $\mathbf{z}$. This vector can be composed for instance of the three translations and three rotations of the base body, the rotations of the revolute joints, the translations of the prismatic joints the two rotations of an universal joint, and so forth. The vector $\mathbf{z}$, thus formed and its derivatives, will

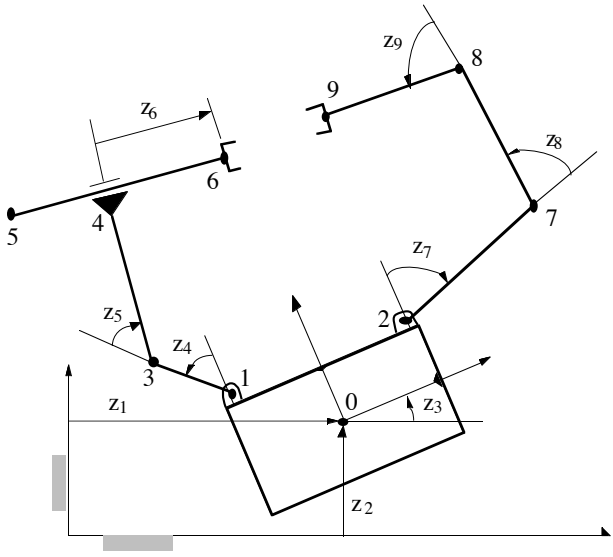


Figure 8.7. Planar system with floating base body and two robot arms.

completely and univocally define the position and motion of the open-chain system.

It is also possible to describe the motion of the open-chain multibody system by means of an augmented set of coordinates $\mathbf{q}$ with n components, where $n>f$. These coordinates will no longer be independent, but interrelated through $(n–f)$ constraint equations of the form:

$$\mathbf{\Phi}(\mathbf{q}) = 0 \tag{8.30}$$

where it has been assumed, without important loss of generality, that the constraints are scleronomous.

Whereas there is a *natural choice* for the independent coordinates $\mathbf{z}$ (degrees of freedom of the base body plus the relative coordinates at the joints), the choice of dependent coordinates $\mathbf{q}$ is not so restrictive. One may make a choice from among the different sets of dependent coordinates based on convenience of implementation or even on personal preference. The following example clarifies this point.

Example 8.1

Figure 8.7 illustrates a planar multibody system with a base element that can move freely and to which two robots with three degrees of freedom each are attached. Therefore, there is a total number of nine degrees of freedom, corresponding to the independent coordinates z_1 to z_9. Among the many possible sets of dependent coordinates, the following set $\mathbf{q}$ of natural coordinates can be chosen,

$$\mathbf{q}^{\mathrm{T}} = \{\mathbf{r}_1^{\mathrm{T}}, \mathbf{r}_2^{\mathrm{T}}, \mathbf{r}_3^{\mathrm{T}}, \mathbf{r}_4^{\mathrm{T}}, \mathbf{r}_5^{\mathrm{T}}, \mathbf{r}_6^{\mathrm{T}}, \mathbf{r}_7^{\mathrm{T}}, \mathbf{r}_8^{\mathrm{T}}, \mathbf{r}_9^{\mathrm{T}}\}$$

where $\mathbf{r}_i^{\mathrm{T}}=\{x_i, y_i\}$. These 18 Cartesian coordinates will be interrelated by means of nine constraint conditions, that the reader may formulate as an exercise following the rules given in Chapter 2.

Another possibility is to form a vector of dependent coordinates q composed of the previous set plus the coordinates of the center of gravity of the base body, those that describe its orientation, and the relative coordinates of the joints. Thus, the following set of 27 mixed coordinates is obtained,

$$\mathbf{q}^{\mathrm{T}} = \{\mathbf{r}_0^{\mathrm{T}}, z_3, \mathbf{r}_1^{\mathrm{T}}, z_4, \mathbf{r}_2^{\mathrm{T}}, z_7, \mathbf{r}_3^{\mathrm{T}}, z_5, \mathbf{r}_4^{\mathrm{T}}, z_6, \mathbf{r}_5^{\mathrm{T}}, \mathbf{r}_6^{\mathrm{T}}, \mathbf{r}_7^{\mathrm{T}}, z_8, \mathbf{r}_8^{\mathrm{T}}, z_9, \mathbf{r}_9^{\mathrm{T}}\}$$

which will require 18 constraint equations.

A third possibility is to form $\mathbf{q}$ from the reference point coordinates of all the elements, including the coordinates of the center of gravity and the angular orientation of each element with respect to the inertial frame, which will lead to the following 21 dependent coordinates

$$\mathbf{q}^{\mathrm{T}} = \{\mathbf{g}_1^{\mathrm{T}}, \varphi_1, \mathbf{g}_2^{\mathrm{T}}, \varphi_2, \mathbf{g}_3^{\mathrm{T}}, \varphi_3, \mathbf{g}_4^{\mathrm{T}}, \varphi_4, \mathbf{g}_5^{\mathrm{T}}, \varphi_5, \mathbf{g}_6^{\mathrm{T}}, \varphi_6, \mathbf{g}_7^{\mathrm{T}}, \varphi_7\}$$

where $\mathbf{g}_i^{\mathrm{T}}=\{g_{ix}, g_{iy}\}$ is the position vector of the center of gravity of body (i). These coordinates can also be augmented by the addition of all or part of the relative coordinates at the kinematic pairs.

The last possibility is to form a vector $\mathbf{q}$ that includes *all* the available information in terms of the position of the basic points, position of the center of gravity, and the rotation matrix that relates the orientation of the element with respect to the inertial frame. Accordingly,

$$\mathbf{q}^{\mathrm{T}} = \{\mathbf{r}_1^{\mathrm{T}}, \mathbf{r}_2^{\mathrm{T}}, \mathbf{g}_1^{\mathrm{T}}, \mathbf{A}_1, \mathbf{r}_3^{\mathrm{T}}, \mathbf{g}_2^{\mathrm{T}}, \mathbf{A}_2, \mathbf{r}_4^{\mathrm{T}}, \mathbf{g}_3^{\mathrm{T}}, \mathbf{A}_3, \ldots..\}$$

where $\mathbf{r}_i$ is the position vector of the point i, and $\mathbf{g}_j$ and $\mathbf{A}_j$ contain respectively the coordinates of the center of gravity and the rotation matrix of element (j). As above, this vector $\mathbf{q}$ can be augmented by the relative coordinates of the joints and by the Cartesian components of the unit vectors in the case of three-dimensional multibody systems.

It has been seen in the previous example that there are many different sets of dependent coordinates that can conveniently represent the position of open-chain multibody systems. One should choose the most favorable one from the viewpoint of practical implementation, and the velocity and acceleration vectors need not be the derivatives, term by term, of the dependent position coordinates. For example, for the dependent position coordinates for a rigid body, one can chose those of the center of gravity and the Euler angles. As dependent velocities, one can chose those of the center of gravity plus the vector of angular velocities $\boldsymbol{\omega}$, which is not obtained through the direct differentiation of the Euler angles. The Euler angles could be replaced by the Euler parameters or by the nine components of the rotation matrix. They would still use as dependent velocities those of the center of gravity and the vector of angular velocities $\boldsymbol{\omega}$.

The reason for this peculiar choice comes from the non-integrability of the vector of angular velocities $\boldsymbol{\omega}$. One is forced to look for other sets of position variables (obviously more complicated than $\boldsymbol{\omega}$) that will enable representation of

the angular orientation. In order to take into account the different character between the position vector and the velocity or acceleration vectors, we introduce the following notation: $\overline{\mathbf{q}}$ will represent the vector of position dependent coordinates; whereas $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ will represent the dependent velocity and acceleration vectors. Accordingly, the following relations will hold:

$$\dot{\mathbf{q}} \neq \dot{\overline{\mathbf{q}}} \tag{8.31}$$

$$\dot{\mathbf{q}} = \mathbf{V}(\overline{\mathbf{q}})\ \dot{\overline{\mathbf{q}}} \tag{8.32}$$

$$\dot{\overline{\mathbf{q}}} = \mathbf{U}(\overline{\mathbf{q}})\ \dot{\mathbf{q}} \tag{8.33}$$

where $\mathbf{V}(\overline{\mathbf{q}})$ is a position-dependent matrix that transforms the derivatives of the position vector $\overline{\mathbf{q}}$ into the velocities $\dot{\mathbf{q}}$. The matrix $\mathbf{U}(\overline{\mathbf{q}})$ represents the velocity inverse transformation between $\dot{\overline{\mathbf{q}}}$ and $\dot{\mathbf{q}}$.

8.2.2 Dependent and Independent Velocities: Matrix **R**

The *velocity transformations* introduced in Chapters 3 and 5 will now be considered, The corresponding equations will be rewritten here for convenience as,

$$\dot{\mathbf{q}} = \mathbf{R}(\overline{\mathbf{q}})\ \dot{\mathbf{z}} + \mathbf{Sb}(\overline{\mathbf{q}}) \tag{8.34}$$

$$\ddot{\mathbf{q}} = \mathbf{R}(\overline{\mathbf{q}})\ \ddot{\mathbf{z}} + \mathbf{Sc}(\overline{\mathbf{q}}, \dot{\overline{\mathbf{q}}}) \tag{8.35}$$

where the dependency of the matrix **R** and the terms (**Sb**) and (**Sc**) on $\overline{\mathbf{q}}$ has been explicitly indicated in (8.34) and (8.35). The term (**Sb**) will be zero if the constraints are scleronomous.

The columns of the matrix **R** constitute a basis for the vector space of all the possible velocity vectors $\dot{\mathbf{q}}$. Any vector $\dot{\mathbf{q}}$ that satisfies the velocity constraint equations may be expressed as a linear combination of the *f* columns of the matrix **R**. As shown in equation (8.34), the components of the vector $\dot{\mathbf{z}}$ of independent velocities are the coefficients of such linear combination.

Equation (8.34) also indicates the physical significance of the columns of the matrix **R**. Column *i* represents the dependent velocities $\dot{\mathbf{q}}$ obtained by giving a unit value to the component $\dot{z}_i$ of the vector $\dot{\mathbf{z}}$, and zero to the rest of the components. Thus,

$$\mathbf{r}^i = \dot{\mathbf{q}}\,\Big|_{\dot{z}_i = 1 \text{ and } \dot{z}_j = 0 \text{ for } j \neq i} \tag{8.36}$$

where $\mathbf{r}^i$ represents the column i of the matrix **R**, as compared with $\mathbf{r}_i$ which represents the position vector of point *i*.

The definition made in equation (8.36) of the columns of the matrix **R** leads for the general case of open- and closed-chain multibody systems, to the methods explained in Chapter 5, such as the LU factorization of the Jacobian $\mathbf{\Phi}_{\mathbf{q}}$,

Singular Value decomposition, and so forth. In the particular case of open-chain multibody systems that are represented by the independent coordinates mentioned above, *the columns of the matrix* $\mathbf{R}$ *can be obtained directly* through velocity computations, without the need of forming and factoring the Jacobian matrix; thus leading to a greatly reduced numerical effort. In addition, the following advantages can also be obtained:

a) The *sparsity* pattern of the matrix $\mathbf{R}$ becomes apparent and can be used in subsequent matrix operations.
b) The part of the matrix $\mathbf{R}$ that affects a particular link or element of the multibody system can be formed independently of the rest of the elements. This property leads to an *element-by-element* treatment of the equations of motion.

These advantages become apparent when considering the physical significance of each of the columns of the matrix $\mathbf{R}$. Column i represents the dependent velocities resulting from a unit value of the independent velocity $\dot{z}_i$ and zero values of the rest of independent velocities. Taking into account that vector $\mathbf{z}$ contains the position coordinates of the base body plus the relative joint coordinates, three possibilities can be considered:

1) $\dot{z}_i$ corresponds to a translational velocity of the base body along one of the inertial axis. All the elements of the multibody system will have a unit translational velocity along the same inertial axis. Only the Cartesian coordinates of the basic points will be affected by this translation. The rest of the dependent velocities including unit vectors and relative joint coordinates will take zero values.
2) $\dot{z}_i$ corresponds to a rotational velocity of the base body along one of the inertial axis. The corresponding column of the matrix $\mathbf{R}$ will contain the rotation velocities of all the points and unit vectors about an axis parallel to the inertial one. This axis goes through the reference point of the base body.
3) $\dot{z}_i$ corresponds to the relative velocity of one of the kinematic joints. Only the distal elements (those after the corresponding joint) of the kinematic chain will be affected by the relative joint velocity.

The following example will make use of these considerations to show the ease by which the different columns of the matrix $\mathbf{R}$ can be calculated in an open-chain multibody system, when the proposed sets of coordinates are used.

Example 8.2

Consider the open-chain multibody system illustrated in Figure 8.7 which is represented by a set of mixed coordinates. Derive the columns of the matrix $\mathbf{R}$ that correspond to the following independent velocities: $\dot{z}_1$, $\dot{z}_3$, and $\dot{z}_7$.

The vector of dependent mixed coordinates is, in this case,

$$\mathbf{q}^{\mathrm{T}} = \{\mathbf{r}_0^{\mathrm{T}}, z_3, \mathbf{r}_1^{\mathrm{T}}, z_4, \mathbf{r}_3^{\mathrm{T}}, z_5, \mathbf{r}_4^{\mathrm{T}}, \mathbf{r}_5^{\mathrm{T}}, z_6, \mathbf{r}_6^{\mathrm{T}}, \mathbf{r}_2^{\mathrm{T}}, z_7, \mathbf{r}_7^{\mathrm{T}}, z_8, \mathbf{r}_8^{\mathrm{T}}, z_9, \mathbf{r}_9^{\mathrm{T}}\}$$

where $\mathbf{r}_i$ is the position vector of point i, and z_j represents the relative coordinates.

a) When $\dot{z}_1$=1, all the points will have the same velocity $\mathbf{n}_1^T=\{1,0\}$, and the relative coordinates will have zero velocity. Accordingly,

$$\mathbf{r}^{1T}=\{\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T,0,\mathbf{n}_1^T\}$$

b) When $\dot{z}_3 = 1$ (unit rotation of the base element), each point will have the following velocity: $\mathbf{v}_i=\dot{\mathbf{r}}_i=\mathbf{k}\wedge(\mathbf{r}_i-\mathbf{r}_0)$, where $\mathbf{k}$ is the unit vector perpendicular to the plane of the multibody system. Therefore

$$\mathbf{r}^{3T}=\{0^T,1,\mathbf{v}_1^T,0,\mathbf{v}_3^T,0,\mathbf{v}_4^T,\mathbf{v}_5^T,0,\mathbf{v}_6^T,\mathbf{v}_2^T,0,\mathbf{v}_7^T,0,\mathbf{v}_8^T,0,\mathbf{v}_9^T\}$$

c) Finally, if a unit relative velocity is introduced in the joint 7, $\dot{z}_7 = 1$, only those points that belong to the distal elements after joint 7 will have a non-zero velocity, which will have the following value: $\mathbf{v}_i=\dot{\mathbf{r}}_i=\mathbf{k}\wedge(\mathbf{r}_i-\mathbf{r}_2)$, $(i=7,8,9)$. Consequently, the seventh column of $\mathbf{R}$ becomes

$$\mathbf{r}^{7T}=\{0^T,0,0^T,0,0^T,0,0^T,0^T,0,0^T,0^T,1,\mathbf{v}_7^T,0,\mathbf{v}_8^T,0,\mathbf{v}_9^T\}$$

d) In order to complete this example, the part of the matrix $\mathbf{R}$ that corresponds to the element that joins the points 3 and 4 will be calculated separately. This part of the matrix $\mathbf{R}$ is composed of the columns that correspond to the degrees of freedom of the base element (z_1, z_2, z_3) and those of the joints (z_4, z_5) that are before the element within the same kinematic chain. The result for the element is

$$\mathbf{R}_{3\text{-}4}=\begin{array}{c}\begin{matrix}\dot{z}_1 & \dot{z}_2 & \dot{z}_3 & \dot{z}_4 & \dot{z}_5\end{matrix}\\ \left[\begin{array}{c|c|c|c|c} 1 & 0 & & & \\ 0 & 1 & \mathbf{k}\wedge(\mathbf{r}_3-\mathbf{r}_0) & \mathbf{k}\wedge(\mathbf{r}_3-\mathbf{r}_1) & 0 \\ 1 & 0 & & & \\ 0 & 1 & \mathbf{k}\wedge(\mathbf{r}_4-\mathbf{r}_0) & \mathbf{k}\wedge(\mathbf{r}_4-\mathbf{r}_1) & \mathbf{k}\wedge(\mathbf{r}_4-\mathbf{r}_3) \end{array}\right]\end{array}$$

The previous example clearly illustrates two very important advantages of this method: a) the way in which the matrix $\mathbf{R}$ can be calculated is direct, systematic, and general; and, b) the procedure can be carried out for each body independently in a *body-by-body* or *element-by-element* basis (rather than recursively). Therefore, the method can take full advantage of parallel computer architectures.

Finally, the columns of matrix $\mathbf{R}$, thus calculated, constitute a basis for the nullspace (subspace of possible motions) of the Jacobian matrix $\mathbf{\Phi_q}$. Although the constraint equations are not explicitly calculated, the following relation will always hold:

$$\mathbf{\Phi_q}\,\mathbf{R} = 0 \tag{8.37}$$

Table 8.1. Algorithm to formulate and integrate the equations of motion of an open-chain system.

Step	Data	Result	Mode
1	$\mathbf{z}$	$\overline{\mathbf{q}}$	recursive
2	$\overline{\mathbf{q}}$	$\mathbf{R}$	e-by-e or rec.
3	$\dot{\mathbf{z}}, \overline{\mathbf{q}}$	$\dot{\mathbf{q}}$	recursive
4	$\dot{\mathbf{z}}, \overline{\mathbf{q}}, \dot{\mathbf{q}}$	$(\mathbf{Sc})$	e-by-e
5	$\mathbf{R}, \mathbf{M}$	$\mathbf{R}^T \mathbf{M} \mathbf{R}$	e-by-e
6	$\mathbf{R}, \mathbf{Q}$	$\mathbf{R}^T \mathbf{Q}$	e-by-e
7	$\mathbf{R}, \mathbf{Sc}, \mathbf{M}$	$\mathbf{R}^T \mathbf{M} \mathbf{Sc}$	e-by-e
8	Linear equations	$\ddot{\mathbf{z}}$	global
9	$(\ddot{\mathbf{z}}, \dot{\mathbf{z}})_t$	$(\dot{\mathbf{z}}, \mathbf{z})_{t+\Delta t}$	global
10	GO TO 1		

8.2.3 *Equations of Motion*

Once the matrix $\mathbf{R}$ is known, the methods presented in Chapter 5 for the formulation of the equations of motion in independent coordinates can be used. We will use equation (5.67) that is written again for convenience as:

$$\mathbf{R}^T \mathbf{M} \mathbf{R} \ddot{\mathbf{z}} = \mathbf{R}^T \mathbf{Q} - \mathbf{R}^T \mathbf{M} \mathbf{S} \mathbf{c} \tag{8.38}$$

where $\mathbf{Q}$ is the vector containing the external forces, those coming from a potential and the velocity-dependent ones. The term $(\mathbf{Sc})$ contains the dependent accelerations $\ddot{\mathbf{q}}$ that are calculated from the true velocities $\dot{\mathbf{q}}$ by equating to zero the independent accelerations $\ddot{\mathbf{z}}$ in equation (5.65).

The algorithm that allows the formulation and numerical integration of the motion differential equations of an open-chain system, according to the proposed method and to equation (8.38), is given in detail in Table 8.1. The integration is carried out in independent coordinates; and therefore no constraint violation stabilization is considered. Table 8.1 also indicates the way each step can be calculated in an optimal way. Some steps can be carried out in an *element-by-element* basis; thus susceptible for being parallelized. Other steps are carried out recursively, and others in a global manner.

The steps included in Table 8.1 will be described more in detail, starting from the position problem.

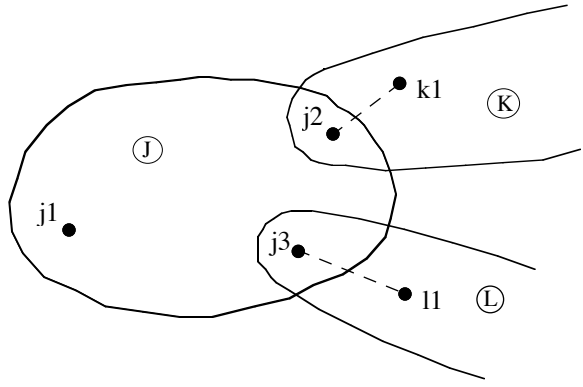


Figure 8.8. Body with one input point and two output points.

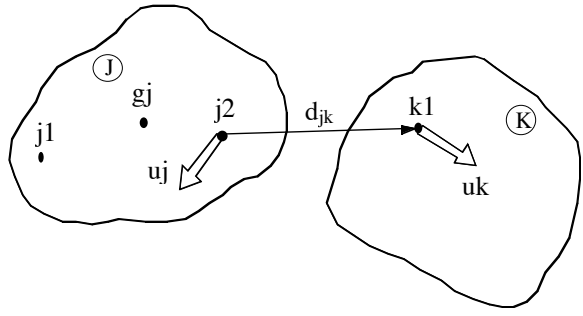


Figure 8.9. Model of a generalized joint.

8.2.4 Position Problem

A recursive solution to the position problem of a spatial open-chain multibody system will be shown in this section. We assume that the multibody system is composed of a base element and a series of branches with arbitrary size and distribution composed of rigid bodies interconnected by any of the following joints: revolute (R), prismatic (P), cylindrical (C), universal (U), or spherical (S). The formulation can also be extended without difficulty to other types of joints.

The position problem, required in Table 8.1, consists in finding the dependent coordinates $\overline{\mathbf{q}}$ that define the position of the individual elements of the multibody system from the independent coordinates $\mathbf{z}$, composed of the degrees of freedom of the base body plus the joint coordinates. Later on, in order to avoid possible singular positions, dependent angular orientation coordinates will be included for the base body and for the spherical joints. The vector of dependent coordinates $\overline{\mathbf{q}}$ includes the natural coordinates of the elements plus the following additional variables:

- Cartesian coordinates of the points $\mathbf{r}_i$ referred to the inertial frame.
- Cartesian components of the unit vectors $\mathbf{u}_i$ referred to the inertial frame.
- Rotation matrices $\mathbf{A}_i$, that relate the current position of every element with respect to a known reference or initial position.
- Cartesian components of the center of gravity $\mathbf{g}_i$ of each body expressed in the inertial frame.

The aim of this section is to calculate the components of the position vector $\overline{\mathbf{q}}$ in a recursive manner (avoiding the expensive Newton-Raphson iterations described in Chapter 3), starting from the base body and moving forward towards the distal elements in the different branches. We will assume that the vector $\overline{\mathbf{q}}_0$ with the initial or reference positions is known.

Figure 8.8 shows a general element J that has an input point *j1*, whose position is already known, and one or more output points *j2*, *j3*, and so forth. These output joints are related to the pairs that join the element J with the posterior elements in the chain. Figure 8.9 illustrates the model of the *generalized joint* that joins elements J and K. The points *j2* and *k1* are the output point of element J and input point of K, respectively. The vectors $\mathbf{u}_j$ and $\mathbf{u}_k$ belong to J and K, respectively, and must be chosen according to the type of the kinematic pair that joins both elements. The relative degrees of freedom allowed by this joint will be represented by a vector of relative joint coordinates $\mathbf{z}_{jk}$. The vector $\mathbf{d}_{jk}$ corresponds to the position vector between points *j2* and *k1*.

Assuming that the position vector $\mathbf{r}_{j1}$ and rotation matrix $\mathbf{A}_j$ corresponding to the element J are known, the solution of the position problem gives the remaining position variables of J (such as the center of gravity $\mathbf{g}_j$, output point $\mathbf{r}_{j2}$ and unit vector $\mathbf{u}_j$), plus the input position variables of element K (position vector $\mathbf{r}_{k1}$ and rotation matrix $\mathbf{A}_k$). The necessary calculations are represented schematically in Figure 8.10. The corresponding analytical expressions are:

$$\mathbf{g}_j = \mathbf{r}_{j1} + \mathbf{A}_j \, (\mathbf{g}_j^o - \mathbf{r}_{j1}^o) \tag{8.39}$$

$$\mathbf{u}_j = \mathbf{A}_j \, \mathbf{u}_j^o \tag{8.40}$$

$$\mathbf{r}_{k1} = \mathbf{r}_{j1} + \mathbf{A}_j \, (\mathbf{r}_{j2}^o - \mathbf{r}_{j1}^o) + \mathbf{d}_{jk} \tag{8.41}$$

$$\mathbf{A}_k = \mathbf{A}_{jk}(\mathbf{z}_{jk}) \, \mathbf{A}_j \tag{8.42}$$

where the superscript $(\text{-})^o$ refers to the corresponding variable to the known initial or reference position defined in the inertial frame. The matrix $\mathbf{A}_{jk}(\mathbf{z}_{jk})$ is the relative rotation matrix of the kinematic joint between elements J and K that depends on the relative rotation. Note that equations (8.39) to (8.42) apply to any output variable and joint of the element J.

Equations (8.39) through (8.42) will be particularized to each of the different joints.

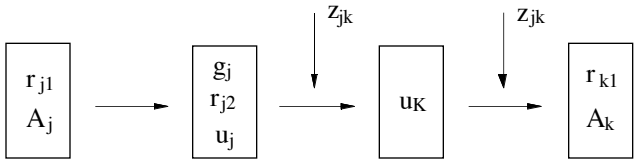


Figure 8.10. Scheme of recursive position calculations.

Revolute Joint R. Equations (8.41) and (8.42) become in this case:

$$\mathbf{d}_{jk} = 0 \tag{8.43}$$

$$\mathbf{A}_{jk} = \mathbf{A}(z_{jk}, \mathbf{u}_j) \tag{8.44}$$

where z_{jk} is the angle rotated by the joint with respect to the reference configuration $\overline{\mathbf{q}}_0$, and $\mathbf{u}_j$ is the unit vector in the direction of the joint axis. The rotation matrix in (8.44) is defined in terms of the angular rotation z_{jk} about the axis defined by the unit vector $\mathbf{u}_j$. It can be demonstrated (See Argyris (1982)) that

$$\mathbf{A}(z_{jk}, \mathbf{u}) = \mathbf{I} + \tilde{\mathbf{u}}\,\tilde{\mathbf{u}}\,(1 - cos\ z_{jk}) + \tilde{\mathbf{u}}\ sin\ z_{jk} \tag{8.45}$$

where $\tilde{\mathbf{u}}$ is the skew-symmetric matrix commonly used to perform the cross product of vectors.

Prismatic Joint P. Considering that the vectors $\mathbf{u}_j$ and $\mathbf{d}_{jk}$ have the direction of the relative translation allowed by the prismatic joint, the following equations describe the relative configuration of the elements:

$$\mathbf{d}_{jk} = z_{jk}\,\mathbf{u}_j \tag{8.46}$$

$$\mathbf{A}_k = \mathbf{A}_j \tag{8.47}$$

Cylindrical Joint C. The vectors $\mathbf{u}_j$ and $\mathbf{d}_{jk}$ are considered to have the direction of the joint axis. Consequently,

$$\mathbf{d}_{jk} = z_{jk}^{t}\,\mathbf{u}_j \tag{8.48}$$

$$\mathbf{A}_k = \mathbf{A}_{jk}\left(z_{jk}^{r}, \mathbf{u}_j\right)\mathbf{A}_j \tag{8.49}$$

where z_{jk}^{r} and z_{jk}^{t} are the relative coordinates that represent the rotation and translation of the joint, respectively.

Spherical Joint S. The spherical joint allows an arbitrary rotation that may be defined either by the Euler angles, the Bryant angles, or the Euler parameters $\mathbf{p}$ which are interrelated by the following condition $p_0^2+p_1^2+p_2^2+p_3^2 = 1$. Equations (8.48) and (8.49) become:

$$\mathbf{d}_{jk} = 0 \tag{8.50}$$

$$\mathbf{A}_k = \mathbf{A}_{jk}(\mathbf{z}_{jk} \equiv \mathbf{p}) \cdot \mathbf{A}_j \tag{8.51}$$

The rotation matrix in terms of the Euler parameters is defined as (Nikravesh (1988) and Haug (1989))

$$\mathbf{A}(\mathbf{p}) = \left(2\, e_o^2 - 1\right) \mathbf{I}_3 + 2\left(\mathbf{e}\ \mathbf{e}^T + e_o\, \tilde{\mathbf{e}}\right) \tag{8.52}$$

where $e_o = p_o$, and $\mathbf{e}^T = (p_1, p_2, p_3)$.

Universal Joint U. This joint allows two rotations z_{jk}^1 and z_{jk}^2 with respect to two perpendicular axes along the unit vectors $\mathbf{u}_j$ and $\mathbf{u}_k$. The axes intersect at the common point between both bodies J and K. The equations that correspond to this joint become

$$\mathbf{d}_{jk} = 0 \tag{8.53}$$

$$\mathbf{u}_j = \mathbf{A}_j\, \mathbf{u}_j^o \tag{8.54}$$

$$\mathbf{u}_k = \mathbf{A}_{jk}^1(z_{jk}^1,\ \mathbf{u}_j)\, \mathbf{A}_j\, \mathbf{u}_k^o \tag{8.55}$$

$$\mathbf{A}_k = \mathbf{A}_{jk}^2(z_{jk}^2,\ \mathbf{u}_k)\, \mathbf{A}_{jk}^1(z_{jk}^1,\ \mathbf{u}_j)\, \mathbf{A}_j \tag{8.56}$$

Base body. Let the independent coordinate vector of the base element $\mathbf{z}_b$ be defined by the coordinates of a reference point *i* and by the Euler parameters. Consequently,

$$\mathbf{z}_b^T \equiv \left(\Delta\mathbf{r}_i^T\ \mathbf{p}^T\right) \tag{8.57}$$

and the following two relations will hold for the reference point and rotation matrix:

$$\mathbf{r}_i = \mathbf{r}_i^o + \Delta\mathbf{r}_i \tag{8.58}$$

$$\mathbf{A} = \mathbf{A}(\mathbf{p}) \tag{8.59}$$

Equations (8.39) to (8.59) allow one to solve recursively the position problem for a wide range of open-chain multibody systems.

8.2.5 Velocity and Acceleration Problems

A recursive process addressed to obtain the dependent velocity and acceleration vectors of a multibody system can be similar and even simpler than that used in the preceding section to solve the position problem.

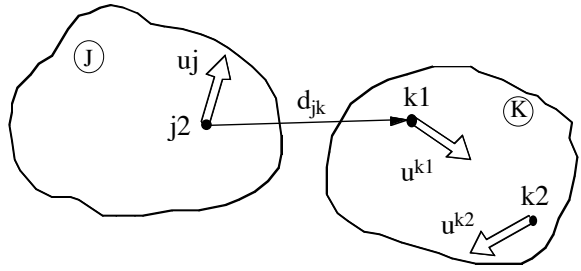


Figure 8.11. Consecutive bodies for recursive velocity calculation (Form. A)

It is possible and even convenient to use a set of dependent velocities and accelerations that does not correspond to the derivatives of the dependent position vector $\overline{\mathbf{q}}$. It has been shown in the previous section that the vector $\overline{\mathbf{q}}$ may be composed of the Cartesian components of basic points, those of the centers of gravity, and the components of unit vectors and the rotation matrices. Two different sets of dependent velocities and accelerations will be considered, that will lead to the following formulations:

a) Formulation A, based on the Cartesian components of basic points and unit vectors. This formulation follows the concept of natural coordinates so much used throughout this book. If an element is defined by two basic points and two unit vectors the corresponding mass matrix is constant and no velocity-dependent inertia terms are involved in the forcing function. It is possible to take advantage of these important facts at the time of implementing the algorithm of Table 8.1.

b) Formulation B, based on the reference point coordinates. This formulation uses the velocity and acceleration vectors of the center of gravity and the angular velocity and acceleration vectors of each element. This leads to some important advantages at the time of calculating the matrix **R** in an *element-by-element* basis, because contrary to the Formulation A, there are no variables shared by the different elements. These advantages materialize in an ease of parallelization and in computational savings.

The analytical expressions necessary to calculate the dependent velocities and accelerations in both formulations will be described next.

8.2.5.1 Formulation A

In this case, it is necessary to compute the velocities of the basic points and unit vectors. Figure 8.11 illustrates two consecutive elements J and K whose positions are known. In addition, it will be assumed that the velocity of J, as well as the relative velocity of the joint $\dot{z}_{jk}$, are also known.

The aim is to compute the velocity of the basic points *k1* and *k2*, and those of the unit vectors $\mathbf{u}_{k1}$ and $\mathbf{u}_{k2}$. These are given by the following velocity equations:

$$\dot{\mathbf{r}}_{k1} = \dot{\mathbf{r}}_{j2} + \dot{\mathbf{d}}_{jk} \tag{8.60}$$

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_j + \boldsymbol{\omega}_{jk} \tag{8.61}$$

$$\dot{\mathbf{r}}_{k2} = \dot{\mathbf{r}}_{k1} + \boldsymbol{\omega}_k \wedge (\mathbf{r}_{k2} - \mathbf{r}_{k1}) \tag{8.62}$$

$$\dot{\mathbf{u}}_{k1} = \boldsymbol{\omega}_k \wedge \mathbf{u}_{k1} \tag{8.63}$$

$$\dot{\mathbf{u}}_{k2} = \boldsymbol{\omega}_k \wedge \mathbf{u}_{k2} \tag{8.64}$$

Similarly, the accelerations can be obtained from the following equations:

$$\ddot{\mathbf{r}}_{k1} = \ddot{\mathbf{r}}_{j2} + \ddot{\mathbf{d}}_{jk} \tag{8.65}$$

$$\dot{\boldsymbol{\omega}}_k = \dot{\boldsymbol{\omega}}_j + \dot{\boldsymbol{\omega}}_{kj} \tag{8.66}$$

$$\ddot{\mathbf{r}}_{k2} = \ddot{\mathbf{r}}_{k1} + \dot{\boldsymbol{\omega}}_k \wedge (\mathbf{r}_{k2} - \mathbf{r}_{k1}) + \boldsymbol{\omega}_k \wedge (\boldsymbol{\omega}_k \wedge (\mathbf{r}_{k2} - \mathbf{r}_{k1})) \tag{8.67}$$

$$\ddot{\mathbf{u}}_{k1} = \dot{\boldsymbol{\omega}}_k \wedge \mathbf{u}_{k1} + \boldsymbol{\omega}_k \wedge (\boldsymbol{\omega}_k \wedge \mathbf{u}_{k1}) \tag{8.68}$$

$$\ddot{\mathbf{u}}_{k2} = \dot{\boldsymbol{\omega}}_k \wedge \mathbf{u}_{k2} + \boldsymbol{\omega}_k \wedge (\boldsymbol{\omega}_k \wedge \mathbf{u}_{k2}) \tag{8.69}$$

These equations will now be particularized to the different types of kinematic joints.

Revolute joint R. In this case, *k1* coincides with *j2*, and the unit vector $\mathbf{u}_{k1}$ coincides with $\mathbf{u}_j$ and the joint axis. Consequently, the following expressions can be easily obtained:

$$\mathbf{d}_{jk} = \dot{\mathbf{d}}_{jk} = \ddot{\mathbf{d}}_{jk} = 0 \tag{8.70}$$

$$\boldsymbol{\omega}_{jk} = \dot{z}_{jk}\,\mathbf{u}_j \tag{8.71}$$

$$\dot{\boldsymbol{\omega}}_{jk} = \ddot{z}_{jk}\,\mathbf{u}_j + \dot{z}_{jk}\,\boldsymbol{\omega}_j \wedge \mathbf{u}_j \tag{8.72}$$

Prismatic joint P. This joint can be modeled by considering $\mathbf{u}_{k1} \equiv \mathbf{u}_j$ in the direction of the joint which is defined by the points *j2* and *k1*. After this consideration, the following equations define the required velocities and accelerations:

$$\dot{\mathbf{d}}_{jk} = \dot{z}_{jk}\mathbf{u}_j + z_{jk}\boldsymbol{\omega}_j \wedge \mathbf{u}_j \tag{8.73}$$

$$\boldsymbol{\omega}_{jk} = \dot{\boldsymbol{\omega}}_{jk} = 0 \tag{8.74}$$

$$\ddot{\mathbf{d}}_{jk} = \ddot{z}_{jk}\,\mathbf{u}_j + 2\dot{z}_{jk}\,\boldsymbol{\omega}_j \wedge \mathbf{u}_j + z_{jk}\,\dot{\boldsymbol{\omega}}_j \wedge \mathbf{u}_j + z_{jk}\,\boldsymbol{\omega}_j \wedge \left(\boldsymbol{\omega}_j \wedge \mathbf{u}_j\right) \tag{8.75}$$

Cylindrical joint C. This joint can be considered as a combination of a revolute and a prismatic joint. Therefore the following equations yield the required velocities and accelerations:

$$\dot{\mathbf{d}}_{jk} = \dot{z}_{jk}\,\mathbf{u}_j + z^{t}_{jk}\,\boldsymbol{\omega}_j \wedge \mathbf{u}_j \tag{8.76}$$

$$\boldsymbol{\omega}_{jk} = \dot{z}^{r}_{jk}\,\mathbf{u}_j \tag{8.77}$$

$$\ddot{\mathbf{d}}_{jk} = \ddot{z}^{t}_{jk}\,\mathbf{u}_j + 2\dot{z}^{t}_{jk}\,\boldsymbol{\omega}_j \wedge \mathbf{u}_j + z^{t}_{jk}\,\dot{\boldsymbol{\omega}}_j \wedge \mathbf{u}_j + z^{t}_{jk}\,\boldsymbol{\omega}_j \wedge \left(\boldsymbol{\omega}_j \wedge \mathbf{u}_j\right) \tag{8.78}$$

$$\dot{\boldsymbol{\omega}}_{jk} = \ddot{z}^{t}_{jk}\,\mathbf{u}_j + \dot{z}^{t}_{jk}\,\boldsymbol{\omega}_j \wedge \mathbf{u}_j \tag{8.79}$$

Spherical joint S. This joint can be modeled by joining points *j2* and *k1* in the system of Figure 8.11. The relative velocities and accelerations of the joint will be defined by the first and second derivatives of the Euler parameters. This is not strictly necessary for the accelerations, since $\dot{\boldsymbol{\omega}}$ is integrable. The expressions that define the motion of the joint are (Nikravesh (1988) and Haug (1989)):

$$\dot{\mathbf{d}}_{jk} = \ddot{\mathbf{d}}_{jk} = 0 \tag{8.80}$$

$$\boldsymbol{\omega}_{jk} = 2\,\mathbf{G}\,\dot{\mathbf{p}} \tag{8.81}$$

$$\mathbf{G} \equiv \begin{bmatrix} -\mathbf{p}_1 & \mathbf{p}_0 & -\mathbf{p}_3 & \mathbf{p}_2 \\ -\mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_0 & -\mathbf{p}_1 \\ -\mathbf{p}_3 & -\mathbf{p}_2 & \mathbf{p}_1 & \mathbf{p}_0 \end{bmatrix} \tag{8.82}$$

$$\dot{\boldsymbol{\omega}}_{jk} = 2\,\mathbf{G}\,\ddot{\mathbf{p}} \tag{8.83}$$

Universal joint U. The points *j2* and *k1* that belong to this joint are joined together, and the unit vectors $\mathbf{u}_j$ and $\mathbf{u}_{k1}$ that belong to elements *J* and *K*, respectively, are perpendicular to each other. The axes of the two relative rotations z^1_{jk} and z^2_{jk}, are $\mathbf{u}_j$ and $\mathbf{u}_{k1}$, respectively. The equations that define the motion of the joint become:

$$\dot{\mathbf{d}}_{jk} = \ddot{\mathbf{d}}_{jk} = 0 \tag{8.84}$$

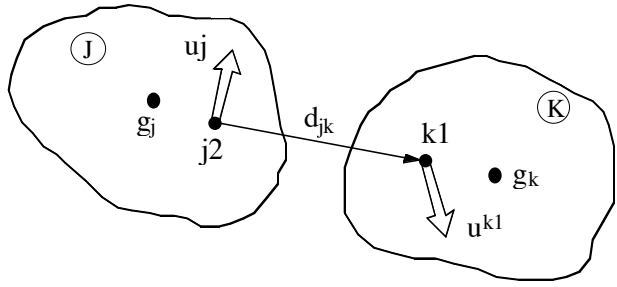


Figure 8.12. Consecutive bodies for recursive velocity calculation (Form. B)

$$\dot{\mathbf{u}}_{k1} = \left(\boldsymbol{\omega}_j + \dot{z}^1_{jk}\,\mathbf{u}_j\right) \wedge \mathbf{u}_{k1} \tag{8.85}$$

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_j + \dot{z}^1_{jk}\,\mathbf{u}_j + \dot{z}^2_{jk}\,\mathbf{u}_{k1} \tag{8.86}$$

$$\dot{\boldsymbol{\omega}}_k = \dot{\boldsymbol{\omega}}_j + \ddot{z}^1_{jk}\,\mathbf{u}_j + \ddot{z}^2_{jk}\,\mathbf{u}_{k1} + \dot{z}^1_{jk}\,\dot{\mathbf{u}}_j + \dot{z}^2_{jk}\,\dot{\mathbf{u}}_{k1} \tag{8.87}$$

8.2.5.2 Formulation B

The main difference between this formulation and the previous one is the kind of dependent variables used to derive the equations of motion. In this formulation, the velocity of the center of gravity and the angular velocity of each body are taken as velocity variables instead of the velocities of points and unit vectors that were used in Formulation A. Using these variables, one can obtain a similar expression of the equations of motion as the one presented in equation (8.38). However, the mass matrix is of size (6x6) with the following expression:

$$\mathbf{M} = \begin{bmatrix} m\,\mathbf{I}_3 & 0 \\ 0 & \mathbf{A}_i\mathbf{J}_i\mathbf{A}_i^T \end{bmatrix} \tag{8.88}$$

An additional term appears in the right-hand side of equation (8.38), to account for the centrifugal terms giving $\mathbf{R}^T(\mathbf{Q}–\mathbf{C})$ instead of $\mathbf{R}^T\mathbf{Q}$ alone.

Consider the general joint of Figure 8.12. The following data is assumed known:

- the position of all the points and the unit vectors,
- the velocity $\dot{\mathbf{g}}_j$ and acceleration $\ddot{\mathbf{g}}_j$ of the center of gravity of body J,
- the angular velocity $\boldsymbol{\omega}_j$ and acceleration $\dot{\boldsymbol{\omega}}_j$ of body J, and,
- the relative velocities $\dot{\mathbf{z}}_{jk}$ and accelerations $\ddot{\mathbf{z}}_{jk}$ of the joint that links bodies J and K.

The velocities of the output points of body J and the input point of body K will be given by:

$$\dot{\mathbf{r}}_{j2} = \dot{\mathbf{g}}_j + \boldsymbol{\omega}_j \wedge \left(\mathbf{r}_{j2} - \mathbf{g}_j\right) \tag{8.89}$$

$$\dot{\mathbf{u}}_j = \boldsymbol{\omega}_j \wedge \mathbf{u}_j \tag{8.90}$$

$$\dot{\mathbf{r}}_{k1} = \dot{\mathbf{r}}_{j2} + \dot{\mathbf{d}}_{jk} \tag{8.91}$$

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_j + \boldsymbol{\omega}_{kj} \tag{8.92}$$

$$\dot{\mathbf{u}}_{k1} = \boldsymbol{\omega}_k \wedge \mathbf{u}_{k1} \tag{8.93}$$

$$\dot{\mathbf{g}}_k = \dot{\mathbf{r}}_{k1} + \boldsymbol{\omega}_k \wedge \left(\mathbf{g}_k - \mathbf{r}_{k1}\right) \tag{8.94}$$

The accelerations will be given by:

$$\ddot{\mathbf{r}}_{j2} = \ddot{\mathbf{g}}_j + \dot{\boldsymbol{\omega}}_j \wedge \left(\mathbf{r}_{j2} - \mathbf{g}_j\right) + \boldsymbol{\omega}_j \wedge \left(\boldsymbol{\omega}_j \wedge \left(\mathbf{r}_{j2} - \mathbf{g}_j\right)\right) \tag{8.95}$$

$$\dot{\boldsymbol{\omega}}_k = \dot{\boldsymbol{\omega}}_j + \dot{\boldsymbol{\omega}}_{kj} \tag{8.96}$$

$$\ddot{\mathbf{r}}_{k1} = \ddot{\mathbf{r}}_{j2} + \ddot{\mathbf{d}}_{jk} \tag{8.97}$$

$$\ddot{\mathbf{g}}_k = \ddot{\mathbf{r}}_{k1} + \dot{\boldsymbol{\omega}}_k \wedge \left(\mathbf{g}_k - \mathbf{r}_{k1}\right) + \boldsymbol{\omega}_k \wedge \left(\boldsymbol{\omega}_k \wedge \left(\mathbf{g}_k - \mathbf{r}_{k1}\right)\right) \tag{8.98}$$

In equations (8.89)-(8.98), the particular values of the variables that define the relative motion of each kind of joint are given by the same equations developed for the Formulation A: equations (8.70) to (8.87).

8.2.6 Element-by-Element Computation of Matrix **R**

The matrix **R** may be obtained *globally* (for all the multibody system) or in an *element-by-element* basis. Each column of matrix **R** represents the result of velocity analysis, in which one computes the dependent velocities corresponding to a unit relative independent velocity (rigid body motion of the base body or relative joint velocity). On the other hand, one may consider separately the rows of matrix **R** that correspond to the dependent velocities of a particular body. In these rows, only the columns corresponding to the independent velocities that affect this body, that is, base body velocities and relative velocities in joints that are located backward in the branch of the body, will contain non-zero elements. This presents very good opportunities for carrying out the computations in parallel.

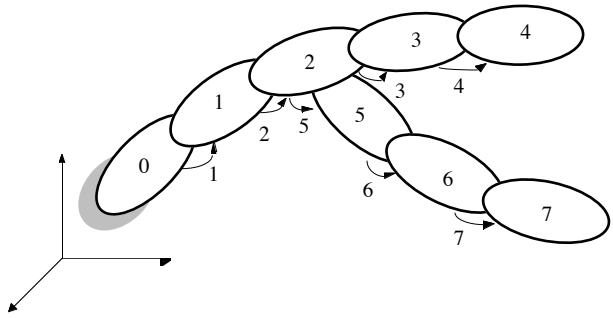


Figure 8.13. Open-chain system with two branches and seven bodies.

Since **R** is a transformation matrix from independent to dependent velocities, **R** may be obtained by either one of the two Formulations A or B, seen in the previous section. The topic of how to obtain **R** using velocity analysis should not present any conceptual difficulty. Formulation B, which seems to be the most efficient method, will be dealt with here. The basic ideas of how to obtain the matrix **R** using Formulation A will be treated through some exercises.

In Section 8.2.2, the physical meaning of the velocity transformation matrix **R** of order ($6n_b \times f$) was explained. In this section, a procedure for the fast parallel computation of matrix **R** will be presented. Recall from Section 8.2.5 that in the Formulation B, the *i-th* column of matrix **R** represents the velocity of the center of gravity and the angular velocity of all bodies, that is vector $\dot{\mathbf{q}}$, when all the relative joint velocities $\dot{\mathbf{z}}$ are zero except the *i-th* component that takes a unit value. Consequently, the columns of matrix **R** can be computed, one at a time, by solving *f* times the velocity problem. The parallelization of the algorithm becomes easier, if an *element-by-element* computational scheme is adopted. This scheme is based on computing 'separately' the rows of **R** corresponding to each body. In this way, most of the computations can be carried out independently for each body and can be computed concurrently.

Example 8.3

Consider the example in Figure 8.13, with seven elements or rigid bodies forming an open-chain with two branches. It is assumed that all the joints have a single degree of freedom, and that the base body is fixed.

In this example, the size of matrix **R** is of (42×7). Suppose that it is necessary to compute the rows of **R** corresponding to body 5. First of all, columns 3, 4, 6, and 7 will be zero, because the unit velocities of joints 3, 4, 6 and 7 will not produce motion in the bodies that are backward in the kinematic chain. Hence, labeling $\mathbf{R}_5$ as the six rows-matrix **R** corresponding to body 5, omitting the zero columns, one can write it as

$$\mathbf{R}_5 = \left[\mathbf{R}_5^1 \, \mathbf{R}_5^2 \, \mathbf{R}_5^5 \right]$$

where $\mathbf{R}_5^i$ represents a (6×1) vector obtained by extracting from $\mathbf{R}$ the rows of column i that correspond to body 5. Therefore, matrix $\mathbf{R}_5$ is of (6×3) size. The number of columns of $\mathbf{R}_j$ in the general case coincides with the number of degrees of freedom found in the path from body J to the base body. Vector $\mathbf{R}_5^i$ can be written in the following form:

$$\mathbf{R}_5^i = \begin{Bmatrix} \dot{\mathbf{g}}_5^i \\ \boldsymbol{\omega}_5^i \end{Bmatrix}$$

where $\dot{\mathbf{g}}_5^i$ and $\boldsymbol{\omega}_5^i$ are, respectively, the velocity of the center of gravity and the angular velocity of body 5, when all the joint relative velocities are zero, except the *i-th* component, which has unit relative velocity.

Generally, the rows of matrix $\mathbf{R}$ corresponding to body b are contained in matrix $\mathbf{R}_b$. The size of matrix $\mathbf{R}_b$ is $(6\times f_b)$, where f_b is the number of degrees of freedom found in the path that goes from body b to the base body including the base body degrees of freedom. Matrix $\mathbf{R}_b$ can be written in the following way:

$$\mathbf{R}_b = \left[\mathbf{R}_b^1\ \mathbf{R}_b^2 \cdots \mathbf{R}_b^{pb}\right] \tag{8.99}$$

where p_b is the number of joints in the path from body b to the base body. Subscripts are used for bodies and superscripts for joints. Each submatrix $\mathbf{R}_b^i$ is of size $(6\times d^i)$, where d^i is the number of degrees of freedom of joint i. In the example of Figure 8.13, d^i takes unit value for all the joints, since all of them were assumed to allow a single degree of freedom.

The computation of matrices $\mathbf{R}_b^i$ for the different types of joints are presented below:

Revolute joint R. Since a revolute joint allows a single degree of freedom, $d^i=1$ in this case. Labeling $\mathbf{u}_i$ to the unit vector that points in direction of the revolute axis, $\mathbf{R}_b^i$ is calculated as

$$\mathbf{R}_b^i = \begin{Bmatrix} \mathbf{u}_i \wedge (\mathbf{g}_b - \mathbf{r}_i) \\ \mathbf{u}_i \end{Bmatrix} \tag{8.100}$$

where $(\mathbf{g}_b - \mathbf{r}_i)$ is the vector that goes from the revolute joint i to the center of gravity of body b.

Prismatic joint P. Again $d^i=1$. Using the same notation, one can obtain

$$\mathbf{R}_b^i = \begin{Bmatrix} \mathbf{u}_i \\ 0 \end{Bmatrix} \tag{8.101}$$

Cylindrical joint C. Since a cylindrical joint allows two degrees of freedom, $d^i=2$; and hence $\mathbf{R}_b^i$ has two columns, that can be written as follows:

$$\mathbf{R}_b^i = \begin{bmatrix} \mathbf{u}_i \wedge (\mathbf{g}_b - \mathbf{r}_i) & \mathbf{u}_i \\ \mathbf{u}_i & 0 \end{bmatrix} \tag{8.102}$$

Spherical joint S. In this case, $d^i=3$. Since the relative angular velocity vector is taken as relative joint velocity, the expression of $\mathbf{R}_b^i$ is

$$\mathbf{R}_b^i = \begin{bmatrix} \mathbf{i} \wedge (\mathbf{g}_b - \mathbf{r}_i) & \mathbf{j} \wedge (\mathbf{g}_b - \mathbf{r}_i) & \mathbf{k} \wedge (\mathbf{g}_b - \mathbf{r}_i) \\ \mathbf{i} & \mathbf{j} & \mathbf{k} \end{bmatrix} \tag{8.103}$$

where **i**, **j**, and **k** are three orthonormal unit vectors parallel to the inertial axes.

Universal joint U. As the cylindrical joint, the universal joint has $d^i=2$. Vectors $\mathbf{u}_{i1}$ and $\mathbf{u}_{i2}$ to the two orthogonal unit vectors that point in the directions of the axes. Hence,

$$\mathbf{R}_b^i = \begin{bmatrix} \mathbf{u}_{i1} \wedge (\mathbf{g}_b - \mathbf{r}_i) & \mathbf{u}_{i2} \wedge (\mathbf{g}_b - \mathbf{r}_i) \\ \mathbf{u}_{i1} & \mathbf{u}_{i2} \end{bmatrix} \tag{8.104}$$

Base body. An unconstrained floating base body has six rigid body degrees of freedom; thus $d^i=6$. As in the spherical joint, the rotational part of matrix **R** is defined from three unit angular velocities in the directions of the inertial axes. Vector $\mathbf{g}_b^i$ is the vector that goes from the center of gravity of the base body (reference point) to the center of gravity of body *b*. Matrix $\mathbf{R}_b^i$ is thus calculated in the following way:

$$\mathbf{R}_b^i = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{i} \wedge \mathbf{g}_b & \mathbf{j} \wedge \mathbf{g}_b & \mathbf{k} \wedge \mathbf{g}_b \\ 0 & 0 & 0 & \mathbf{i} & \mathbf{j} & \mathbf{k} \end{bmatrix} \tag{8.105}$$

where columns one to three correspond to the translational degrees of freedom, and columns four to six to the rotational ones.

The procedure described in this section for the computation of matrix **R** works independently for each of the bodies. Thus, matrices $\mathbf{R}_b$ can be computed concurrently.

8.2.7 Computation of Mass Matrices $\mathbf{M}_b$

As it can be seen in Nikravesh (1988) and Haug (1989), with reference point coordinates the multibody mass matrix **M** can be obtained by assembling the body mass matrices $\mathbf{M}_b$. It is not necessary to obtain explicitly matrix **M**, but only to compute $\mathbf{M}_b$ and perform the required product with matrix $\mathbf{R}_b$.

The body mass matrix $\mathbf{M}_b$ is a (6×6) matrix formed by two (3×3) submatrices on its diagonal. The leading (3×3) submatrix is the unit matrix **I** times the mass of the element. The second (3×3) submatrix $\mathbf{J}_b$, which represents the inertia tensor of body *b* expressed in the inertial reference frame, requires further computation because it is position dependent. Using the well-known tensor transformation expression between two reference frames, one can obtain

$$\mathbf{J}_b = \mathbf{A}_b \, \bar{\mathbf{J}}_b \, \mathbf{A}_b^T \tag{8.106}$$

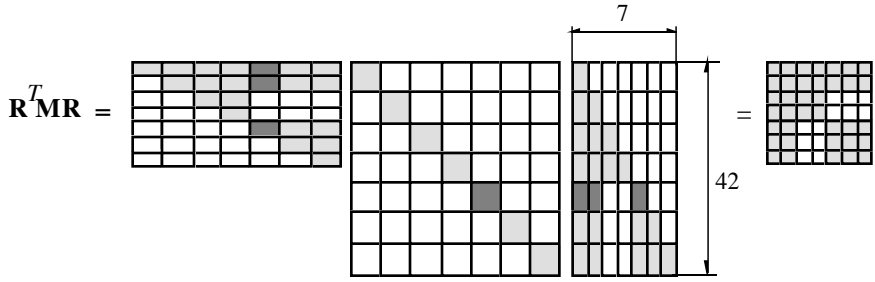


Figure 8.14. Non-zero pattern for the product $\mathbf{R}^T\mathbf{MR}$ in Example 8.3.

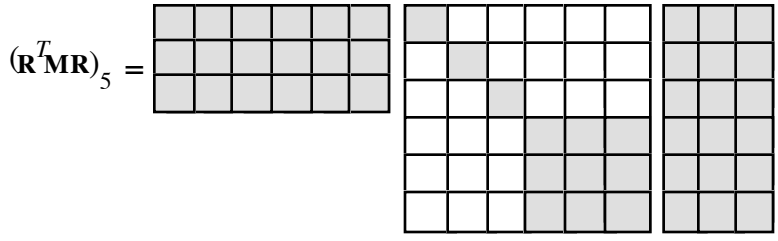


Figure 8.15. Non-zero pattern for the product $\mathbf{R}^T\mathbf{MR}$ for body 5 in Example 8.3.

where matrix $\bar{\mathbf{J}}_b$ is the constant inertia tensor expressed in the moving frame of body *b*.

8.2.8 Computation of the Matrix Product $\mathbf{R}^T\mathbf{MR}$

One of the most attractive features of the choice of dependent coordinates that has been made (reference point coordinates) is that the multibody system mass matrix $\mathbf{M}$ has a block-diagonal structure without coupling terms between contiguous bodies. The immediate application of this property is that the triple matrix product ($\mathbf{R}^T\mathbf{MR}$) that appears in the equations of motion (8.38) can be computed efficiently on an *element-by-element* basis. This means this triple product can be computed as

$$\mathbf{R}^T\ \mathbf{M}\ \mathbf{R} = \sum_{i=1}^{N_b}{}^{*}\ \mathbf{R}_i^T\,\mathbf{M}_i\,\mathbf{R}_i \tag{8.107}$$

where the symbol $\sum^*$ represents the combined action of summation and assembly of the resulting matrix. Due to the fact that there are no coupling terms in matrix $\mathbf{M}$, equation (8.107) shows that the triple product can be performed *ele-*

ment-by-element without increasing the total number of arithmetic operations. This product can be computed independently for each body which allows an easy parallelization that fits perfectly into the whole computational scheme.

With this formulation, there is no need to treat in a special way the *junction bodies*, or bodies that are linked to more than two other bodies. In the example of Figure 8.13, the only junction body is body 2, which is linked to bodies 1, 3, and 5. The term in the summation corresponding to it can be computed exactly in the same way as all the other terms.

If one returns again to the multibody system of Example 8.3 (which is depicted in Figure 8.13), one can symbolically represent the product $\mathbf{R}^T\mathbf{M}\mathbf{R}$ a s shown in Figure 8.14. Submatrices that contain non-zero terms have been represented with a shaded pattern. When the product $\mathbf{M}\mathbf{R}$ is calculated, the distribution of zero and non-zero terms in the resulting matrix coincides with the one in matrix $\mathbf{R}$. This means that the product between $\mathbf{R}^T$ and $\mathbf{M}\mathbf{R}$ involves two matrices with exactly the same structure of matrix $\mathbf{R}$. Then it is possible to obtain the whole product as a summation of individual terms corresponding to each body. The terms that involve body 5, $(\mathbf{R}^T\mathbf{M}\mathbf{R})_5$ have been outlined with a dark shaded pattern and are represented separately in Figure 8.15, once the zero columns of matrix $\mathbf{R}$ have been removed. Again, shaded terms correspond to non-zero elements in the different matrices. The result of this small triple product is a symmetric matrix. Only the terms above the diagonal need to be computed. Finally, the partial result corresponding to body 5 has to be assembled in the final matrix on rows and columns 1, 2, and 5. This computational scheme can be applied to all the bodies in the system, so as to obtain the matrix $\mathbf{R}^T\mathbf{M}\mathbf{R}$ in a very efficient way.

8.2.9 *Computation of the Matrix Product* $\mathbf{R}^T\mathbf{M}\mathbf{S}\mathbf{c}$

Looking at equation (5.65) in Chapter 5, it can be seen that the product $(\mathbf{S}\mathbf{c})$ has a clear physical meaning, that makes it very easy to compute. If one makes the independent accelerations $\ddot{\mathbf{z}}$ equal to zero and then computes the dependent accelerations $\ddot{\mathbf{q}}$, the resulting vector turns out to be $(\mathbf{S}\mathbf{c})$. Therefore, $(\mathbf{S}\mathbf{c})$ represents the vector of dependent accelerations that depends only on the independent velocities. The recursive computation of accelerations has been described in Section 8.2.5. It can be applied to this particular case.

Once the term $(\mathbf{S}\mathbf{c})$ has been computed, the final product $\mathbf{R}^T\mathbf{M}\mathbf{S}\mathbf{c}$ can be computed again in an *element-by-element* basis, recovering the partial product $\mathbf{R}^T\mathbf{M}$ from the previous section.

8.2.10 *Computation of the Term* $\mathbf{R}^T(\mathbf{Q} - \mathbf{C})$

Once more, this product corresponds to the projected external and velocity-dependent inertia forces and does not offer any special difficulty. This product can be computed on an *element-by-element* basis and then conveniently assembled to produce the corresponding forcing term in the equations of motion.

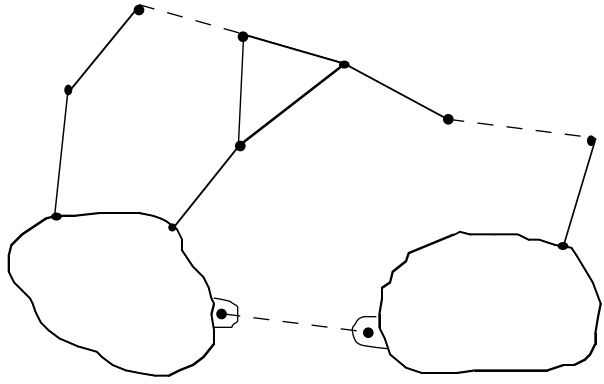


Figure 8.16. Opening a closed-chain system by removing rigid body constraints.

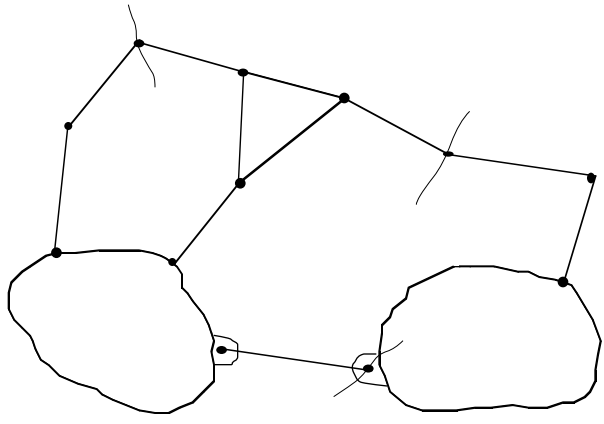


Figure 8.17. Opening a closed-chain system by removing joint constraints.

8.3 Velocity Transformations for Closed-Chain Systems

It was shown in Section 5.3 how a closed-chain multibody system can be transformed to an open-chain by simply eliminating certain constraint equations that enforce the closure of the loops.

Two different formulations, A and B, have been introduced in Section 8.2.5. Which one is used depends on how the velocities and accelerations are represented. Formulation A considers as variables the *natural coordinates*, that is, the Cartesian components of basic points and unit vectors. The constraint equations that these coordinates generate (See Chapter 2) are primarily due to the rigid body conditions. Therefore, the way to proceed in this case to open the kinematic chain is by *eliminating certain rigid body conditions* of some of the

elements of the multibody system. Figure 8.16 illustrates how this procedure may be applied to a complex system using the concepts just explained.

Formulation B considers that the dependent velocities of each element are characterized by the velocity of the center of gravity and by the angular velocity of the element. In this case, the constraint equations (See Chapter 2) are originated not at the elements but at the kinematic joints. Therefore, in order to open a loop, one needs to *cut a joint* by removing the corresponding constraints. Figure 8.17 illustrates this procedure.

Regardless of what formulation is used to represent the dependent velocities and accelerations of the multibody system, it is always possible to divide the constraint equations into two major groups: the first, that will be represented by the superscript 1, is formed by the constraints of the open-chain multibody system that result from the opening of the kinematic loops; the second, that in the sequel will be represented by the superscript 2, will be formed by those constraints needed to close the loops previously opened. Consequently, the velocity constraint equations become

$$\begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{q}}^{1} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{2} \end{bmatrix} \{ \dot{\mathbf{q}} \} = \{ \mathbf{b} \} \equiv - \begin{Bmatrix} \boldsymbol{\Phi}_{t}^{1} \\ \boldsymbol{\Phi}_{t}^{2} \end{Bmatrix} \tag{8.108}$$

where $\mathbf{b}=\mathbf{0}$, if the constraints are scleronomous. Let m_1 and m_2 be the number of rows of $\boldsymbol{\Phi}_{\mathbf{q}}^{1}$ and $\boldsymbol{\Phi}_{\mathbf{q}}^{2}$, respectively. Assume that $m_1 >> m_2$ (m_1 much greater than m_2), since the opening of the loops will be done by removing only a few constraint equations.

If the number of dependent coordinates is n, the number of degrees of freedom of the open- and closed-chain subdivisions will be $f_1=n-m_1$ and $f_2=n-m_1-m_2$, respectively.

The key point in this formulation is the fact that the matrix $\mathbf{R}^1$, which defines a basis for the nullspace of $\boldsymbol{\Phi}_{\mathbf{q}}^{1}$, *can be directly obtained* by the procedure explained in Section 8.2 without forming and triangularizing the Jacobian matrix $\boldsymbol{\Phi}_{\mathbf{q}}^{1}$. This leads to important savings in computational costs. Even though the matrix $\boldsymbol{\Phi}_{\mathbf{q}}^{1}$ is never formed explicitly, the following relationships will still be satisfied:

$$\boldsymbol{\Phi}_{\mathbf{q}}^{1} \mathbf{R}^{1} = 0 \tag{8.109}$$

$$\dot{\mathbf{q}} = \mathbf{R}^{1} \dot{\mathbf{z}}^{1} \tag{8.110}$$

$$\ddot{\mathbf{q}} = \mathbf{R}^{1} \ddot{\mathbf{z}}^{1} + \mathbf{S}^{1} \mathbf{c}^{1} \tag{8.111}$$

where the vector $\mathbf{z}^1$ is formed by the base body and relative joint coordinates of the open-chain system. Now, in the closed-chain system, these coordinates $\mathbf{z}^1$ are not independent, because they are interrelated through the constraints $\boldsymbol{\Phi}^2$. The problem is that the constraints $\boldsymbol{\Phi}^2$ are not written in terms of $\mathbf{z}^1$ but in terms of $\mathbf{q}$. However, this problem may be easily solved as follows.

The equations of motion seen in Chapter 5 that are based on the Lagrange multipliers technique will be considered. These equations, applied to the problem at hand, become

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^{1\mathrm{T}} & \boldsymbol{\Phi}_{\mathbf{q}}^{2\mathrm{T}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{1} & 0 & 0 \\ \boldsymbol{\Phi}_{\mathbf{q}}^{2} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda}^{1} \\ \boldsymbol{\lambda}^{2} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q} \\ \mathbf{c}^{1} \\ \mathbf{c}^{2} \end{Bmatrix} \tag{8.112}$$

where $\boldsymbol{\lambda}^1$ and $\boldsymbol{\lambda}^2$ are the multipliers that correspond to the partitions $\boldsymbol{\Phi}_{\mathbf{q}}^1$ and $\boldsymbol{\Phi}_{\mathbf{q}}^2$, respectively. The vector $\mathbf{Q}$ includes the external as well as all the velocity dependent inertia forces.

Substituting equation (8.111) into equation (8.112) and pre-multiplying the first row of equation (8.111) by $(\mathbf{R}^1)^{\mathrm{T}}$, one obtains

$$\begin{bmatrix} \mathbf{R}^{1\mathrm{T}}\mathbf{M}\,\mathbf{R}^{1} & \mathbf{R}^{1\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{q}}^{1\mathrm{T}} & \mathbf{R}^{1\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{q}}^{2\mathrm{T}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{1}\mathbf{R}^{1} & 0 & 0 \\ \boldsymbol{\Phi}_{\mathbf{q}}^{2}\mathbf{R}^{1} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{z}}^{1} \\ \boldsymbol{\lambda}^{1} \\ \boldsymbol{\lambda}^{2} \end{Bmatrix} = \begin{Bmatrix} \mathbf{R}^{1\mathrm{T}}\mathbf{Q} - \mathbf{R}^{1\mathrm{T}}\mathbf{M}\,\mathbf{S}^{1}\mathbf{c}^{1} \\ \mathbf{c}^{1} - \boldsymbol{\Phi}_{\mathbf{q}}^{1}\mathbf{S}^{1}\mathbf{c}^{1} \\ \mathbf{c}^{2} - \boldsymbol{\Phi}_{\mathbf{q}}^{2}\mathbf{S}^{1}\mathbf{c}^{1} \end{Bmatrix} \tag{8.113}$$

However, introducing equation (8.109), the second row and column of equation (8.113) cancels out. It follows that the coefficient of $\boldsymbol{\lambda}^1$ vanishes and the following equation is obtained:

$$\begin{bmatrix} \mathbf{R}^{1\mathrm{T}}\mathbf{M}\,\mathbf{R}^{1} & \mathbf{R}^{1\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{q}}^{2\mathrm{T}} \\ \boldsymbol{\Phi}_{\mathbf{q}}^{2}\mathbf{R}^{1} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{z}}^{1} \\ \boldsymbol{\lambda}^{2} \end{Bmatrix} = \begin{Bmatrix} \mathbf{R}^{1\mathrm{T}}\mathbf{Q} - \mathbf{R}^{1\mathrm{T}}\mathbf{M}\,\mathbf{S}^{1}\mathbf{c}^{1} \\ \mathbf{c}^{2} - \boldsymbol{\Phi}_{\mathbf{q}}^{2}\mathbf{S}^{1}\mathbf{c}^{1} \end{Bmatrix} \tag{8.114}$$

The new mass matrix is of size $(f_1 \times f_1)$, instead of $(n \times n)$ as was the original mass matrix $\mathbf{M}$. The new projected Jacobian matrix ($\boldsymbol{\Phi}_{\mathbf{z}^1}=\boldsymbol{\Phi}_{\mathbf{q}}^2\mathbf{R}^1$) is of size $(m_2 \times f_1)$. Since $m_2 << m_1$ and $f_1 << n$, this new Jacobian matrix is much smaller than the original in equation (8.112), that had a dimension of $((m_1+m_2)\times n)$.

The matrix transformations implied in equation (8.114) may be performed in an *element-by-element* basis and thus can be parallelized in an optimal manner.

Therefore, the reduced system of equations (8.114) has a size much smaller than that of system (8.112). The equations of motion (8.114) may be solved by either one of the following methods, studied in detail in Chapter 5:

a) *Lagrange multipliers*. The system of equations (8.114) can be solved as is or including the Baumgarte stabilization terms, as explained in Section 5.1.

b) *Penalty formulation*. The application of the penalty formulation is straightforward. By introducing $\boldsymbol{\Phi}_{\mathbf{z}^1} = \boldsymbol{\Phi}_{\mathbf{q}}^2\mathbf{R}^1$ and $\mathbf{Q}_{\mathbf{z}^1} = \mathbf{R}^{1\mathrm{T}}(\mathbf{Q}-\mathbf{M}\mathbf{S}^1\mathbf{c}^1)$, this formulation (See equation (5.37)) leads to

$$\begin{aligned} \left(\mathbf{R}^{1\mathrm{T}}\mathbf{M}\mathbf{R}^{1} + \alpha\ \boldsymbol{\Phi}_{\mathbf{z}^1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{z}^1}\right)\ddot{\mathbf{z}}^{1} = \mathbf{Q}_{\mathbf{z}^1} - \alpha\,\boldsymbol{\Phi}_{\mathbf{z}^1}^{\mathrm{T}}\Big(\dot{\boldsymbol{\Phi}}_{\mathbf{z}^1}^{\mathrm{T}}\dot{\mathbf{z}}^{1} - \dot{\boldsymbol{\Phi}}_{t}(\mathbf{z}^{1}) + \\ + 2\Omega\,\mu\,\dot{\boldsymbol{\Phi}}(\mathbf{z}^{1}) + \Omega^{2}\,\boldsymbol{\Phi}(\mathbf{z}^{1})\Big) \end{aligned} \tag{8.115}$$

c) *Independent coordinates*. The kinematic part of (8.114) can be written in the following form:

$$\boldsymbol{\Phi}_{\mathbf{z}^1}\,\ddot{\mathbf{z}}^1 = \bar{\mathbf{c}}^{\,2} \tag{8.116}$$

where $\boldsymbol{\Phi}_{\mathbf{z}^1} = \boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{R}^1$ and $\bar{\mathbf{c}}^2 = \mathbf{c}^2 - \boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{S}^1\mathbf{c}^1$. One can always choose a subset of independent accelerations $\ddot{\mathbf{z}}$ from the vector $\ddot{\mathbf{z}}^1$. Using the method of the Boolean matrix $\mathbf{B}$ (See Section 3.3), the following relation is satisfied:

$$\mathbf{B}\,\ddot{\mathbf{z}}^1 = \ddot{\mathbf{z}} \tag{8.117}$$

By simply joining equations (8.116) and (8.117), the following equation is obtained:

$$\begin{bmatrix} \boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{R}^1 \\ \mathbf{B} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{z}}^1 \end{Bmatrix} = \begin{Bmatrix} \bar{\mathbf{c}}^2 \\ \ddot{\mathbf{z}} \end{Bmatrix} \tag{8.118}$$

If the unit values of the matrix $\mathbf{B}$ have been chosen according to the pivot structure of the matrix $\boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{R}^1$, the leading matrix of equation (8.118) can be inverted to yield

$$\begin{Bmatrix} \ddot{\mathbf{z}}^1 \end{Bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{R}^1 \\ \mathbf{B} \end{bmatrix}^{-1} = \begin{Bmatrix} \bar{\mathbf{c}}^2 \\ \ddot{\mathbf{z}} \end{Bmatrix} \equiv \mathbf{S}^2\,\bar{\mathbf{c}}^2 + \mathbf{R}^2\,\ddot{\mathbf{z}} \tag{8.119}$$

Equation (8.119) leads to the definition of the matrices $\mathbf{R}^2$ and $(\mathbf{S}^2\bar{\mathbf{c}}^2)$. The columns of the matrix $\mathbf{R}^2$ constitute a basis of the nullspace of the projected Jacobian matrix $\boldsymbol{\Phi}_{\mathbf{z}^1}=\boldsymbol{\Phi}^2_{\mathbf{q}}\mathbf{R}^1$. Substituting equation (8.119) into equation (8.114) and pre-multiplying by $\mathbf{R}^{2\mathrm{T}}$, one obtains

$$\begin{aligned} &\mathbf{R}^{2\mathrm{T}}\,\mathbf{R}^{1\mathrm{T}}\,\mathbf{M}\,\mathbf{R}^1\,\mathbf{R}^2\,\ddot{\mathbf{z}} = \\ &= \mathbf{R}^{2\mathrm{T}}\left(\mathbf{R}^{1\mathrm{T}}\left(\mathbf{Q} - \mathbf{M}\,\mathbf{S}^1\,\mathbf{c}^1\right) - \mathbf{R}^{1\mathrm{T}}\,\mathbf{M}\,\mathbf{R}^1\,\mathbf{S}^2\,\bar{\mathbf{c}}^2\right) \end{aligned} \tag{8.120}$$

which is the final equation of motion in independent coordinates.

While the matrix $\mathbf{R}^1$ (of size $(n\times f_1)$) corresponding to $\boldsymbol{\Phi}^1_{\mathbf{q}}$, has been *directly obtained without forming and triangularizing* $\boldsymbol{\Phi}^1_{\mathbf{q}}$, the matrix $\mathbf{R}^2$ (of order $(f_1\times f_2)$) has to be computed *numerically*. The size of $\mathbf{R}^2$ can be significantly smaller than $\mathbf{R}^1$ as may be seen in Figure 8.18, where the sizes of the matrices involved in this formulation are illustrated for a typical case.

Table 8.2 shows the scheme of a numerical algorithm applicable to closed-loop multibody systems using the method presented in this section, with the Lagrange multipliers version.

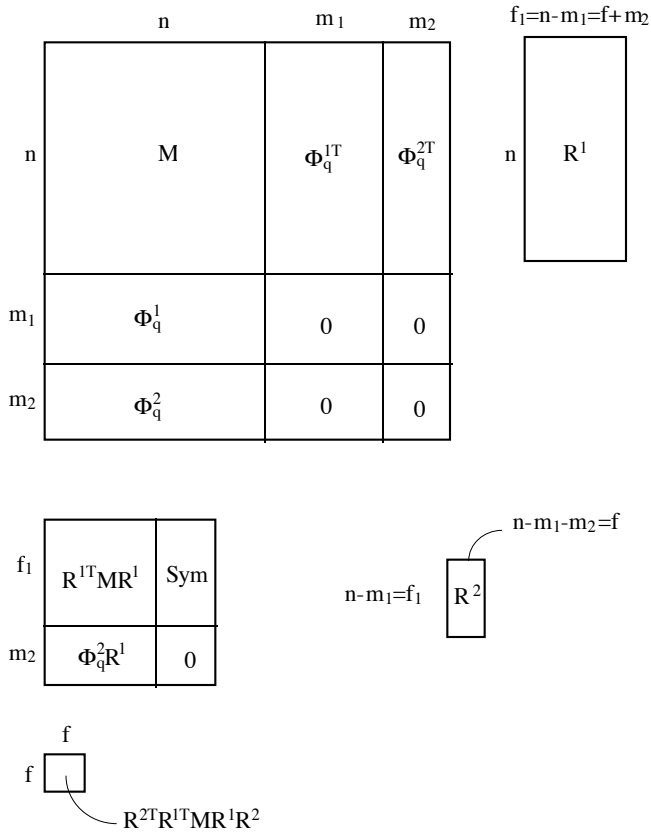


Figure 8.18. Matrix pattern for the method based on velocity transformations.

It may be seen that dealing with closed-chains does not complicate the formulation very much. Thus a good numerical efficiency and a very simple implementation can be expected.

8.4 Examples Solved by Velocity Transformations

In Sections 8.2 and 8.3, the application of velocity transformations to open- and closed-chain kinematic chains has been presented. In this section, two rather complex theoretical examples of the above methods will be described in detail. Afterwards, some numerical results will be presented to demonstrate the efficiency of these formulations.

Table 8.2. Algorithm to formulate and integrate the equations of motion of an closed-chain system, by the Lagrange multipliers method.

Step	Data	Result	Mode
1	$\mathbf{z}_1$	$\overline{\mathbf{q}}$	recursive
2	$\overline{\mathbf{q}}$	$\mathbf{R}^1$	e-by-e or rec.
3	$\dot{\mathbf{z}}_1, \overline{\mathbf{q}}$	$\dot{\mathbf{q}}$	recursive
4	$\dot{\mathbf{z}}_1, \overline{\mathbf{q}}, \dot{\mathbf{q}}$	$(\mathbf{S}^1\mathbf{c}^1)$	e-by-e
5	$\mathbf{R}^1, \mathbf{M}$	$\mathbf{R}^{1\mathrm{T}}\ \mathbf{M}\ \mathbf{R}^1$	e-by-e
6	$\mathbf{R}^1, \mathbf{Q}$	$\mathbf{R}^{1\mathrm{T}}\ \mathbf{Q}$	e-by-e
7	$\mathbf{R}^1, \mathbf{S}^1\mathbf{c}^1, \mathbf{M}$	$\mathbf{R}^{1\mathrm{T}}\ \mathbf{M}\ \mathbf{S}^1\mathbf{c}^1$	e-by-e
8	$\overline{\mathbf{q}}, \dot{\mathbf{q}}$	$\mathbf{\Phi}^2_{\mathbf{q}}, \mathbf{c}^2$	e-by-e
9	$\mathbf{\Phi}^2_{\mathbf{q}}, \mathbf{R}^1$	$\mathbf{\Phi}^2_{\mathbf{q}}\,\mathbf{R}^1$	e-by-e
10		$\mathbf{c}^2 - \mathbf{\Phi}^2_{\mathbf{q}}\mathbf{S}^1\mathbf{c}^1$	e-by-e
11	linear equations	$\ddot{\mathbf{z}}_1$	global
12	$(\ddot{\mathbf{z}}_1, \dot{\mathbf{z}}_1)_t$	$(\dot{\mathbf{z}}_1, \mathbf{z}_1)_{t+h}$	global
	GO TO 1		

8.4.1 Open-Chain Example: Human Body

Figure 8.19 shows an interesting example of open kinematic chain: a dynamic model of the human body with 40 rigid bodies and 45 degrees of freedom. All joints are of the revolute type. Some of them are defined by sharing one point and one unit vector and others are defined by sharing two points.

This mechanical model of the human body has been used for some time in several studies on sport biomechanics, as the ones shown in Figures 1.5 and 1.6 which include also realistic geometric models.

Table 8.3 summarizes the resulting theoretical number of floating-point arithmetic operations both for Formulations A and B. It may be seen that important savings can be obtained with Formulation B. The most important conclusion is that, according to the number of floating-point operations of Formulation B, a 6 Mflops computer would be enough to get a function evaluation every 10 msec. This CPU performance is available in many RISC low-cost workstations.

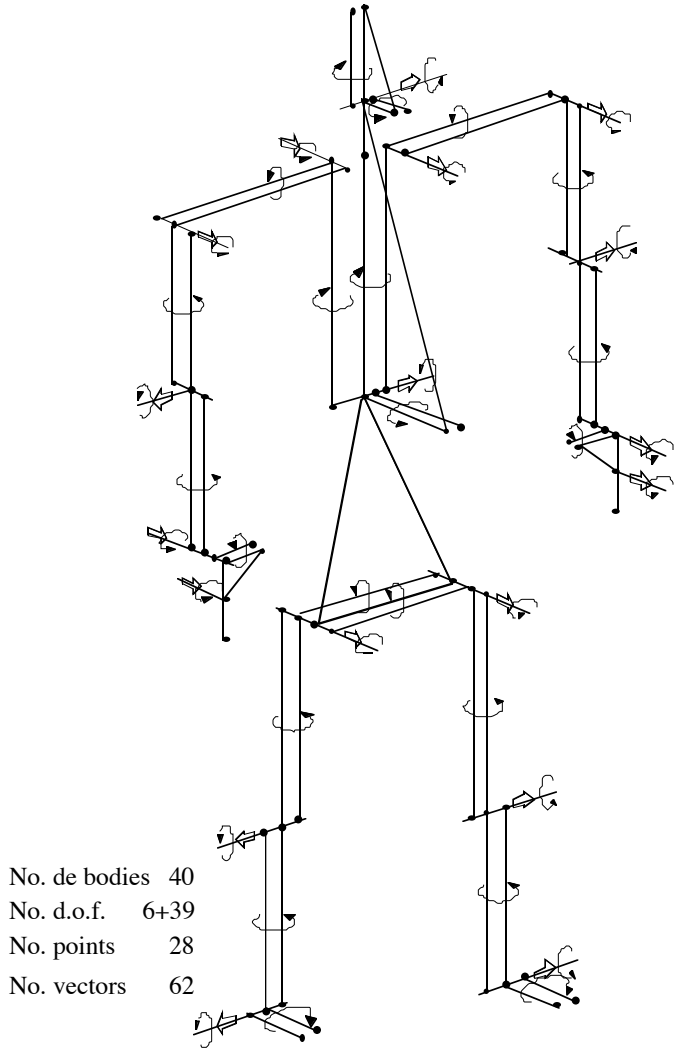
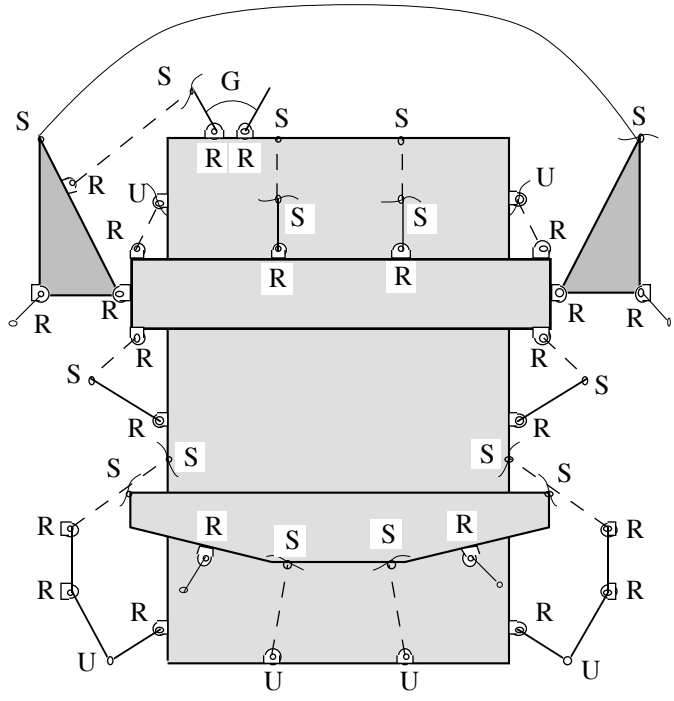


Figure 8.19. Mechanical model of the human body.

Many additional unit vectors (not represented in Figure 8.19) have been introduced in Formulation A. This was done in order for the model to have constant mass matrices with natural coordinates.

8.4.2 Closed-Chain Example: Heavy Truck

Figure 8.20 shows the scheme of a heavy truck suspension and steering system. This system consists of the chassis, the front axle, the rear axle, the leaf-springs modeled by four articulated rigid bodies, the front and rear stabilization bars, the



No. of rigid bodies:	33 (21 + 12)
Joints:	23R + 15S +6U + 1G
Open-loop d.o.f.:	39
No. d.o.f.:	18
Dependent coordinates:	285 (246 + 39)

Figure 8.20. Mechanical model of a heavy truck.

elements of the steering system, and the wheels. There are 33 rigid bodies and 18 degrees of freedom. With Formulation A the loops have been opened by removing the constraint equations corresponding to the elements represented by dashed lines. In this case there are 39 open-chain degrees of freedom or relative coordinates.

With formulation B, some joints have been cut to obtain an open-chain system with 48 degrees of freedom. The cut joints are shown in Figure 8.20 with a line crossing the corresponding joint. Table 8.4 presents the theoretical count of floating-point operations for both Formulations. Again important savings are obtained with Formulation B. In spite of the closed loops of this system, it needs less arithmetic operations than the human model of Figure 8.19, because the branches of the open-chain model of the truck are shorter than in the human body.

Table 8.3. Number of floating-point arithmetic operations for the human body (Formulations A and B).

Function	Mode	Formulation A	Formulation B
$\mathbf{z} \rightarrow \mathbf{q}$	Glob-rec.	1820m + 859a	1560m + 960a
$\dot{\mathbf{z}} \rightarrow \dot{\mathbf{q}}$	Glob-rec.	615m + 441a	360m + 360a
$\ddot{\mathbf{q}}\,\vert_{\ddot{\mathbf{z}}=0} = \mathbf{S}\ \mathbf{c}$	Glob-rec.	1089m + 915a	840m + 840a
$\mathbf{M}$	e-by-e	--	1080m + 480a
$\mathbf{c}$	e-by-e	--	600m + 360a
$\mathbf{R}$	e-by-e	4716m + 3537a	1050m + 525a
$\mathbf{R}^{\mathrm{T}}\mathbf{M}\mathbf{R}$	e-by-e	25824m + 22142a	7482m + 5285a
$\mathbf{R}^{\mathrm{T}}\mathbf{M}\mathbf{S}\mathbf{c}$	e-by-e	3060m + 2885a	1050m + 875a
$\mathbf{R}^{\mathrm{T}}(\mathbf{Q} - \mathbf{c})$	e-by-e	3060m + 2885a	1050m + 995a
$\boldsymbol{\phi}_{\mathbf{q}}^{2}\mathbf{R}$	e-by-e	--	--
$\mathbf{c}_2 - \boldsymbol{\phi}_{\mathbf{q}}^{2}\mathbf{S}\,\mathbf{c}$	e-by-e	--	--
Assembly	e-by-e	3918a	700a
Linear eqs.	Global	16710m + 16170a	16710m + 16170a
TOTAL		56894m + 53752a	31782m + 27550a

Table 8.4. Number of floating-point arithmetic operations for a heavy truck (Formulations A and B).

Function	Mode	Formulation A	Formulation B
$\mathbf{z} \rightarrow \mathbf{q}$	Glob-rec.	1500m + 1222a	1170m + 720a
$\dot{\mathbf{z}} \rightarrow \dot{\mathbf{q}}$	Glob-rec.	507m + 627a	585m + 360a
$\ddot{\mathbf{q}}\,\vert_{\ddot{\mathbf{z}}=0} = \mathbf{S}\ \mathbf{c}$	Glob-rec.	561m + 492a	870m + 540a
$\mathbf{M}$	e-by-e	--	837m + 372a
$\mathbf{c}$	e-by-e	--	465m + 279a
$\mathbf{R}$	e-by-e	960m + 726a	306m + 153a
$\mathbf{R}^{\mathrm{T}}\mathbf{M}\mathbf{R}$	e-by-e	25890m + 20893a	1947m + 1394a
$\mathbf{R}^{\mathrm{T}}\mathbf{M}\mathbf{S}\mathbf{c}$	e-by-e	4947m + 4120a	306m + 255a
$\mathbf{R}^{\mathrm{T}}(\mathbf{Q} - \mathbf{c})$	e-by-e	2040m + 1870a	306m + 348a
$\boldsymbol{\phi}_{\mathbf{q}}^{2}\mathbf{R}$	e-by-e	2220m + 1850a	1224m + 1374a
$\mathbf{c}_2 - \boldsymbol{\phi}_{\mathbf{q}}^{2}\mathbf{S}\,\mathbf{c}$	e-by-e	432m + 432a	108m + 108a
Assembly	e-by-e	897a	900a
Linear eqs.	Global	16000m + 16000a	16000m + 16000a
TOTAL		55084m + 49129a	24124m + 23406a

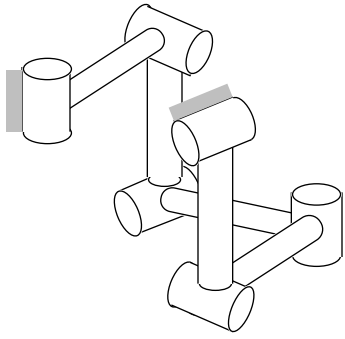


Figure 8.21. Bricard mechanism.

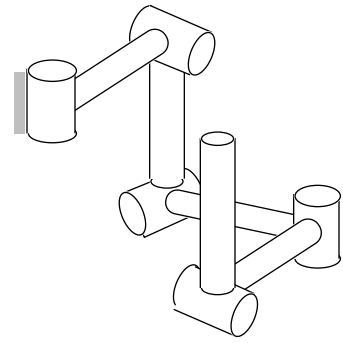


Figure 8.22. Five-bar pendulum.

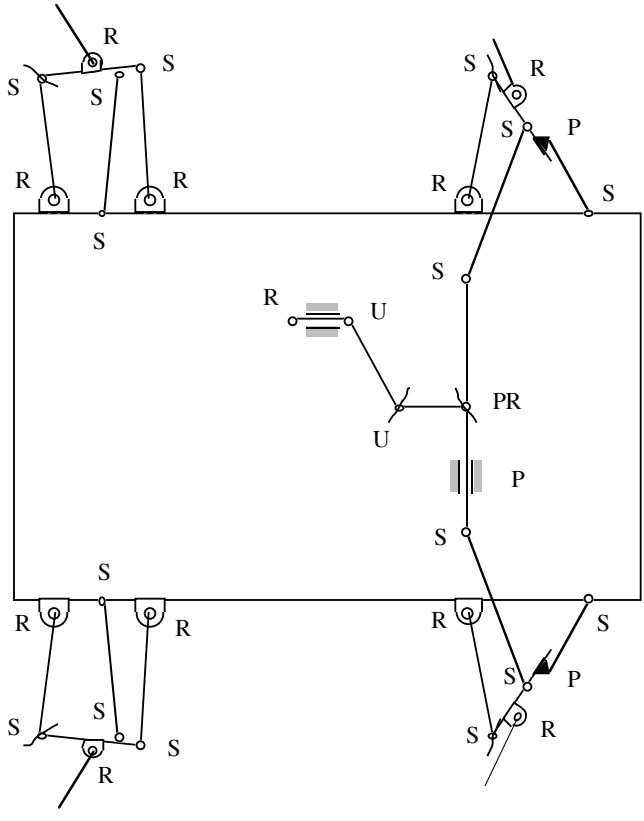


Figure 8.23. Multibody model of a car suspension and steering system.

Table 8.5. Comparative results in CPU milliseconds for function evaluation.

	Penalty method (Section 5.1.4)	Matrix R (Section 5.2.3)	Formulation B (Section 8.2.5.2)
Bricard	8,01	8,73	1,33
Five-bar pendulum	12,00	18,65	1,29
Car model	87,79	224,54	12,21
Human body	91,30	942,64	23,39

8.4.3 Numerical Results

Finally, some comparative numerical results obtained with the Formulation B, and with two other standard dynamic formulations explained in Chapter 5 will be presented. Four examples will be considered. Two of them are very simple. The other two present some degree of complexity.

Figures 8.21 and 8.22 show two simple three-dimensional multibody systems with five bodies and revolute joints only. Figure 8.21 shows the Bricard mechanism which is an over-constrained closed-chain system. According to the Grübler criterion, this mechanism has zero degrees of freedom. Because of the particular orientation of the axes in the revolute joints, it actually has one degree of freedom. Figure 8.22 illustrates a five-bar three-dimensional pendulum that can be obtained directly from the Bricard mechanism by opening the chain through the removal of one of the fixed revolute joints. This pendulum has five degrees of freedom.

Figure 8.23 shows a car model that includes the chassis, the steering and the suspension system. The front suspension is a McPherson type and the rear suspension is based on the five-point system. The complete model has 25 rigid bodies with general mass and inertia properties and 15 degrees of freedom. This car model corresponds to the more realistic model shown in Figure 1.2 of Chapter 1. The fourth example is based on the 45 bodies and 45 degrees of freedom human body model shown in Figure 8.19 which constitutes the basis of the models presented in Figures 1.5 and 1.6 of Chapter 1.

These four examples with different topology and degree of complexity have been analyzed using three dynamic formulations: a) the penalty method, based on dependent coordinates (Section 5.1.4); b) the method of independent coordinates based on the matrix **R** (Section 5.2.3); and c) the Formulation B previously explained in this chapter (Section 8.2.5.2).

Table 8.5 contains the CPU time per function evaluation (computation of accelerations from positions and velocities) expressed in milliseconds for the four examples and using the three different formulations. The results have been obtained with a SG workstation using a MIPS 3000 processor rated at 25 MHz

(3.9 DP Mflops in the Linpack test). Taking into account that there are currently newer low-cost workstations that are much more powerful, it may be concluded from the results shown in Table 8.5 that real time dynamic simulation is at hand for these systems, at least from the point of view of fast function evaluation.

These results do not provide an idea of the absolute efficiency of each method, since the total time of integration depends not only on the time required per function evaluation, but also on the type of numerical method used for the integration of the equations of motion. Standard ODE integrators, such as the DE Shampine and Gordon routine, require less function evaluations with the matrix **R** or Formulation B than with the penalty method. The penalty method leads to stiffer equations and works more efficiently in conjunction with more stable integrators (See Section 8.5). In order to get the best possible response, the concepts developed in Chapters 7 and 8 must be jointly considered along with the particular physical characteristics of the case at hand.

8.5 Special Implementations Using Dependent Natural Coordinates

So far in this chapter dynamic formulations have been considered that, although they may use dependent coordinates to define the motion of the system, try to ultimately solve the equations of motion through a minimum set of independent (in the case of open-chain systems) or dependent coordinates (in closed chain systems). In our search for formulations suitable for real time analysis, we present in this section a method proposed by Bayo et al. (1991) which is an alternative to the methods seen already in this chapter. This method is based on formulating and solving the equations of motion with respect to a full set of dependent coordinates without intermediate transformations. The constraints are considered via a modification of the penalty formulation seen in Chapter 5. The numerical integration is carried out using the trapezoidal rule (See Chapter 7) with the positions, rather than accelerations, as primary variables.

Among the possible sets of dependent coordinates, the natural coordinates (explained in Chapter 2) provide the best setting. They present important advantages over other possible sets: first, the use of elements with two basic points and two unit vectors leads to a constant mass matrix in the inertial frame and to the absence of velocity dependent (centrifugal and Coriolis) inertia forces in the formulation; and secondly, the natural coordinates originate quadratic constraint equations that yield linear Jacobian matrices. As clearly explained in Chapter 2, these Jacobian matrices can be evaluated by a number of arithmetic operations that is considerably lower than those required by other types of coordinates.

Other choices of 12 Cartesian components that define the position and orientation of the rigid body will also lead to a constant mass matrix, such as one point and three Cartesian vectors, four points, and so forth. The choice of two points and two unit vectors is due to the fact that these variables can be shared

by contiguous elements; thus leading to a lesser number of dependent coordinates that will represent the multibody system.

8.5.1 Differential Equations of Motion in the Natural Coordinates

A dynamic system shall be considered whose configuration is characterized by a vector $\mathbf{q}$ of n fully Cartesian coordinates that satisfy m constraint conditions $\mathbf{\Phi}=\mathbf{0}$. Let $L=T-V$ be the *Lagrangian* function of the system, where T and V are the kinetic and potential energies, respectively.

In order to introduce the constraint conditions, a penalty formulation slightly different from that explained in Section 5.2 will be used, that will also lead to a set of ordinary differential equations (ODE), and will guarantee the fulfillment of the constraint equations. In order to introduce these constraints, a fictitious potential is added to the expression of the Hamilton's principle:

$$V^* = \frac{1}{2}\,\mathbf{\Phi}^{\mathrm{T}}\,\boldsymbol{\alpha}\,\mathbf{\Phi} \tag{8.121}$$

and a set of Rayleigh's dissipative forces

$$\mathbf{G}^* = -\boldsymbol{\alpha}\,\boldsymbol{\mu}\,\dot{\mathbf{\Phi}} \tag{8.122}$$

The penalty factors α_k, $k= 1,2,... m$ (with each constraint having a different factor) are large real numbers that represent the spring values of the physical system associated with the constraint $\Phi_k = 0$, and $(\alpha_k \mu_k)$ represents the damping characteristics. Similar to the development already carried out in Chapter 5, the application of the Lagrange's equations directly leads to

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\alpha}\left(\mathbf{\Phi} + \boldsymbol{\mu}\,\dot{\mathbf{\Phi}}\right) = \mathbf{Q} \tag{8.123}$$

where $\mathbf{\Phi_q}$ is the *(m×n)* Jacobian matrix of the constraint equations, $\mathbf{M}$ the mass matrix, and $\mathbf{Q}$ the external forces plus those coming from a potential acting on the system. The matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ are *(m×m)* diagonal matrices that contain the values of the penalty numbers and damping coefficients. If the same values are used for all the constraints these matrices become identity matrices multiplied by constant factors. Remember that the *natural coordinates* described above yield a constant mass matrix. In addition, neither Coriolis nor centrifugal forces are present in the vector $\mathbf{Q}$. Consequently, the only nonlinear components in (8.123) except for position-dependent forces are the penalty terms which physically represent the forces necessary to maintain the constraint conditions. The product $\boldsymbol{\alpha}(\mathbf{\Phi}+\boldsymbol{\mu}\dot{\mathbf{\Phi}})$ represents an approximation to the Lagrange multipliers that arise in the classical formulation.

The nonlinear system (8.123) without the velocity constraints and the penalty system only composed of springs is merely stable. Depending on the type of integrator used, the numerical integration may lead to numerical instabilities in long simulations when using large time steps. These problems disappear when

the velocity constraints consisting of fictitious dissipative forces are included in the formulation as done in (8.123), or when the numerical integrator supplies artificial numerical damping (for example the α-*method* of Hilber, Hughes, and Taylor seen in Chapter 7).

Augmented Lagrangian formulation. Using the integration procedure described below and penalty factors equal to 10^7 in double precision arithmetic, the solution of (8.123) can be achieved with an excellent satisfaction of the constraints. In case a wider range of penalty values is desired, a correcting scheme can be introduced by means of an augmented Lagrangian formulation also seen in Chapter 5. These methods assure convergence within the desired constraint tolerances without the need for very large penalty factors, thereby avoiding ill-conditioning problems.

The expression of the equations of motion can be augmented by adding the Lagrange multipliers as follows:

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\alpha}\left(\mathbf{\Phi} + \boldsymbol{\mu}\,\dot{\mathbf{\Phi}}\right) + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\lambda}^{*} = \mathbf{Q} \tag{8.124}$$

It may be seen by comparing equation (8.124) with the classical Lagrange multipliers technique (equation (5.8)) that

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}^{*} + \boldsymbol{\alpha}\left(\mathbf{\Phi} + \boldsymbol{\mu}\,\dot{\mathbf{\Phi}}\right) \tag{8.125}$$

where $\boldsymbol{\lambda}$ are the true multipliers. Since the values of $\boldsymbol{\lambda}^{*}$ are unknown *a priori*, an iterative procedure is necessary to solve equation (8.125). In each iteration a new value of $\boldsymbol{\lambda}^{*}$ is computed as follows:

$$\boldsymbol{\lambda}_{i+1}^{*} = \boldsymbol{\lambda}_{i}^{*} + \boldsymbol{\alpha}\left(\mathbf{\Phi}_{i} + \boldsymbol{\mu}\,\dot{\mathbf{\Phi}}_{i}\right) \tag{8.126}$$

with $\boldsymbol{\lambda}_{0}^{*}=0$ for the first iteration. In the limit $\boldsymbol{\lambda}_{0}^{*}=\boldsymbol{\lambda}$, however, enough accuracy is obtained in one or two iterations. Equation (8.126) physically represents the introduction at step *i+1* of forces that tend to compensate the fact that the constraints are not exactly zero. A consequence of this iteration is that small deviations in the constraints resulting from either the integration process or small penalty factors will be eliminated by the Lagrange multiplier terms of (8.124). The additional computational effort that it requires is not significant when compared to that necessary to solve the system of nonlinear differential equations.

8.5.2 Integration Procedure

For real time integration, it is necessary to use a fixed integration formula that will yield the same computational time in each integration step which has to be smaller than the step size. This condition imposes severe limitations to integration methods. They must be computationally inexpensive with few function evaluations and iterations in each step and must allow large step sizes without introducing excessive loss of accuracy and –most importantly– without becom-

ing unstable. Because of the ease of implementation, the trapezoidal rule has been chosen for the integration of the equations of motion in real time. This method is implicit, *A*-stable, and second order. It was shown in the numerical example of Chapter 7 how this rule performs, when it is directly combined with the equations of motion taking the positions as primary variables. It will be shown here how this integration scheme fits quite well into the framework provided by the fully Cartesian coordinates and the penalty formulation. Other recently proposed methods based on the midpoint rule that preserve energy and angular momentum (Simo and Wong (1991)) could also be used within this context.

The trapezoidal rule can be written as:

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \frac{h}{2}\left(\dot{\mathbf{q}}_n + \dot{\mathbf{q}}_{n+1}\right) \\ \dot{\mathbf{q}}_{n+1} &= \dot{\mathbf{q}}_n + \frac{h}{2}\left(\ddot{\mathbf{q}}_n + \ddot{\mathbf{q}}_{n+1}\right) \end{aligned} \tag{8.127}$$

where h is the time step. These finite difference expressions can be rewritten, considering $\mathbf{q}_{n+1}$ as the primary variable and solving for the velocities and accelerations at step (n+1). Consequently,

$$\begin{aligned} \dot{\mathbf{q}}_{n+1} &= \frac{2}{h}\mathbf{q}_{n+1} - \hat{\dot{\mathbf{q}}}_{n+1} \\ \ddot{\mathbf{q}}_{n+1} &= \frac{4}{h^2}\mathbf{q}_{n+1} - \hat{\ddot{\mathbf{q}}}_{n+1} \end{aligned} \tag{8.128}$$

Vectors $\hat{\dot{\mathbf{q}}}_{n+1}$ and $\hat{\ddot{\mathbf{q}}}_{n+1}$ depend on the positions, velocities, and displacements at step n and can be written as:

$$\begin{aligned} \hat{\dot{\mathbf{q}}}_{n+1} &= \dot{\mathbf{q}}_n + \frac{2}{h}\mathbf{q}_n \\ \hat{\ddot{\mathbf{q}}}_{n+1} &= \ddot{\mathbf{q}}_n + \frac{4}{h}\dot{\mathbf{q}}_n + \frac{4}{h^2}\mathbf{q}_n \end{aligned} \tag{8.129}$$

The setting up of the difference equations as done in (8.128) adds numerical efficiency to the computer implementation of the algorithm. Knowing that ($\dot{\boldsymbol{\Phi}} = \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} + \boldsymbol{\Phi}_t$), the substitution of (8.128) into the equations of motion (8.124) yields the following expression:

$$\begin{aligned} \frac{4}{h^2}\mathbf{M}\,\mathbf{q}_{n+1} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\alpha}\left(\boldsymbol{\Phi} + \boldsymbol{\Phi}_{\mathbf{q}}\left(\frac{2}{h}\boldsymbol{\mu}\,\mathbf{q}_{n+1} - \boldsymbol{\mu}\,\hat{\dot{\mathbf{q}}}_{n+1}\right) + \boldsymbol{\mu}\,\boldsymbol{\Phi}_t\right) = \\ = \mathbf{Q} + \mathbf{M}\,\hat{\ddot{\mathbf{q}}}_{n+1} \end{aligned} \tag{8.130}$$

which constitutes a set of nonlinear algebraic equations with $\mathbf{q}_{n+1}$ as the only unknowns. The use of Newton-Raphson iteration leads to the following iteration process:

$$\Delta \mathbf{q}_{i+1} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{q}}\right)_i^{-1} \mathbf{f}(\mathbf{q}_i) \tag{8.131}$$

where *i* represents the iteration number. The function **f** is defined as

$$\mathbf{f} = \mathbf{M}\,\ddot{\mathbf{q}}_{n+1} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\alpha}\left(\boldsymbol{\Phi} + \boldsymbol{\mu}\,\dot{\boldsymbol{\Phi}}\right) - \mathbf{Q} \tag{8.132}$$

and the tangent matrix as

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}} = \frac{4}{h^2}\,\mathbf{M} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\beta}\,\boldsymbol{\Phi}_{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{qq}}^{\mathrm{T}}\,\boldsymbol{\alpha}\left(\boldsymbol{\Phi} + \boldsymbol{\mu}\,\dot{\boldsymbol{\Phi}}\right) + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\alpha}\,\boldsymbol{\mu}\,\boldsymbol{\Phi}_{\mathbf{qq}}\dot{\mathbf{q}} - \mathbf{Q}_{\mathbf{q}} \tag{8.133}$$

where $\boldsymbol{\beta}=\boldsymbol{\alpha}\,(1 + 2\mu/h)$. $\mathbf{Q}_{\mathbf{q}}$ is the contribution to the tangent matrix coming from the nonlinear generalized forces (such as springs). A close examination at the tangent matrix reveals that the second term of the RHS of equation (8.133) is always much larger than the third and the fourth. This is so: first, by virtue of the fully Cartesian coordinates, $\boldsymbol{\Phi}_{\mathbf{qq}}$ is a very sparse third order tensor in which the only non-zero terms are constants of value 2 (these correspond to the exponents of the quadratic terms); and secondly, the values of $\boldsymbol{\Phi}$ and $\mu\,\dot{\boldsymbol{\Phi}}$ are always negligible compared with the values of $\boldsymbol{\Phi}_{\mathbf{q}}$. Furthermore, the parameter μ, which is an order of magnitude smaller than h, will make the fourth term negligible. This will be shown in the next section. Neglecting these terms may no longer assure the quadratic convergence of the Newton-Raphson iteration. However, accuracy and quasi-second order behavior are still satisfied. As a consequence, a quasi-tangent matrix can be defined as

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}} \cong \frac{4}{h^2}\,\mathbf{M} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\beta}\,\boldsymbol{\Phi}_{\mathbf{q}} - \mathbf{Q}_{\mathbf{q}} \tag{8.134}$$

where the diagonal matrix $\boldsymbol{\beta}$ becomes a constant times the identity matrix, when all the constraints are assigned the same penalty values.

8.5.3 Numerical Considerations

Improving convergence. The iteration process defined by (8.131) is carried out until $\|\Delta\mathbf{q}\|$ is smaller than a prescribed tolerance. For real time simulation, the iteration is stopped after a fixed number of iterations, that is one or at most two per time step. Convergence can be accelerated at each time step if the iteration is started not from the solution at the previous time step, but from the solution given by a predictor. A good second order predictor is the *modified trapezoidal rule* or *Heun method* which gives the following explicit coordinate interpolation:

$$\tilde{\mathbf{q}}_{n+1} = \mathbf{q}_n + h\,\dot{\mathbf{q}}_n + \frac{h^2}{2}\,\ddot{\mathbf{q}}_n \tag{8.135}$$

Once the solution has been obtained at step *n*, the iteration process of (8.131) can be started at step *n+1* with $\tilde{\mathbf{q}}_{n+1}$ rather than $\mathbf{q}_n$. Since no function evaluation

Table 8.6. Percentage of the total time required by each algorithm phase.

	% of CPU time
Solutions of equations	35.0
Forming the tangent matrix	33.5
Forming the residual	14.6
Evaluating the Jacobian matrix and constraints	13.0
Updating velocities and accelerations	3.5
Predictor	0.4
Total	100.0

is required in equation (8.135), the use of this predictor adds an insignificant amount of computations. As will be shown in an example below, it accelerates the integration process substantially.

Computer implementation. Fully Cartesian coordinates yield a constant mass matrix. This obviously represents a substantial savings in numerical computations at the time of forming the quasi-tangent matrix of equation (8.134). The major burden in computing this matrix lies in the product $\mathbf{\Phi}_\mathbf{q}^T \boldsymbol{\beta} \mathbf{\Phi}_\mathbf{q}$. Note that $\mathbf{\Phi}_\mathbf{q}$ is a sparse matrix and that due to the type of coordinates chosen the only nonzero terms are linear. The formation of this linear Jacobian matrix is rather inexpensive computationally. Furthermore, the product of the Jacobian matrices can be optimized by the use of sparse matrix operations. Note that the coordinates can be numbered so that a minimum profile of the final matrix is obtained.

This way of introducing the constraints through a penalty method leads to a tangent matrix that is dominated by the terms in the main diagonal. Consequently, the triangularization process does not require pivoting. Although the number of arithmetic operations is problem dependent, Table 8.6 gives an indication of the percentage time that each of the different phases of the solution process takes as a fraction of the total time. This Table corresponds to an average of many simulations performed by the authors. The most time consuming parts correspond to the formation of the tangent matrix and the solution of the resulting equations. In all the cases studied the use of the *Heun method* as a predictor eliminates the need for an extra iteration. Yet it only takes 0.4% of the time taken to complete one iteration. For real time applications, the time step size should be chosen so that at most two iterations are performed per time step.

Choosing the velocity constraint factor. The characteristic equation corresponding to the constraint condition $\mu\dot{\Phi}+\Phi=0$ is $\mu A+A=0$. and its root is $\lambda=-1/\mu$. The region of absolute stability for the trapezoidal rule is the negative half-plane. The product $h\lambda$ must lie in the negative real axis. Thus, $h/m>0$. At this stage, there are a series of possibilities. If both constraints Φ and $\dot{\Phi}$ are wanted to be

satisfied within the same accuracy, then both should be penalized with the same factor. Therefore μ should be equal to one. This value, however, tends to introduce damping in the system's response, producing a gradual decrease in the energy of the system. If the intention is to just eliminate the possible instabilities during the integration process, then h/μ can be chosen away from the area of the stability region where the response of the multi-body system lies. In the examples shown below, values of h/μ were chosen between 30 and 80 to eliminate the high frequency response in the acceleration. This yielded excellent results. Very small values of the parameter μ may not provide sufficient damping to eliminate the numerical instability. On the other hand, large values of μ will eliminate the instability but may introduce artificial damping in the response of the multibody system.

The method explained above is systematic and general, and shows very good convergence characteristics, even for large time steps. A numerical simulation is shown next which demonstrates its capabilities.

Example 8.6

Five-link open-chain multibody system. The multibody system shown in Figure 8.21 is composed of five links interconnected by revolute joints and has a total of 30 coordinates, 25 constraint conditions, and five degrees of freedom. Each link has a unit mass which is equally lumped at the joints to yield a stronger motion. The motion is due to its self-weight. The simulation is carried out using the algorithms described above for 20 seconds with a time step h=0.008 seconds.

The multibody system is analyzed with $\mu=h/80$ and $\alpha=10^7$. The response is not sensitive to the value of μ. Similar responses are obtained with $h/40 < \mu < h/120$ with slightly more damping for larger values of μ. The time history of the vertical tip acceleration is depicted in Figure 8.24. This time history gives a clear indication that the multibody system undergoes a very strong motion with peak accelerations of the order of 180 m/sec^2. This again illustrates the excellent convergence characteristics of the algorithm. In fact, the analysis was stopped after 10 minutes of simulation without any appearance of convergence problems. Each time step requires two iterations with a tolerance in the positions of $5*10^{-7}$, and each iteration takes 6.5 milliseconds of CPU time of a DECstation 3100.

The time history of the energy is shown in Figure 8.25. This time history shows how the energy is well preserved for this strong motion, even with a relatively large time step (the maximum error is 1.3%). Figure 8.26 illustrates the time histories of the constraint errors for a penalty factor of: 10^7 (solid line), 10^6 (short dashed curve), and 10^6 with 1 iteration of the augmented Lagrangian formulation. It can be seen how the maximum constraint error using the penalty value of 10^7 is 10^{-5}. With the penalty value of 10^6 the maximum constraint error is about 10^{-4}, but the use of one iteration of the augmented Lagrangian method brings it down an order of magnitude.

Figure 8.24. Vertical tip acceleration of the open-chain multibody system.

Figure 8.25. Total energy time history of the open-chain multibody system.

Figure 8.26. Maximum errors in the constraints using a penalty value of 10^7 (solid line), 10^6 (short dashed line), and 10^6 with augmented Lagrangian formulation (long dashed line).

Figure 8.27. Vertical acceleration at the middle link of the Bricard mechanism.

Figure 8.28. Total energy time history of the Bricard mechanism.

Example 8.7

Five-link closed-chain multibody system. The multibody system shown in Figure 8.21 is transformed into the closed-chain Bricard mechanism shown in Figure 8.22. The simulation is carried out for 40 seconds with a time step of 0.03 seconds, a penalty value of 10^7, and $\mu=h/60$. Figure 8.27 illustrates the acceleration time history of the middle link which shows that this multibody system undergoes a smoother motion than the previous one. Each time step requires two iterations with a tolerance in the positions of $5*10^{-7}$. Each iteration takes 5.0 milliseconds of CPU time. The total CPU time required to simulate 40 seconds simulation is about 13 seconds. The time history of the energy is depicted in Figure 8.28 which again shows how well the energy is preserved. The maximum error is 0.03%.

References

Argyris, J., "An Excursion into Large Rotations", *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, pp. 85-155, (1982).

Armstrong, W.W., “Recursive Solution to the Equations of Motion of an N-Link Manipulator”, Proc. 5th World Congress on Theory of Machines and Mechanisms, Vol. 2, pp. 1343-1346, Montreal, (1979).

Avello, A., Jiménez, J.M., Bayo, E., and García de Jalón, J., "A Simple and Highly Parallelizable Method for Real-Time Dynamic Simulation Based on Velocity Transformations", to appear in *Computer Methods in Applied Mechanics and Engineering*, (1993).

Bae, D.-S. and Haug, E.J., "A Recursive Formulation for Constrained Mechanical System Dynamics. Part I: Open-Loop Systems", *Mechanics of Structures and Machines*, Vol. 15, pp. 359-382, (1987).

Bae, D.-S. and Haug, E.J., "A Recursive Formulation for Constrained Mechanical System Dynamics. Part II: Closed-Loop Systems", *Mechanics of Structures and Machines*, Vol. 15, pp. 481-506, (1987-88).

Bae, D.-S., Hwang, R.S., and Haug, E.J., "A Recursive Formulation for Real-Time Dynamic Formulation", *1988 Advances in Design Automation*, ed. by S.S. Rao, ASME, pp. 499-508, (1988).

Bae, D.-S. and Won, Y.S., "A Hamiltonian Equation of Motion for Real Time Vehicle Simulation", *1990 Advances in Design Automation*, ed. by B. Ravani, ASME, pp. 151-157, (1990).

Bayo, E., García de Jalón, J., Avello, A., and Cuadrado, J., "An Efficient Computational Method for Real Time Multibody Dynamic Simulation in Fully Cartesian Coordinates", *Computer Methods in Applied Mechanics and Engineering*, Vol. 92, pp. 377-395, (1991).

Featherstone, R., "The Calculation of Robot Dynamics Using Articulated Body Inertias", *The Int. Journal of Robotic Research*, Vol. 2, pp. 13-30, (1983).

Featherstone, R., *"Robot Dynamics Algorithms"*, Kluwer, (1987).

García de Jalón, J., Jiménez, J.M., Avello, A., Martín, F., and Cuadrado, J., "Real Time Simulation of Complex 3-D Multibody Systems with Realistic Graphics", *Advanced Research Workshop on Real Time Integration Methods for Mechanical System Simulation,* NATO, *Snowbird*, UTAH, August, (1989).

Haug, E.J., *Computer-Aided Kinematics and Dynamics of Mechanical Systems. Volume I: Basic Methods*, Allyn and Bacon, (1989).

Jerkovsky, W., "The Structure of Multibody Dynamic Equations", *Journal of Guidance and Control*, Vol. 1, pp. 173-182, (1978).

Jiménez, J.M., "Formulaciones Cinemáticas y Dinámicas para la Simulación en Tiempo Real de Sistemas de Sólidos Rígidos", Ph.D. Thesis, University of Navarre, San Sebastián, (1993).

Kim, S.S. and Vanderploeg, M.J., "A General and Efficient Method for Dynamic Analysis of Mechanical Systems Using Velocity Transformations", *ASME Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 108, 176-182, (1986).

Luh, J.Y.S., Walker, M.W., and Paul, R.P.C., "On Line Computational Scheme for Mechanical Manipulators", *Journal of Dynamic Systems, Measurements, and Control*, ASME, Vol. 102, pp. 69-76, (1980).

Nikravesh, P.E., *Computer-Aided Analysis of Mechanical Systems*, Prentice-Hall, (1988).

Rodriguez, G., Jain, A., and Kreutz, K., "A Spatial Operator Algebra for Manipulator Modeling and Control", *International Journal of Robotics Research*, Vol. 10, pp. 371-381, (1991).

Simo, J.C. and Wong, S., "Unconditionally Stable Algorithms for Rigid Body Dynamics that Exactly Preserve Energy and Momentum", *International Journal for Numerical Methods in Engineering*, Vol. 31, pp. 19-52, (1991).

Walker, M.W. and Orin, D.E., “Efficient Dynamic Computer Simulation of Robotic Mechanisms”, Journal of Dynamic Systems, Measurements, and Control, ASME, Vol. 104, pp. 205-211, (1982).