Modeling Multibody Systems with ROBOTRAN

www.robotran.be

July 3, 2024

Contents

1 Introduction

The generation of the kinematic and dynamic equations of multibody systems (MBS) like mechanisms, vehicles, robots, human body, satellites, etc. (Fig. 1) can be quite tedious both because of the size and the complexity of the underlying mathematical models.

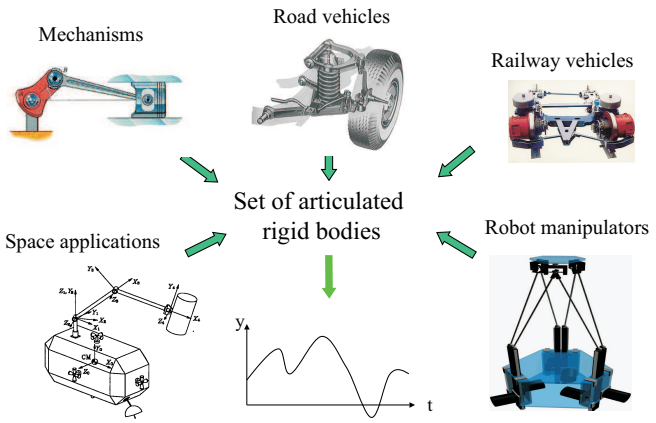


Figure 1: Multibody applications.

Among the possible options to generate the multibody equations (kinematic, dynamic), the *symbolic* approach, which appeared in the eighties, is a powerful tool to *drastically simplify* mathematical expressions and to confer the equations *a high portability* towards other scientific disciplines like control, optimization, dimensioning, etc. For that purpose, the ROBOTRAN environment was developed at UCL-MEED, according to Fig. 2. Starting from a graphical description of the MBS (with the MBsysPAD graphical editor), the MBS equations can be generated – in a few milliseconds – by the symbolic translator MBsysTran[1] in MATLAB , PYTHON or C syntax; these symbolic equations are then automatically interfaced with the simulator (MBsysLab and MbsysC). 3D animation of the virtual system can then be performed via the MBsysPad editor.

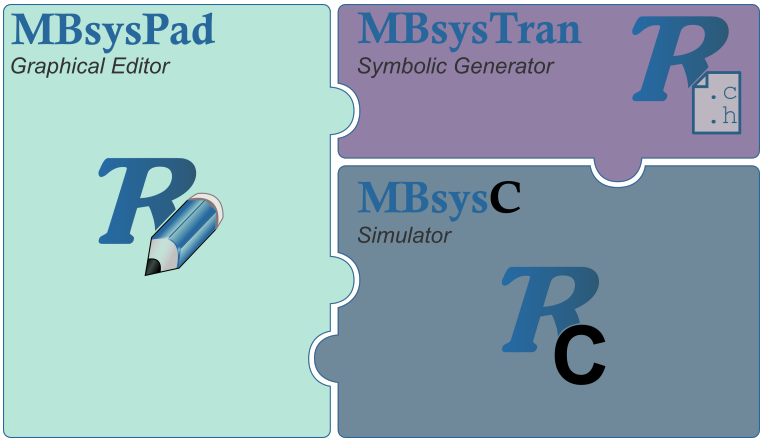


Figure 2: The ROBOTRAN program (e.g., c-code implementation).

[1]This symbolic generator was historically named ROBOTRAN : this acronym ROBOTRAN presently refers to the full modeling environment.

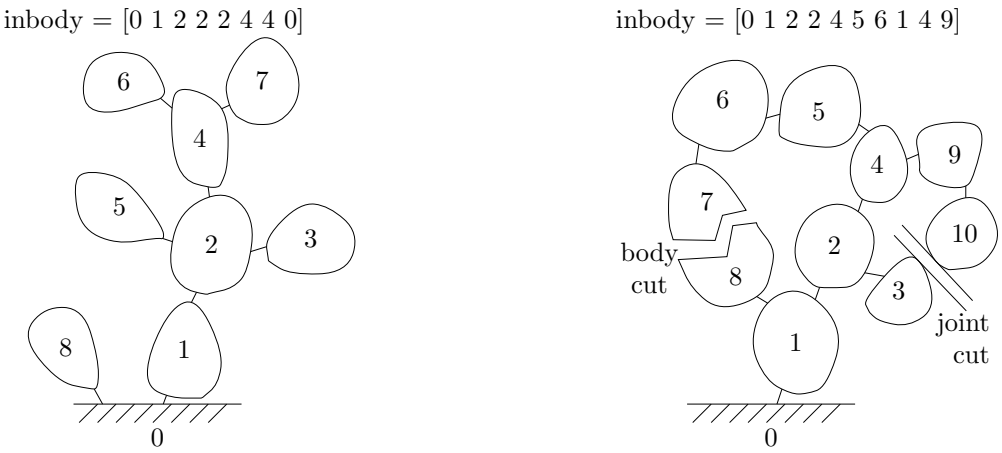
About this document The purpose of this text is really to invite the ROBOTRAN user to "read before acting". Indeed, roughly speaking, in terms of "skills", a multibody program is not like a text-processor, a minimum of knowledge about the underlying conventions, terminology and theory is indispensable before starting to draw the simplest multibody system, e.g., the one-body pendulum! In addition to this document, a more in-depth theoretical presentation can be found in the book [1] which underlies the ROBOTRAN program or in publication [2].

In Section 2, the conventions and notations used in ROBOTRAN are introduced. Then in Section 3, the dynamic formalisms for unconstrained and constrained MBS are presented while Section 4 focuses on some specific modeling features. Finally, Section 5 describes the different type of analysis that can be carried out on a MBS.

2 Definitions, Conventions and Notations

2.1 Topology

A general *multibody system* is a mechanical system composed of n rigid bodies connected by joints.
A tree-like structure (Figure 3_a), in which there are no loops of bodies, is always used to describe the *topology* of the system, i.e., the way the bodies are connected. Serial robots or cranes are examples of tree-like multibody systems. Each time a closed-loop of bodies is present in the system, a particular procedure will be used to *restore* a temporary tree-like structure by either cutting a body into two parts or by cutting a joint (Figure 3_b). Parallel robots, vehicle multi-link suspensions, four-bar and slider-crank mechanisms are examples of closed-loop multibody systems.
The topological vector *inbody* is defined as the vector whose i^{th} element contains the index of the parent of body i (see the example of figure 3_a).



(a) Tree-like structure

(b) Closed-loop structure

Figure 3: System topology.

2.2 Frames, vectors and tensors

As shown in Figure 4, an orthogonal right-handed body-frame $\{\hat{\mathbf{X}}^i\}$ is rigidly attached to each body i and located at its center of mass CM^i. It is generally simpler to choose this frame in such a way that its axes correspond to the principal inertia axes of the body, but this is not indispensable. Also shown in Figure 4 is $\{\hat{\mathbf{I}}\}$, the inertial frame attached to the base body 0. Before listing the necessary model data, let us introduce the following notations to define vectors and tensors.

A vector $\mathbf{a}$ is described in a given body frame $\{\hat{\mathbf{X}}^i\}$ by its three components a_x, a_y, a_z:

$$\mathbf{a} = a_x\,\hat{\mathbf{X}}^i_x + a_y\,\hat{\mathbf{X}}^i_y + a_z\,\hat{\mathbf{X}}^i_z \tag{1}$$

Writing the frame $\{\hat{\mathbf{X}}^i\}$ under the form of a 3×1 column matrix of unit *vectors*:

$$[\hat{\mathbf{X}}^i] \triangleq \begin{pmatrix} \hat{\mathbf{X}}^i_x \\ \hat{\mathbf{X}}^i_y \\ \hat{\mathbf{X}}^i_z \end{pmatrix} \tag{2}$$

and applying classical matrix multiplication rules, we can write any vector $\mathbf{a}$ in the following concise form:

$$\mathbf{a} = [\hat{\mathbf{X}}^i]^t \; a \text{ where } a \triangleq (a_x \;\; a_y \;\; a_z)^t \tag{3}$$

Where the superscript ^{t} denotes the transpose of the matrix. The same notation holds for a tensor $\mathbf{T}$ of order 2 (ex: the inertia tensor of a body with respect to its center of mass):

$$\mathbf{T} = [\hat{\mathbf{X}}^i]^t \, T \, [\hat{\mathbf{X}}^i] \text{ with } T = \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{pmatrix} \tag{4}$$

2.3 Bodies characterization

In Fig. 4, a body i is represented as well as its – unique – parent body h and its "children" bodies j and k. By convention, a joint which precedes a body in the structure has the same index as this body (see joints i, j and k in Fig. 4). For any joint i, two reference anchor points are defined: the first one, O^i, on the parent body h and the second one, O'^i, on the body i.

The following quantities are introduced for describing each body i ($i = 1 : n$):

- $\mathbf{d}^j = [\hat{\mathbf{X}}^i]^t \, d^j$, the *joint vector* which locates on body i the connection point O^j of joint j with respect to the connection point O'^i of joint i. Body i being rigid, the components of $\mathbf{d}^j$ are constant in the $\{\hat{\mathbf{X}}^i\}$ frame. Such a vector is defined for each of the children $j, k, ...$ of body i (see Figure 4);
- $\mathbf{l}^i = [\hat{\mathbf{X}}^i]^t \, l^i$, the position vector of the center of mass CM^i of body i with respect to the connection point O'^i of joint i. The components of $\mathbf{l}^i$ are constant in the $\{\hat{\mathbf{X}}^i\}$ frame;
- m^i, the mass of body i;
- $\mathbf{I}^i = [\hat{\mathbf{X}}^i]^t \, I^i \, [\hat{\mathbf{X}}^i]$, the symmetric inertia tensor of body i with respect to its center of mass CM^i. Its components are constant in the $\{\hat{\mathbf{X}}^i\}$ frame.

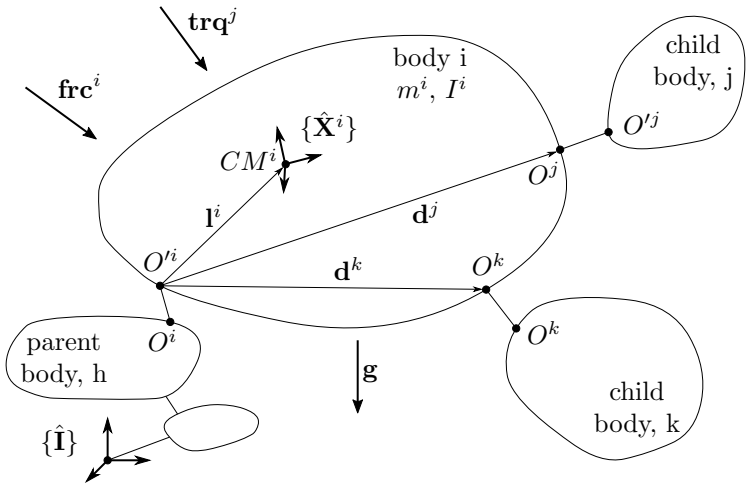


Figure 4: Body characterization.

In ROBOTRAN , the center of mass and the different anchor points of the body are defined with respect to the joint anchor point (O'^i in Figure 4) and in the same basis as the $\{\hat{\mathbf{X}}^i\}$ frame.

2.4 Joints characterization

Joints between bodies represent devices such as telescopic arms, hinges, universal joints, ball joints, etc. as shown in the examples of Fig. 5.

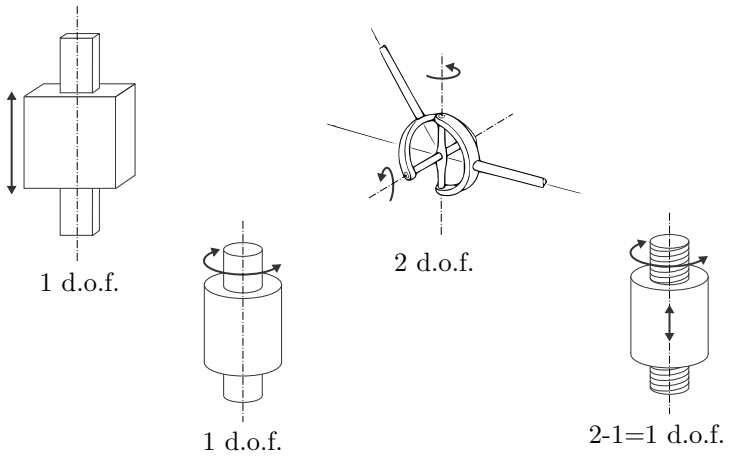


Figure 5: Example of joints.

Relative motions which occur in these joints are described in ROBOTRAN by *relative* generalized coordinates q and their time derivatives $\dot{q}$ and $\ddot{q}$. The absolute configuration of the MBS is computed in terms of these relative coordinates and time derivatives.

2.4.1 One degree of freedom joints

In order to avoid the description of a large database of joints, it is assumed in ROBOTRAN that joints have only one degree of freedom, either *prismatic* (denoted "T") or *revolute* (denoted "R") as depicted in Figure 6. Each joint j is assumed to be massless and connects body j to its parent i at anchor reference points, O'^j, O^j respectively, as already mentioned. These points[2] are *distinct* for a prismatic joint but has been arbitrarily chosen *identical* for a revolute joint (Figure 6).

Let us introduce the joint unit vector $\hat{\mathbf{e}}^j$ aligned with the joint axis according to Figure 6.
In joint j, the generalized coordinate q^j represents:

- a relative displacement if j is prismatic (units = $[m]$) such that $\overrightarrow{O^j\,O'^j} = q^j\,\hat{\mathbf{e}}^j$ (q^j is positive in Fig. 6_a).
- a relative angle if j is revolute (units = $[rad]$), which characterizes the rotation along the unit vector $\hat{\mathbf{e}}^j$ of body frame $\{\hat{\mathbf{X}}^j\}$ with respect to parent body frame $\{\hat{\mathbf{X}}^i\}$. The sign of the angle is dictated by the corkscrew rule applied to body j with respect to body i along $\hat{\mathbf{e}}^j$.

(a) Prismatic joint (b) Revolute joint

Figure 6: Elementary joints.

[2]which can be arbitrarily chosen on the joint axis for dynamic calculations purposes.

More elaborate joints (spherical, universal, or even a general 6 d.o.f. joint) can be straightforwardly modeled as a succession of these 2 elementary joints and so-called *fictitious* bodies as shown in the example of Fig. 7 which refers to the 2 d.o.f. universal joint of Fig. 5. Fictitious bodies have no dimension nor mass but they are numbered and they possess a body-fixed frame, exactly as for the real bodies. For instance, in order to confer 6 d.o.f. to a vehicle chassis, six joints – three prismatic and three revolute successively – will be inserted between the base (body 0) and the chassis (body 6), with five intermediate fictitious bodies (1...5)[3].

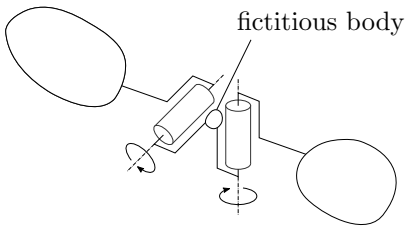


Figure 7: Modeling of a 2 d.o.f. universal joint.

2.4.2 Joint modeling hypothesis

To avoid more than one frame per body, the following convention is adopted in ROBOTRAN , related to the joint direction:

- The axis of any (prismatic or revolute) joint i must be aligned with one of the axis directions (x, y or z) of the frame attached to body i and to its parent body h ($= inbody(i)$) (Figure 9). The unit joint vector $\hat{\mathbf{e}}^i$ is such that, at any time:

$$\hat{\mathbf{e}}^i = \hat{\mathbf{X}}^i_x = \hat{\mathbf{X}}^h_x \text{ or } \hat{\mathbf{e}}^i = \hat{\mathbf{X}}^i_y = \hat{\mathbf{X}}^h_y \text{ or } \hat{\mathbf{e}}^i = \hat{\mathbf{X}}^i_z = \hat{\mathbf{X}}^h_z \tag{5}$$

  for a joint aligned with the x, y, z direction respectively.

2.4.3 Skewed joints

Although the previous hypothesis related to the joint directions could seem restrictive, this is not really the case because if a joint is "skewed" in the real system[4], it can nevertheless be simply modeled by introducing intermediate artificial joints (and fictitious massless bodies) which will be locked to a constant value during the simulation, as illustrated in Fig. 8. In this example, the axis of a revolute joint i (q^i) in the real system (Fig. 8_a) forms a constant angle α with respect to the $\{x, z\}$ plane of the parent body frame $\{\hat{\mathbf{X}}^h\}$ along the $\hat{\mathbf{X}}^h_x$ unit vector. Inserting a fictitious body i and a revolute joint ($q^i = \alpha$) solves the problem (Fig. 8_b). Indeed:

- the ROBOTRAN reference configuration (Figure 9 of section 2.5) is obtained for $q^i = 0$,
- the presence of this constant angle α in the real system will be taken into account in the numerical analysis by considering this joint as being driven (see Section 4.1 for more details) and then locked to a constant value, i.e.: $q^i = \alpha$, $\dot{q}^i = \ddot{q}^i = 0 \ \forall t$. During the numerical analysis, the corresponding equation of motion will simply be disregarded.

[3] these fictitious bodies do *not* need to be added in the graphical editor and are handled internally by the generator.
[4] Ex.: the handlebars steering of a bicycle

(a) real system (b) model

Figure 8: Modeling of a skewed joint.

2.5 Multibody reference configuration

We can now define a very important notion for modeling MBS with ROBOTRAN : the *reference configuration* of the tree-like[5] system defined as follows (Figure 9):

- All[6] body-fixed frames $\{\hat{\mathbf{X}}^i\}$ are aligned with the inertial frame $\{\hat{\mathbf{I}}\}$
- All the generalized coordinates are equal to zero: $q^i = 0\,, \forall$ joint i, revolute *and* prismatic.

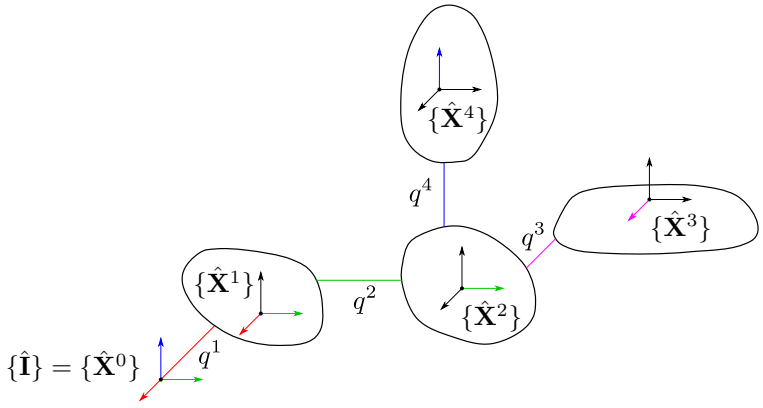


Figure 9: Multibody system - reference configuration with all frames aligned ($q^i = 0\ \forall i$).

ROBOTRAN : Bodies and joints (topology, type, shapes) are graphically sketched by the user in the MBsysPad editor *in the MBS reference configuration*. Numerical data are also introduced there (masses, inertia, vector components, etc.). Closed-loop systems must be converted into tree-like structures by suitable loop cuts, also introduced in the MBsysPad editor.

[5] always required and thus restored for closed-loop systems
[6] being fictitious or not!

2.6 Forces and Torques

As regards forces and torques acting on body i, we split them into three categories:

1. The *gravitational* force is considered separately in the formalism. This force is computed from the gravity vector $\mathbf{g}$ whose three components g are provided in the inertial frame $\{\hat{\mathbf{I}}\}$ for obvious reasons: $\mathbf{g} = [\hat{\mathbf{I}}]^t\, g$ (Figure 4).

2. The *joint* forces/torques, which act on a body via its parent and children joint connections. Their contribution to the equations of motion – denoted Q^i for joint i – will be detailed in Section 4.2.

3. The *external* forces/torques: besides joint forces and gravity, all other forces and torques are considered as *external* to the MBS and gathered together, for each body i, into a resultant force vector $\mathbf{frc}^i$ and a resultant torque vector $\mathbf{trq}^i$ with respect to the body center of mass CM^i. Section 4.3 will present them in more details.

In addition to these three categories and for practical reasons, we have introduced the so-called "link forces", which are very useful in practice; A link force represents a force that acts between two *materials points* of the MBS. Being not associated with any joint, the link force is considered by ROBOTRAN as two "action-reaction" external forces (i.e., belonging to the third category), while being internal to the system. Section 4.4 will present them in more details.

3 Multibody Formalisms

3.1 Dynamics of Tree-like Multibody Systems

In this section, formalisms for tree-like unconstrained MBS are proposed. Since constrained systems (i.e., containing loops of bodies) are modeled in ROBOTRAN by first restoring a tree-like structure as mentioned in Section 2.1, those formalisms are necessary for any kind of MBS.

Various multibody formalisms can be used to derive the equations of motion for a multibody system (ex. based on a virtual principle, the Lagrange equations or the standard Newton/Euler laws formulated recursively, ...). The equations describing the dynamics can be generated in the following *semi-explicit* form in terms of the generalized accelerations. The semi-explicit formulation is also referred as the direct dynamics model in the literature.

$$M(q,\delta)\ddot{q} + c(q,\dot{q},\delta,frc,trq,g) = Q(q,\dot{q}) \tag{6}$$

In the previous equations:

- M $[n*n]$ is the symmetric generalized mass matrix of the system,
- $c\,[n*1]$ is the nonlinear dynamic vector which contains the gyroscopic, centrifugal and gravity terms as well as the contribution of external resultant forces, frc and torques, trq,
- q $[n*1]$ denotes the relative generalized coordinates,
- δ $[10\,n*1]$ gathers together the dynamic parameters of the system (body masses m^i, centers of mass (the three components of vectors $\mathbf{l}^i$) and the inertia matrices with respect to the center of mass (the six components of tensors $\mathbf{I}^i$ in the body-fixed frame)),
- Q $[n*1]$ represents the generalized joint forces (torques).

Furthermore, the equations of motions can be expressed in a more compact form when expressions of the mass matrix and the accelerations are not explicitly required. This results in an expression of the generalized joint forces $Q(q,\dot{q})$ expressed as a function ϕ of the MBS kinematics (q,$\dot{q}$,$\ddot{q}$) and its dynamic parameters. This formulation of the equations is said to be *implicit* in terms of the generalized accelerations (this formulation is also referred as the inverse dynamics model in the literature).

$$Q(q,\dot{q}) = \phi(q,\dot{q},\ddot{q},\delta,frc,trq,g) \tag{7}$$

ROBOTRAN – Both the *semi-explicit* form (6) and the *implicit* form (7) are used for the MBS analyses (Section 5). Based on a MBS topological description (i.e., in MBSysPad), the equations are generated symbolically and stored in separated files (.c, .m or .py). The `dirdyna` file outputs both the mass matrix term M and the vector c as a function of the system configuration $(q,\dot{q},\ddot{q})$. The `invdyna` file outputs the right-hand side term ϕ as a function of the system configuration.

3.2 Dynamics of Constrained Multibody Systems

3.2.1 Dynamic and constraint equations

Most multibody applications contain loops of bodies (car suspension, parallel robots, mechanisms, etc., as shown in Fig. 10) which impose the generalized joint coordinates q to satisfy geometrical constraints at any time, denoted $h_l(q) = 0$. Beside these loop constraints, other algebraic constraints, denoted $h_a(q) = 0$, can arise for other reasons (Section 4.6).

Gathering these m constraints together, we can write:

$$h(q) = \begin{pmatrix} h_l(q) \\ h_a(q) \end{pmatrix} = 0 \quad [m * 1]$$

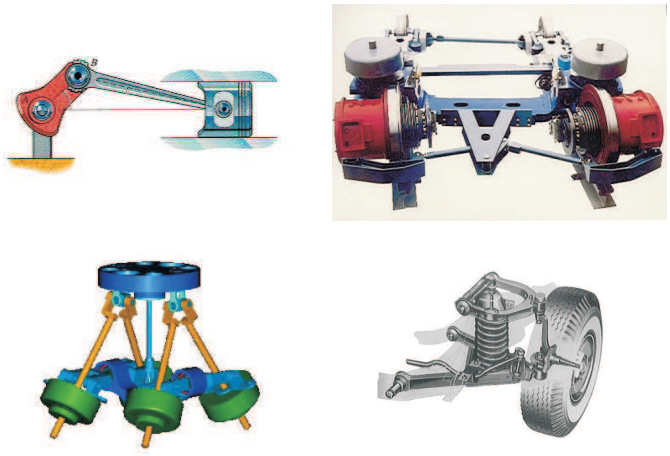


Figure 10: Example of closed-loop multibody systems

In order to fully describe the system, these constraints and their first and second time derivatives must be added to the equations of motion[7], in which constraint forces are introduced via the Lagrange multipliers technique. The *semi-explicit* form (6) transforms in the following differential algebraic equations (DAE) system (8-11):

$$M(q)\,\ddot{q} + c(q, \dot{q}, g) = Q(q, \dot{q}) + J^{\mathrm{T}}\lambda \tag{8}$$

$$h(q) = 0 \tag{9}$$

$$\dot{h}(q, \dot{q}) = J(q)\dot{q} = 0 \tag{10}$$

$$\ddot{h}(q, \dot{q}, \ddot{q}) = J(q)\ddot{q} + \dot{J}\dot{q}(q, \dot{q}) = 0 \tag{11}$$

where:

- $J = \frac{\partial h}{\partial q^t}$ denotes the constraint Jacobian matrix (dimension: $[m * n]$),
- $\dot{J}\dot{q}(q, \dot{q})$ is the quadratic term (involving the products $\dot{q}^i\,\dot{q}^j$) of the constraints at acceleration level (dimension: $[m * 1]$),
- λ represents the Lagrange multipliers associated with the constraints (dimension: $[m * 1]$).

[7] in which, for legibility reasons, we will no longer indicate the dependence with respect to the dynamic parameters, δ and the external forces, frc and torques, trq.

NB: at this level, only holonomic scleronomic[8] constraints (i.e., which can be written in the form of algebraic equations $h(q) = 0$) have been considered. Non-holonomic constraints (such as the tridimensional pure rolling motion between bodies), which can only be written at the velocity level $w(q, \dot{q}) = 0$, are not considered in this text. Nevertheless, they can formally be added — at velocity and acceleration levels (equations (10-11))— to the set of holonomic constraints, but they must be processed differently during the numerical resolution: in particular their time integration — if required — must be performed numerically since the algebraic form (9) does not exist for these non-holonomic constraints.

The compact *implicit* formulation (7) can also be written for a constrained MBS. In that case, the equations of motion are written as the DAE system (12-15).

$$\phi(q, \dot{q}, \ddot{q}) = Q(q, \dot{q}) + J^{\mathrm{T}}\lambda \tag{12}$$

$$h(q) = 0 \tag{13}$$

$$\dot{h}(q, \dot{q}) = J(q)\dot{q} = 0 \tag{14}$$

$$\ddot{h}(q, \dot{q}, \ddot{q}) = J(q)\ddot{q} + \dot{J}\dot{q}(q, \dot{q}) = 0 \tag{15}$$

3.2.2 Index reduction

Various methods can be used to solve the DAE systems formed by (8-11) or (12-15). Among these, one can opt for a full index reduction[9] of the system to a purely differential form, which can be obtained by means of the so-called *Coordinate Partitioning Method* [3].

The Jacobian matrix $J(q)$ is assumed to have full rank m. In this case, the constraints $h(q) = 0$ are independent and m generalized coordinates can be locally expressed as functions of the $(n - m)$ others (n denoting both the total number of coordinates and joints). In this way, it becomes possible to reduce the original DAE system (8 - 11) to a set of $(n - m)$ ordinary differential equations (ODE) in these $(n - m)$ independent coordinates. This reduced set represents the equations of motion of the constrained multibody system, where $(n-m)$ also corresponds to its number of degrees of freedom. Mathematically speaking, the reduction amounts to project the equations of motion (8) in the constraints space along the permissible motion directions.

The first step is to perform the **coordinate partitioning** by defining two partitions within the generalized coordinates: the independent and the dependent ones. After reordering the vector of generalized coordinates q (and the columns of the constraint Jacobian J accordingly), the following partition holds:

$$q = \begin{pmatrix} q_u \\ q_v \end{pmatrix} \; ; \; J = \begin{pmatrix} J_u & J_v \end{pmatrix} \tag{16}$$

Where q_u denotes the subset of $(n-m)$ *independent* coordinates and q_v denotes the subset of *dependent* coordinates. If the subset q_v is chosen correctly, the m by m matrix J_v will be regular (Section 5.1 for more details).

Once the coordinate partitioning is established, the second step is to perform the **index reduction** which simply relies on matrix permutations and operations to produce the reduced ODE system.

Starting from equation (8), let us first partition and rearrange the generalized mass matrix M and the vector c according to the coordinate partitioning (16):

$$\begin{pmatrix} M_{uu} & M_{uv} \\ M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + \begin{pmatrix} c_u \\ c_v \end{pmatrix} = \begin{pmatrix} Q_u \\ Q_v \end{pmatrix} + \begin{pmatrix} {J_u}^t \\ {J_v}^t \end{pmatrix} \lambda \tag{17}$$

[8] Rheonomic constraints have the form $h(q, t) = 0$ with an explicit dependence on time t. They are not considered here for conciseness: in Section 4.1 we will consider *driven variables* which enable to deal with rheonomic constraints without resorting to an explicit dependency of time t.

[9] from an index-3 DAE system to a index-0 DAE system (i.e., a ODE system)

Since the constraints are assumed independent, J_v is regular which allows for the elimination of the unknowns λ using the lower part of system (17):

$$\begin{pmatrix} M_{uu} & M_{uv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + {B_{vu}}^t \begin{pmatrix} M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + c_u + {B_{vu}}^t c_v = Q_u + B_{vu}^t Q_v \tag{18}$$

where we define the so-called coupling matrix: $B_{vu} \triangleq -(J_v)^{-1} J_u$. The algebraic constraints have now to be solved in order to eliminate the dependent variables q_v. While analytical solutions can exist for specific cases, the algebraic constraints (9) are generally nonlinear and require an iterative procedure to be solved. The Newton-Raphson algorithm can be used to express q_v for a given q_u through successive estimations of q_v:

$$q_v^{k+1} = q_v^k - (J_v)^{-1} h\,|_{q_v = q_v^k} \tag{19}$$

Where the right-hand side is evaluated for $q_v = q_v^k$ and the values of q_u correspond to the instantaneous configuration. Using the first (eq. (10)) and second derivatives (eq. (11)) of the constraints, the generalized velocities and accelerations $\dot{q}_v$ and $\ddot{q}_v$ are respectively expressed via classical linear algebra techniques.

$$\dot{q}_v = B_{vu}\,\dot{q}_u \tag{20}$$

$$\ddot{q}_v = B_{vu}\,\ddot{q}_u + b \quad \text{with} \quad b \triangleq -J_v^{-1}\,(\dot{J}\,\dot{q}) \tag{21}$$

And can also be eliminated from the differential equations (18). This produces the final *reduced* system:

$$\left(M_{uu} + M_{uv}B_{vu} + {B_{vu}}^t M_{vu} + {B_{vu}}^t M_{vv} B_{vu}\right) \ddot{q}_u + \left(M_{uv} + {B_{vu}}^t M_{vv}\right) b + (c_u + {B_{vu}}^t c_v) - (Q_u + B_{vu}^t Q_v) = 0$$

which can be concisely written as:

$$M_r(q_u)\ddot{q}_u + F_r(\dot{q}_u, q_u) = 0 \tag{22}$$

The set of ordinary differential equations (22) constitutes the equations of motion of the constrained system described in terms of the $n - m$ independent generalized coordinates q_u. The workflow to obtain the system (22) through the index reduction is summarized in Figure 11.

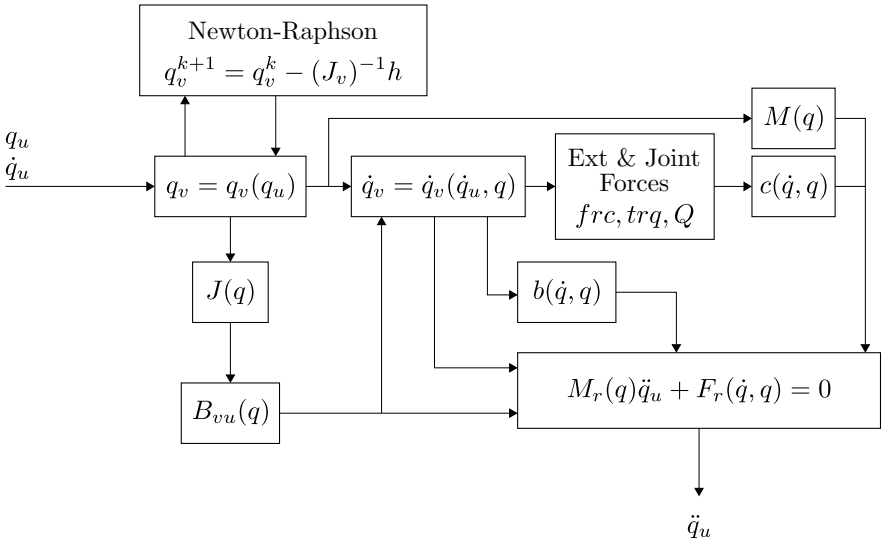


Figure 11: ODE model of constrained MBS.

The independent accelerations $\ddot{q}_u$ are easily expressed as a function of $(q_u, \dot{q}_u)$ by solving the system (22) for $\ddot{q}_u$. The solving and the reduction (17-21) coupled with the computation of M and c can be encapsulated in one optimized file `accelred` generated symbolically (available on demand). This file is both system- and partitioning-dependent.

Given the coordinate partitioning (q_u,q_v), the reduction can also be applied to the implicit form (12 - 15) :

$$\begin{pmatrix} \phi_u \\ \phi_v \end{pmatrix} = \begin{pmatrix} Q_u \\ Q_v \end{pmatrix} + \begin{pmatrix} {J_u}^t \\ {J_v}^t \end{pmatrix} \lambda \tag{23}$$

As for the semi-explicit case, from the second line of the above equation we can compute the Lagrange multipliers, the matrix J_v being regular, and substitute them in the first line. We obtain a reduced set of $n-m$ equations which can be used to express the joint forces in the independent joints, Q_u, as a function of the MBS dynamics (ϕ) and the dependent joint forces Q_v.

$$Q_u = \phi_u + B_{vu}^t(\phi_v - Q_v) \tag{24}$$

The right-hand side in equation (24) consists in the ϕ computation and the index reduction of the implicit equations. The computation of the right-hand side is encapsulated in one optimized file `invdynared` generated symbolically. This file is both system- and partitioning-dependent.

Although equations (22) are sufficient to analyze the MBS dynamics, some "interesting" variables have been eliminated in the reduction process: the dependent accelerations $\ddot{q}_v$ and the Lagrange multipliers λ. For a given system state ($\ddot{q}_u, \dot{q}_u, q_u$), they can be easily computed as follows:

- $\ddot{q}_v$ can be obtained directly from system (21).
- As regards the Lagrange multipliers λ, the lower part of equation (17) can be used to recover them:

$$\lambda = ({J_v}^t)^{-1} \{M_{vu}\ddot{q}_u + M_{vv}\ddot{q}_v + c_v - Q_v\} \tag{25}$$

The lagrange multipliers λ are automatically computed and put in a result file. See next sections for more information.

3.2.3 Generation of loop constraints

When dealing with closed loop MBS, a so-called *cutting* procedure is implemented to artificially restore a tree-like structure (recall Fig. 3_b). This procedure is at the root of the automatic generation by ROBOTRAN of the loop constraints $h_l(q) = 0$ and their Jacobian matrix J_l. Three cutting procedures are systematized in ROBOTRAN and presented here below.

Based on a given MBS topological description, the symbolic generator creates two files: `mbs_cons_hJ` where the loop constraints and their Jacobian matrix are computed and `mbs_cons_jdqd` where the term $\dot{J}_l\dot{q}$ in $\ddot{h}_l(q,\dot{q})$ is computed.

Cut of a body (type 1) This first procedure is the most general[10] one and consists in cutting a body of the loop into two parts denoted the *original* body o and the *shadow* body sh. Figure 12 illustrates this cut for which six constraints must be satisfied :

1. three position constraints ($h_1(q), h_2(q), h_3(q)$) imposing that bodies o and sh have a common point $\mathrm{P}_o \equiv \mathrm{P}_{sh}$ at any time,
2. three orientation constraints (denoted $h_4(q), h_5(q), h_6(q)$) expressing that frames $\{\hat{\mathbf{X}}^o\}$ and $\{\hat{\mathbf{X}}^{sh}\}$ must coincide at any time.

Although this first cutting procedure is general, it leads to six constraints and, most of the time, practical applications can avoid this first technique by resorting to one of the following two procedures.

This first type of cut is defined in the graphical editor MBsysPad. To characterize it, we need to define the components of the vector $\mathbf{d}_{lp}$ locating – in the shadow body sh, with respect to joint sh – the point P_{sh} (the anchor point of joint o in the case illustrated in Fig. 12). The components of $\mathbf{d}_{lp}$ are constant in the $\{\hat{\mathbf{X}}^{sh}\}$ frame.

[10] but paradoxically, seldom used because it is the most costly in terms of arithmetical operations.

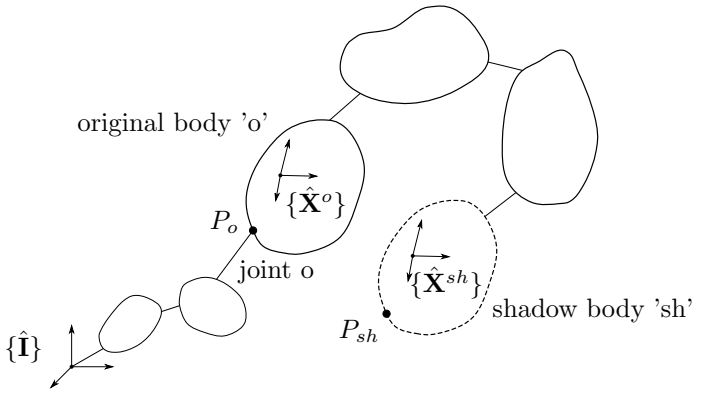


Figure 12: Cutting procedure of type 1: cut of a body.

Cut of a ball joint (type 2) This procedure applies to loops which contain a *ball joint* that can be considered as "ideal" from both the geometrical (no backlash) and dynamic (no torque transmitted) points of view. As shown in Fig. 13, the spherical joint under consideration is removed from the system[11]. Denoting the center point P of the spherical joint by P_p for the *primary* kinematic chain and by P_s for the *secondary* kinematic chain respectively, three position constraints suffice to close the loop: they simply express that the position vectors $\mathbf{x}(P_p)$ and $\mathbf{x}(P_s)$ coincide at any time:

$$\mathbf{x}(P_p) - \mathbf{x}(P_s) = 0 \quad , \quad \forall\, t$$

In case of a planar loop, the cut can be applied to any "ideal" hinge: only 2 constraints will be expressed in the plane of the loop.

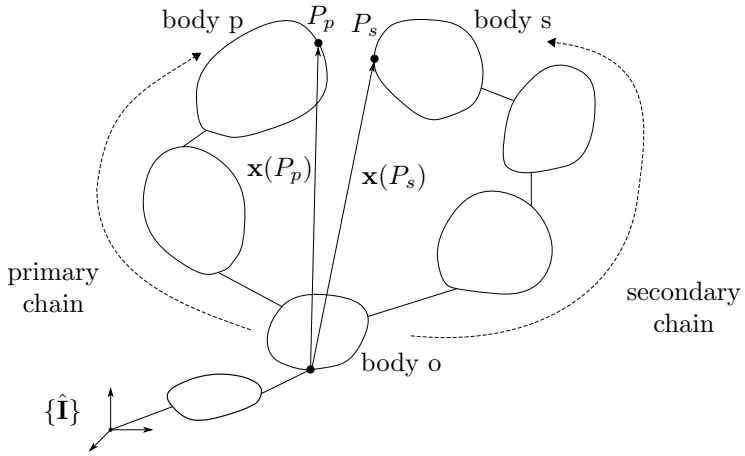


Figure 13: Cutting procedure of type 2: cut of a ball joint.

This second loop cut is introduced in the graphical editor MBsysPad. The *ball* cut is defined by connecting two anchor points. The constraints will be expressed in the frame associated with the youngest common ancestor body (body o in Fig. 13). The λ are the force components execed by the child (second body) to the parent (first body).

[11] no additional "shadow" body is required in this case.

Cut and removal of a connecting rod (type 3) This procedure applies to loops which contain a *connecting rod* whose mass and inertia can be neglected and whose (two) connecting joints can be considered as "ideal", as previously defined for the spherical joint.

As shown in Figure 14, the connecting rod under consideration is simply removed from the system. Denoting by P_p and P_s the ends of rod on the primary body p and secondary body s respectively, a single constraint is sufficient to satisfy the loop kinematics, expressing that the distance l_{rod} between points P_p and P_s remains constant at any time. The constraint reads:

$$\|\mathbf{x}(P_p) - \mathbf{x}(P_s)\|^2 - l_{rod}^2 = 0 \quad , \quad \forall\, t \text{ and } l_{rod} > 0$$

The procedure holds for a planar loop, also leading to a unique constraint.

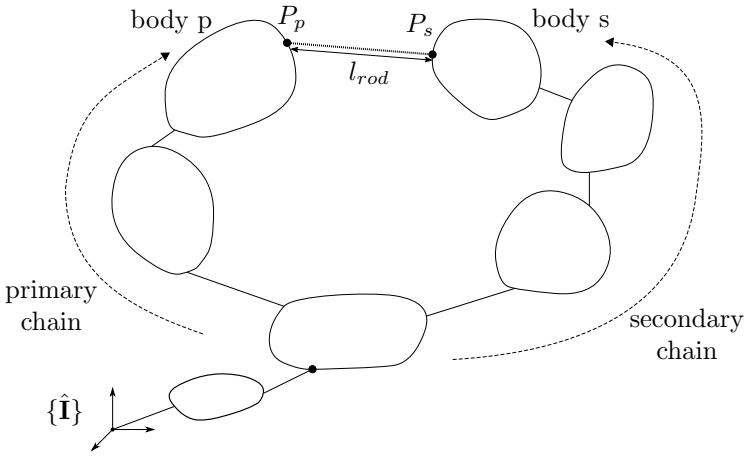


Figure 14: Cutting procedure of type 3 : removal of a connecting rod.

This third type of loop cut is introduced in the graphical editor MBsysPad. The *rod* cut is defined by connecting two anchor points. The length l_{rod} of the connecting rod, that must be strictly positive, has to be provided. The corresponding λ are provided in the results file as the force along the rod (negative value is traction, positive is compression).

4 Special Features

This Section describes how to implement and define specific constitutive laws that could not be derived by the symbolic equations generator alone (e.g., aerodynamic external forces or the Hooke's law for a spring element). It is a complement to the MBS formalism description (Section 3).

4.1 Driven variables

In various situations, some of the generalized joint coordinates q^i are no longer considered as variables because their motion is either simply *locked* (e.g., a skewed joint, see Section 2.4.3), or *forced* to obey a specific function of time t. In the present formalism, those variables are denoted *driven variables* and identified by the subscript c.
Formally if joint i is driven, then q^i, denoted q^i_c, is such that

$$q^i_c = f(t) \;\; ; \;\; \dot{q}^i_c = \dot{f}(t) \;\; ; \;\; \ddot{q}^i_c = \ddot{f}(t) \tag{26}$$

in which the function $f(t)$ is imposed (and implemented by the user). Strictly speaking, equation (26) represents an additional constraint on the system involving the sole variable q^i_c. Its simple form enables us to treat this constraint separately (from the set $h(q)$) in a more efficient way, as we shall see here below.

Some typical examples of driven variables are:

- a sinusoidal excitation: $q^i_c = a \sin \omega\, t \;\; ; \;\; \dot{q}^i_c = a\omega \cos \omega\, t \;\; ; \;\; \ddot{q}^i_c = -a\omega^2 \sin \omega\, t$
- the constant rotation of a body (ex. a crank): $q^i_c = \omega\, t \;\; ; \;\; \dot{q}^i_c = \omega \;\; ; \;\; \ddot{q}^i_c = 0$
- the locked position of a joint variable: $q^i_c = \alpha \;\; ; \;\; \dot{q}^i_c = 0 \;\; ; \;\; \ddot{q}^i_c = 0$

To take this type of variable into account in the formalism, and in particular in the coordinate partitioning reduction, we first consider them as being part of the *independent set of variables* q_u[12], but whose value obeys a prescribed law of type (26). Of course, driven variables *cannot* be time integrated[13]. The difference in the reduction process starts when expressing the final forms (22) and (24).

Formally, let us split q_u into *driven* variables q_{uc} and *really* independent variables q_{uu}:

$$q_u = \begin{pmatrix} q_{uu} \\ q_{uc} \end{pmatrix} \tag{27}$$

The semi-explicit system (22) can then be partitioned accordingly and becomes, using the Lagrange multipliers technique:

$$\begin{pmatrix} M_{r_{uu}} & M_{r_{uc}} \\ M_{r_{cu}} & M_{r_{cc}} \end{pmatrix} \begin{pmatrix} \ddot{q}_{uu} \\ \ddot{q}_{uc} \end{pmatrix} + \begin{pmatrix} F_{r_u} \\ F_{r_c} \end{pmatrix} = \begin{pmatrix} 0 \\ \lambda^c \end{pmatrix} \tag{28}$$

In which λ^c denotes the Lagrange multipliers associated with the driven motion constraints (26). The reduced form of the dynamic equations of a constrained multibod system containing driven variables finally reads:

$$M_{r_{uu}}\, \ddot{q}_{uu} + (F_{r_u} + M_{r_{uc}}\, \ddot{q}_{uc}) = 0 \tag{29}$$

The implicit system (24) can also be partitioned accordingly and becomes, using the Lagrange multipliers technique:

$$\begin{pmatrix} \phi_{uu} \\ \phi_{uc} \end{pmatrix} - \begin{pmatrix} Q_{uu} \\ Q_{uc} \end{pmatrix} - \begin{pmatrix} \xi_u \\ \xi_c \end{pmatrix} = \begin{pmatrix} 0 \\ \lambda^c \end{pmatrix} \tag{30}$$

with $\xi \triangleq B^t_{vu}(Q_v - \phi_v)$. The first line of (30) can be used to compute Q_{uu} for a given system dynamics.

$$Q_{uu} = \phi_{uu} - \xi_u \tag{31}$$

If the Lagrange multipliers λ^c associated with the driven motion are required as outputs, the lower part of equation (28) or equation (30) can both be used:

$$\lambda^c = M_{r_{cu}}\, \ddot{q}_{uu} + M_{r_{cc}}\, \ddot{q}_{uc} + F_{r_c} \tag{32}$$

$$\lambda^c = \phi_{uc} - Q_{uc} - \xi_c \tag{33}$$

[12] Obviously, considering a driven variable as a dependent one is intrinsically aberrant!
[13] or subject to any other numerical process.

Lagrange multipliers λ^c are often of practical interest[14]. Indeed they represent the joint forces (resp. torques) for driven prismatic (resp. revolute) joints which are of great interest for dimensioning purpose in a fully dynamic situation. In fact, those equations represent a local inverse dynamics problem.

Driven variables are first declared in the MBsysPad editor and then their motion law at position, velocity and acceleration levels (26) are introduced in the `user_DrivenJoint` function for which a template is provided in `\userfctR` project folder.

4.2 Joint forces and torques

These forces (torques) act inside the joints and their sole contribution, Q^i, to the i^{th} equations of motion in system (6), (7), (8) or (12) is the one which is aligned with the joint axis (ex. a friction torque in a door hinge or a viscous force in a telescopic device).

Using the definition (5) of the joint unit vector $\hat{\mathbf{e}}^i$, let us introduce the vector $\mathbf{Q}^i$ as the force (resp. the torque) on body i produced in the prismatic (resp. revolute) joint i by its parent h in the direction of the joint axis :

$$\mathbf{Q}^i = Q^i \, \hat{\mathbf{e}}^i$$

Q^i, the i^{th} element of Q in (6) or (8), represents the component of this force (resp. torque) along the joint vector axis. Let us point out that the action/reaction principle obviously holds for joint forces but the sole contribution – Q^i – to be introduced in the MBS equations (6) or (8) is the one applied "by the parent h on the child i".

The constitutive equation of Q^i is problem dependent and thus its description is left to the user. The following cases are given as illustrative examples:

- a perfect (prismatic/revolute) joint: $Q^i = 0$,
- a linear prismatic (resp. revolute) spring with stiffness K: $Q^i = -K\,(q^i - q^i_n)$ where q^i_n denotes the spring's neutral displacement $[m]$ (resp. $[rad]$),
- a linear prismatic (resp. revolute) damper (with damping coefficient D): $Q^i = -D\,\dot{q}^i$,
- a general law in terms of q^i and $\dot{q}^i$ and some parameters p^j: $Q^i = g\,(q^i,\, \dot{q}^i, p^j)$.

Joint forces and torques do not require any intervention in the graphical editor MBsysPad. Their constitutive law will be implemented in the user function `user_JointForces` for which a template is provided in the `\userfctR` project folder.

4.3 External forces and torques

External forces and torques represent any force or torque acting on the bodies, in addition to the joint forces/torques and to gravity. Their number, direction and nature are left to the user. In ROBOTRAN , they are gathered together for body i in a *unique* resultant force vector $\mathbf{frc}^i$ and a *unique* resultant torque vector $\mathbf{trq}^i$ with respect to the body center of mass CM^i. Let us point out that the torque resultant $\mathbf{trq}^i$ contains the sum of the moments of all the external forces plus the – possible – pure torque resultant applied to body i. A typical example of this is the tire-ground contact problem in vehicle dynamics: the contact force produces a moment with respect to the wheel center of mass, to which a pure torque arising from the local spin contact must be added to form the $\mathbf{trq}^i$ vector.

External forces and torques on body i must be introduced by the user in terms of their components in the inertial frame $[\hat{\mathbf{I}}]$:

$$\mathbf{frc}^i = [\hat{\mathbf{I}}]^t \, frc^i \quad \text{and} \quad \mathbf{trq}^i = [\hat{\mathbf{I}}]^t \, trq^i \tag{34}$$

External forces and torques are first graphically introduced by the user in the MBsysPad editor (via a `F` sensor). Afterwards – as for joint forces – their constitutive laws are implemented by the user in the function `user_ExtForces` for which a template is provided in the `\userfctR` project folder. The inputs and outputs of this user fonction are given in the inertial frame.

[14] sometimes more than those associated with the loop constraints

4.4 Link forces

By experience, numerous practical applications require the introduction of contributive forces (e.g., spring, actuator, damper, ...) acting between two points of the system (e.g., a parallel spring/damper element in a suspension). In ROBOTRAN , such an element is called a "link" and is defined between two anchor points. While acting internally between bodies of the structure, they are considered by the symbolic generator as a pair of action-reaction *external* forces (see above section) as the link is massless and not necessarily aligned along a joint axis.

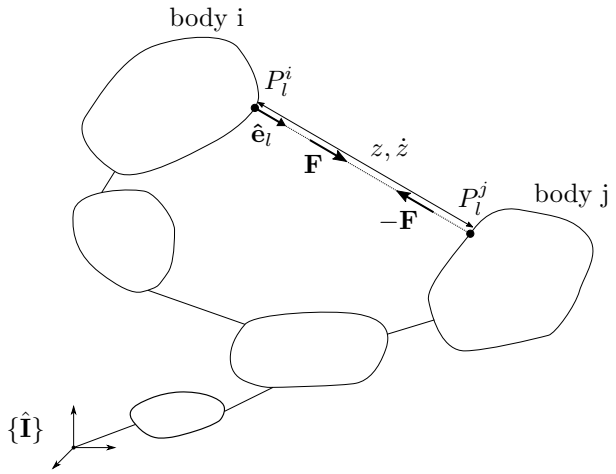


Figure 15: Link between body i and j.

Link forces as pairs of external forces and torques Let us define a unit vector $\hat{\mathbf{e}}_l$ along the link axis (from i to j) defined by anchor points P_l^i and P_l^j. This allows us to define vectors for the relative displacement and the relative velocity of point P_l^j with respect to point P_l^i along the link axis:

$$\mathbf{z} = z\,\hat{\mathbf{e}}_l \tag{35}$$

$$\dot{\mathbf{z}} = \dot{z}\,\hat{\mathbf{e}}_l \tag{36}$$

Let us now assume a given constitutive law f for the link force $\mathbf{F}$ which might depend on the displacement z[15] and the relative velocity $\dot{z}$ between the two anchor points.

$$F_{link} = f(z, \dot{z}) \tag{37}$$

The vector link force representing the force applied by body j to body i is given by:

$$\mathbf{F} = F_{link}\,\hat{\mathbf{e}}_l \tag{38}$$

Consequently, one can now compute the link force contribution to the resultant external vector force $\mathbf{frc}^i$ and external torque $\mathbf{trq}^i$ applied to the body i. For this purpose, we denote by $\mathbf{x}_l^i(q)$ the absolute vector position of the attachment point of the link on the body i, and similarly by $\mathbf{x}_l^j(q)$ the vector position of its attachment point on the body j.

The contributions to $\mathbf{frc}^i$ and $\mathbf{trq}^i$ are given by:

$$\begin{aligned} \mathbf{frc}^i &= \mathbf{F} \\ \mathbf{trq}^i &= \left(\mathbf{x}_l^i - \mathbf{x}_{CM}^i\right) \times \mathbf{F} \end{aligned} \tag{39}$$

Where $\mathbf{x}_{CM}^i$ denotes the position vector of the center of mass of the body i. Similarly, the contributions to the external resultant force/torque on body j read:

$$\begin{aligned} \mathbf{frc}^j &= -\mathbf{F} \\ \mathbf{trq}^j &= -\left(\mathbf{x}_l^j - \mathbf{x}_{CM}^j\right) \times \mathbf{F} \end{aligned} \tag{40}$$

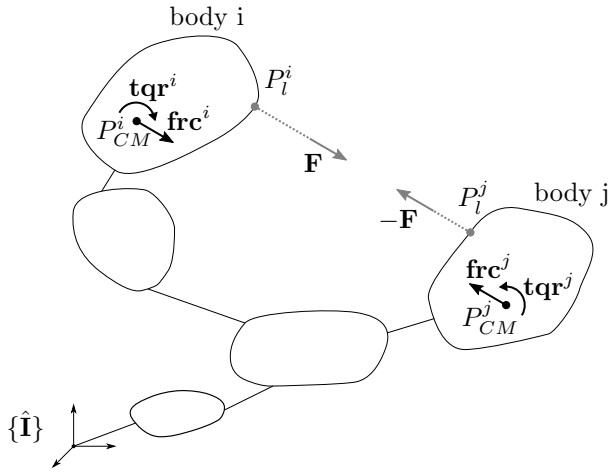


Figure 16: The link force is transformed into a 2 sets of external forces and torques.

In order to transform any constitutive equation of type (37) characterizing an internal link of the multibody system into *external forces/torques* $(\mathbf{frc}^i, \mathbf{trq}^i)$ applied to the involved bodies, we thus require, as functions of the joint generalized coordinates, the expressions of the position vectors of the two anchor points (in addition to the position vectors of the centre of mass CM^i).

Link forces : MBS implementation In order to compute the contribution of a link to the MBS dynamics, different calculation steps are realized either in the symbolic file (S) generated by ROBOTRAN or in the user function (U) implemented by the user:

- S – compute the absolute position and velocity of the two anchor points $\mathbf{x}_l^i$ and $\mathbf{x}_l^j$,
- S – compute the distance Z and/or the relative velocity $\dot{Z}$ between the two anchor points,
- U – introduce the constitutive equation $F_{link} f(Z, \dot{Z}, p, t)$ in which p denotes any kind of user parameters,
- S – project $\mathbf{F}_{link}$ and $-\mathbf{F}_{link}$ onto the inertial frame to respectively obtain $\mathbf{frc}^i$ and $\mathbf{frc}^j$,
- S – compute the resultant body torques $\mathbf{trq}^i$ and $\mathbf{trq}^j$ in the inertial frame.

Regarding the calculations, the user is only responsible for the constitutive equations F_{link}=$F_{link}(Z, \dot{Z}, p, t)$.

Link forces are first graphically introduced in the MBsysPad editor. The link element is defined by connecting two anchor points. Then, link force constitutive equations are implemented by the user in the user function `user_LinkForces` for which a template is provided in the `\userfctR` project folder.

4.5 Sensor kinematics

For various reasons, it can be of interest to obtain the symbolic expression of the forward kinematics of any sub-chain in the multibody system, i.e., the position, the orientation, the Jacobian matrix, the linear/angular velocities and/or the linear/angular accelerations of a given body (and of a particular point S of this body denoted "*sensor*") with respect to a given frame.

For instance, sub-chain forward kinematics is useful for:

- the introduction of a user constraint in the MBS (e.g., $h_a(q) = 0$ in Section 4.6)
- the introduction of a specific force whose constitutive equation requires the computation of the kinematics of a given point (e.g., the contact slip velocity of a wheel/ground force model);

[15]Note that the displacement z is always positive in the tridimensional.

- the computation of a specific result, such as the absolute position or acceleration of a point (e.g., the vertical acceleration of a car passenger).

Sub-chains are defined for a tree-like MBS by covering the system from the base to the leaves bodies (ex.: sub-chain 2-4-7 in Fig. 3_a and 2-4-9-10 in Fig. 3_b). Since closed structures are first cut in order to restore a tree, sub-chain kinematics can be used in any situation.
Of course, if one plans to compute the position $\mathbf{p}_{sens}(q)$ of a given sensor S, the loop constraints *must* be solved previously in such a way that the computation of vector $\mathbf{p}_{sens}$ is made with the correct values of the generalized coordinates $q = \{q_u, q_v\}$.

Let us consider in Fig. 17, the sub-chain $\{o...i\}$ in which o represents the original body and i the body carrying the sensor S, located by a constant position vector $\mathbf{d}_i = [\hat{\mathbf{X}}^i]^t\, d_i$ with respect to the joint i connection point (O'^i):

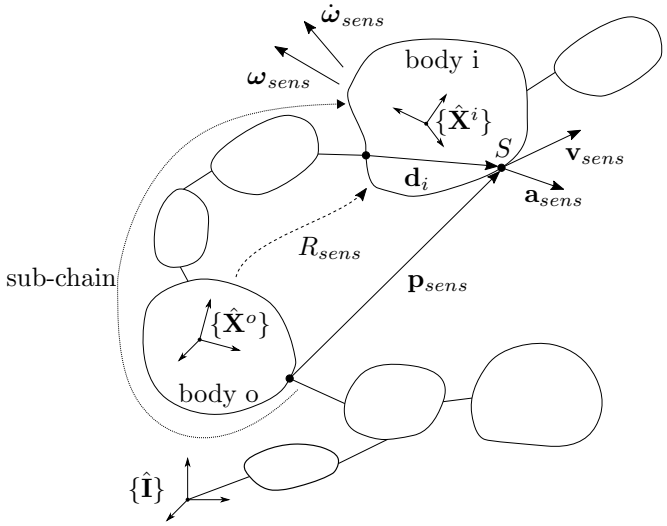


Figure 17: Sensor for a sub-chain kinematics.

For this sub-chain, the forward kinematics aims at computing:

- $\mathbf{p}_{sens} = [\hat{\mathbf{X}}^o]^t\, p_{sens}$: the position vector of sensor S with respect to point O'^o,
- R_{sens}: the rotation matrix from inertial frame $\{\hat{\mathbf{X}}^o\}$ to body frame $\{\hat{\mathbf{X}}^j\}$ such that $[\hat{\mathbf{X}}^j] = R_{sens}[\hat{\mathbf{X}}^o]$ and such that, for a given vector $\mathbf{x}$, its components in these frames obey: $[x^i] = R_{sens}\,[x^o]$,
- $\mathbf{v}_{sens} = [\hat{\mathbf{X}}^o]^t\, v_{sens}$: the relative velocity of point S with respect to frame $\{\hat{\mathbf{X}}^o\}$,
- $\boldsymbol{\omega}_{sens} = [\hat{\mathbf{X}}^o]^t\, \omega_{sens}$: the relative angular velocity of frame $\{\hat{\mathbf{X}}^i\}$ with respect to frame $\{\hat{\mathbf{X}}^o\}$,
- $J_{sens} = \begin{pmatrix} \frac{\partial v_{sens}}{\partial \dot{q}^t} \\ \frac{\partial\, \omega_{sens}}{\partial \dot{q}^t} \end{pmatrix}$, the *Jacobian* matrix containing the partial derivative of the components v_{sens} with respect to $\dot{q}$ (rows 1,2,3) and of the components ω_{sens} with respect to $\dot{q}$ (rows 4,5,6) (N.B.: column k corresponds to the generalized joint velocity $\dot{q}_k$),
- $\mathbf{a}_{sens} = [\hat{\mathbf{X}}^o]^t\, a_{sens}$: the relative acceleration of point S with respect to frame $\{\hat{\mathbf{X}}^o\}$
- $\dot{\boldsymbol{\omega}}_{sens} = [\hat{\mathbf{X}}^o]^t\, \dot{\omega}_{sens}$: the relative angular acceleration of frame $\{\hat{\mathbf{X}}^i\}$ with respect to frame $\{\hat{\mathbf{X}}^o\}$.

Most of the time, the desired computation is related to *absolute* quantities (position, velocities, ...) with respect to the inertial frame $\{\hat{\mathbf{I}}\}$. In that case, the original body o will simply be the base (body 0).

Sensors are first graphically introduced in the MBsysPad editor. The sensor is always associated with an anchor point. The computation of their kinematics with respect to the inertial frame is automatically generated in symbolic form in the function `mbs_projectname_sensor` that the user can call for any purpose and from any of his user functions.

4.6 Algebraic constraints

Besides the loop constraints described in section 3.2.3, user constraints, denoted $h_a(q) = 0$, can also be introduced between the generalized coordinates. For instance, they can be used in order to impose parallelism between two bodies ($\theta^i = \theta^j$) or to model the motion in a screw joint with a lead: $\beta\ q^i = \beta\, q^j$, with $\beta[\frac{m}{rad}]$, etc.

$$h_a(q) = 0 \tag{41}$$

For these user constraints, the associated Jacobian matrix $J_a(q)$ and the quadratic term $(\dot{J}\dot{q}(q,\dot{q}))_a$ appearing in the constraints derivatives need to be explicitly provided.

The user constraints themselves are implemented in the user functions `user_cons_hJ` (for the constraints and the Jacobian matrix) and `user_cons_jdqd` (for the quadratic term). In addition, the number of user constraints h_a has to be defined by the user himself in the main simulation file.

4.7 Constitutive differential equations

Differential Equations can be added to the existing multibody equations of motion (22) or (24). Such feature is very useful to deal with multiphysics problems (e.g., electrical motors, pneumatic air flow, PID controller, ...), by coupling the domains at the equational level rather than by resorting to cosimulation techniques that complexify the numerical process. The additional state variables, called u_x, are added to the set of mechanical variables q_u and $\dot{q}_u$[16]. In a time integration context, there are $2n_u + n_{u_x}$ first-order differential equations, the first $2n_u$ are simply derived from equation (22) and the n_{u_x} additional differential equations are specified by the user.

$$\begin{cases} \dfrac{d}{dt}\begin{pmatrix} q_u \\ \dot{q}_u \end{pmatrix} = \begin{pmatrix} \dot{q}_u \\ f(q_u, \dot{q}_u) \end{pmatrix} \\ \\ \dot{u}_x = f_{usr}(u_x, q_u, \dot{q}_u) \end{cases} \tag{42}$$

Where f stands for the solution of the linear system (22) for the independent generalized acceleration $\ddot{q}_u$.

The expression for the differential equations f_{usr} is specified in the user function `user_derivative`. In addition, the number of state variables u_x has to be defined by the user himself in the main simulation file.

[16] where u denotes the non-driven independent variables in that case

5 Analysis Modules

This section briefly describes the numerical methods that are widely used in multibody dynamics, and implemented in the ROBOTRAN software. For the sake of clarity, we consider the case without driven variables q_c; hence we have $q_u = q_{uu}$.

5.1 Coordinate partitioning

In order to solve the DAE system inherent to the constrained MBS, an order reduction is performed based on the Coordinate Partitioning method (see section 3.2.1). Hence, once the constraints $h(q) = 0$ and their time derivatives have been established, a partitioning of the coordinates must be done: $q = \{q_u, q_v\}$ (see eq. 16), in which the number of independent coordinates q_u corresponds exactly to the number of d.o.f. of the system.

The problem is to select the "best" choice of q_u and q_v for numerical conditioning purposes. The latter is mostly related to the constraint sub-Jacobian matrix J_v which must be inverted[17] at different places (see Section 3.2.1):

- to make the Newton-Raphson converge (eq. (19));
- to compute the coupling matrix B_{vu} (eq. (18));
- to compute the Lagrange multipliers $\lambda = ({J_v}^t)^{-1}$... (eq. (25)).

The best conditioned this matrix will be, the more robust the system reduction will be. As soon as there are a lot of possibilities (or combinations) in selecting m variables q_v among $n + m$ variables q, we will prefer the one which makes J_v very well conditioned: this can be performed by means of a $LU-$factorization of the whole rectangular matrix $J(q)$ with full (rows and columns) or partial (columns) pivoting. This well-known technique aims at placing the largest pivots – in absolute value – on the factorized matrix diagonal, by permuting rows and/or columns accordingly. Regarding the constraints, permuting columns i and j amounts to permute variables q_i and q_j.

Once the factorization is finished, the m first columns will be associated with the best dependent variables q_v, the other ones being the independent ones q_u. An interesting sub-product of this permutation process is the detection of possible redundant constraints (that is, constraints which are linear combinations of other ones) or of trivial constraints ($0 = 0$): in that case, the LU factorization – done with permutations – stops before reaching the last line of matrix J, by detecting a null pivot on the diagonal. This last factorization step indicates the rank r of J.

If the rank of J equals the number of constraints, $r = m$, there are no redundant constraints (J is full rank). Else, if $r < m$, redundant constraints are detected for the considered configuration and will be disregarded in all subsequent processes. Only r independent constraints will be considered and r dependent variables q_v. The system has thus $n - r$ degrees of freedom.

The coordinate partitioning is normally performed once for a given analysis. As said before, many combinations are possible and a intuitive reasoning, based on the system morphology is often a good start (ex. for the slider-crank mechanism on top of Fig. 10, the crank rotation can obviously be chosen as the independent variable q_u). For a tree-like MBS exempt from constraints, there are obviously no needs for partitioning.

First, the user can — a priori — indicate his own choice of partitioning between independent and dependent variables in the graphical editor MBsysPad. Then, the CoordPart module takes the user's choice into account (verification/validation) and proposes the best partitioning. This module is a prerequisite to any other analysis on constrained MBS. The resulting partitioning (i.e., the partition of q_u and q_v) is stored in the data structure by the CoordPart module which also closes the loops and solve the constraints $h(q) = 0$. Afterwards, the stored partitioning can be used within other modules which require the system reduction.

[17] more precisely, factorized

5.2 Time integration

The motion of the multibody system $(q_u(t), \dot{q}_u(t))$ starting from an initial configuration $(q(t=0), \dot{q}(t=0))$ can be predicted by time-integrating the acceleration $\ddot{q}_u(t)$. The principle for a tree-like (i.e., unconstrained) and a constrained MBS is exactly the same. The framework for the constrained MBS is first presented. Then the tree-like case is briefly treated.

To time integrate the motion, the *direct dynamics*[18] of the constrained MBS can be used, remembering that the accelerations $\ddot{q}_u$ can be obtained by solving the linear system (22).

$$\ddot{q}_u = f(q_u, \dot{q}_u)$$

For this purpose, rather than "blindly" inverting the reduced mass matrix $M_r(q_u)$, we preferably resort to linear algebra techniques (i.e., a Cholesky decomposition of the symmetric positive-definite mass matrix M_r followed by a backward/forward substitution applied to the linear system).

The sequence of computations for time integrating the *constrained* MBS equations of motion (22) is illustrated in the block diagram of Figure 18 largely based on that of Figure 11.

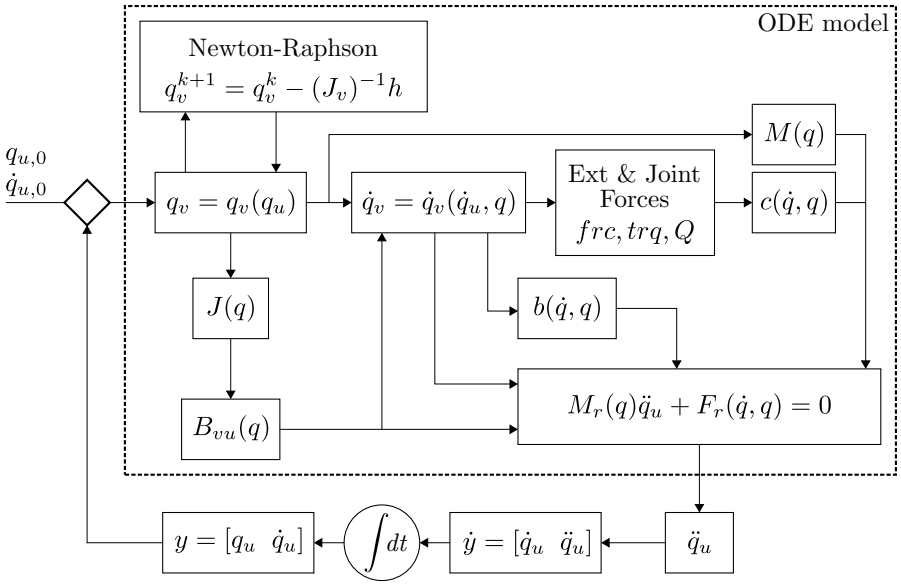


Figure 18: Time integration process of a constrained MBS.

The dashed box represents the *ODE model* whose inputs are the independent position q_u and velocities $\dot{q}_u$ and outputs are the independent accelerations $\ddot{q}_u$. This model is perfectly suited to classical integrator scheme after reducing the system order from two to one:

$$y = \begin{pmatrix} q_u \\ \dot{q}_u \end{pmatrix} \rightarrow \dot{y} = \frac{d}{dt}\begin{pmatrix} q_u \\ \dot{q}_u \end{pmatrix} = \begin{pmatrix} \dot{q}_u \\ f(q_u, \dot{q}_u) \end{pmatrix} \tag{43}$$

For the case of an *unconstrained* MBS, the generalized accelerations are computed based on system (6).

$$\ddot{q} = f(q, \dot{q})$$

In this specific case, the ODE *model* solely consists in the computation of frc, trq, Q, M, c as functions of q and $\dot{q}$ and the solving of the linear system (through the Cholesky decomposition of M).

Time integration is performed via the Dirdyna module. The time integrator can be chosen according to the problem at hand through the setting of specific options.

[18] the direct dynamics is the computation of the generalized joint accelerations $\ddot{q}_u$ for a given configuration $(q_u, \dot{q}_u)$ of the system to which forces and torques are applied.

5.3 Equilibrium

Generally, an equilibrium is a configuration for which the forces are in balance with no absolute acceleration among the multibody system. In order to compute it, the semi-explicit reduced equations of motion (22) or its implicit form (24) can be used in their residual forms[19]:

$$R_r(q_u, \dot{q}_u, \ddot{q}_u) = M_r \ddot{q}_u + F_r = \phi_u - Q_u - B_{vu}^t(Q_v - \phi_v) = 0 \tag{44}$$

Thereafter, the equilibrium variables are denoted as x and the equilibrium equations as F. The number of equilibrium variables n_x must be consistent with the number of equations. A solution of the equilibrium problem is generally denoted with a superscript *.

$$F(x^*) = 0 \tag{45}$$

Equilibrium is also an indispensable pre-process for the Modal Analysis (see section 5.4).

5.3.1 Static equilibrium

If a static equilibrium is considered, the joint velocities and accelerations are null (i.e., $\dot{q}_u = 0$ and $\ddot{q}_u = 0$). The reduced form of the equations of motion transform in a set of non-linear algebraic equations in terms of the independent coordinates, q_u:

$$R_r(0, 0, q_u) = F_r(q_u) = 0 \tag{46}$$

This nonlinear system of equations can be solved iteratively with the Newton-Raphson algorithm:

$$x^{k+1} = x^k - \left(\frac{\partial F}{\partial x}\right)^{-1} F\,|_{x=x^k} \tag{47}$$

with $x = q_u$ and $F = F_r$ in case of static equilibrium.

Even though the iteration process is performed on the *independent* coordinates: $x = q_u$ (excluding the driven variables), the equilibrium configuration concerns all the variables including the dependent ones.

5.3.2 Non-sensitive variables

Some variables are sometimes non-sensitive with respect to the equilibrium configuration in the sense that their *algebraic* value does *not* affect the term F, and thus the equilibrium (e.g., the x and y position of a stationary vehicle on a flat ground). A non-sensitive variable x_i leads to a null gradient for the equilibrium equations which can make the algorithm fail:

$$\frac{\partial F}{\partial x^i} = 0 \tag{48}$$

These non-sensitive variables must be either omitted in the computation or replaced by other sensitive variables. In the first case, their values can then be chosen arbitrarily and will not evolve during the iterative process.

5.3.3 Exchange of variables

For some application, equilibrium variables must be exchanged and the equilibrium procedure can be performed on a different set of n_x variables, which are by default the independent coordinates. One of the independent coordinates q_i is then replaced by another parameter of the system(e.g., a mass, a stiffness coefficient, a length,...), denoted x_{ch}.

$$x = [q_u \backslash \{q_i\}, x_{ch}] \tag{49}$$

The equilibrium equations remain the same, i.e., $R_r = 0$. However in this case, the replaced coordinates q_i are frozen and kept constant during the equilibrium process. It is important that the exchange makes sense physically and that the exchanged variables x_{ch} (i.e., the new ones) are sensitive to the equilibrium equations.

[19] numerically, it is always more efficient to work with the implicit form.

5.3.4 Extra variables

In line with Section 4.7, equilibrium equations can be added to the multibody equations of motion. Such feature can be used to deal with multiphysics problems or to make some adjustment on the static configuration of the MBS (e.g., adjusting the track rod length of a car to obtain a given toe-in or toe-out characteristics for the front wheels). This implies to define extra equations of equilibrium and extra equilibrium variable such that:

$$x = \begin{pmatrix} q_u \\ x_e \end{pmatrix} \qquad F = \begin{pmatrix} R_r \\ F_e \end{pmatrix} \tag{50}$$

As previously, the Newton-Raphson will iterate on the $n_x = n_u + n_e$ variables until it finds the equilibrium configuration x^* that satisfies $F(x) = 0$.

5.3.5 Dynamic Equilibrium

When the velocities and/or the accelerations are non-zero, the equilibrium is said to be *dynamic*. For instance, forces (or torques) F_{dyn} act against resistance forces (often friction) to maintain the vehicle in a continuous constant motion. The main application is vehicle dynamics for which constant speed straight line and steady-state cornering are of interest. Computing such equilibrium utilizes the notion of non-sensitive variables and variable exchanges.

The independent position q_i associated with a non-zero velocity $\dot{q}_i$ is not relevant and is considered non-sensitive (e.g., the angular position of a wheel q^i). This variable q^i can be removed from the set of equilibrium variables and exchanged for a sensitive one. There are two ways to compute the steady-state equilibrium. Depending on which approach is chosen, the exchange variable is the force F_{dyn} or the speed $\dot{q}_i$. In other words,

- Fix the speed and determine the forces (or torques) necessary to maintain this speed

$$\text{Fixed } \dot{q}_i \qquad x = [q_u \backslash \{q_i\}, F_{dyn}] \quad \rightarrow \quad F(x) = 0$$

- Fix the forces (or torques) and determine the corresponding speed

$$\text{Fixed } F_{dyn} \qquad x = [q_u \backslash \{q_i\}, \dot{q}_i] \quad \rightarrow \quad F(x) = 0$$

It is important to notice that the exchange only concerns the joint i and that the other independent variables from q_u are still part of the equilibrium process.

Equilibrium is preferably performed via the Equilibrium module. From a good initial set of the independent variables q_u (stored in the data structure), the module tries to find the static equilibrium solution (q_u^*) iteratively. The module option allows for variable exchanging or equation addition. This allows the user to perform

- an inverse equilibrium: by replacing a q_u subset by a body mass, a spring stiffness, a body length, etc.;
- a dynamic equilibrium: by replacing a non-sensitive coordinate q_u by its velocity $\dot{q}_u$.

5.4 Modal analysis

A modal analysis is the study of the dynamic properties of free systems in the frequency domain. Through the modal analysis, the eigenmodes and eigenfrequencies of the system are calculated. In the case of multibody system, it is particularly interesting for control purpose as it allows to identify the stable and unstable modes of the system. It is also useful to study the vibration behavior of the system and to identify unstable modes or critical frequencies of excitation that could lead to resonances. The modal analysis is performed on the multibody system alone without considering any additional constitutive ODE that could have been added through the user derivative (Section 4.7).

5.4.1 Eigensystem

Before carrying out a modal analysis, a linearization of the reduced set of equations (22) around the equilibrium configuration x^* is indispensable. For a general MBS model, this procedure must be performed numerically, producing the following linearized system:

$$M_r(q_u^*)\,\Delta\ddot{q}_u + G_r(q_u^*, \dot{q}_u^*)\,\Delta\dot{q}_u + K_r(q_u^*)\,\Delta q_u = 0 \tag{51}$$

in which $\Delta q_u = q_u - q_u^*$, M_r, G_r and K_r respectively stand for the constant mass, gyroscopic/damping and stiffness tangent matrices (size $n-m$ = number of d.o.f.) found by a linearization procedure. To transform this second-order linear system into the following first order form

$$\dot{x} = Ax, \tag{52}$$

Let us define x as follows:

$$x = \begin{pmatrix} \Delta q_u \\ \Delta\dot{q}_u \end{pmatrix} \;\rightarrow\; \dot{x} = \begin{pmatrix} \Delta\dot{q}_u \\ \Delta\ddot{q}_u \end{pmatrix}$$

System (51) can be written as:

$$\begin{pmatrix} I & 0 \\ 0 & M_r \end{pmatrix}\dot{x} + \begin{pmatrix} 0 & -I \\ K_r & G_r \end{pmatrix} x = 0 \tag{53}$$

which can finally be written on the desired form (52):

$$\dot{x} = Ax \tag{54}$$

with

$$A \triangleq -\begin{pmatrix} I & 0 \\ 0 & M_r \end{pmatrix}^{-1}\begin{pmatrix} 0 & -I \\ K_r & G_r \end{pmatrix} = \begin{pmatrix} 0 & I \\ -M_r^{-1}K_r & -M_r^{-1}G_r \end{pmatrix} \tag{55}$$

Considering a particular solution $x_i(t)$ of (54) whose components have the same time dependency (harmonic, exponential):

$$x_i(t) = v_i e^{\lambda t}$$

with $x_i = \{x_i^1, x_i^2, ..., x_i^{2(n-m)}\}$, the previous differential equation $\dot{x} = Ax$ becomes an algebraic system ("eigenvalue system"):

$$(A - \lambda E)\,v = 0 \tag{56}$$

In which E stands for the unit matrix.

From system (56) the $2(n-m)$ eigenvalues λ_i (and their eigenvectors v_i) can be found[20] by solving:

$$det(A - \lambda E) = 0$$

The $2(n-m)$ eigenvalues λ_i can be:

- real, corresponding to a non-oscillatory stable ($\lambda_i < 0$) or unstable ($\lambda_i > 0$) eigenmode
- purely imaginary (conjugate pair), corresponding to infinitively oscillating undamped eigenmode
- complex (conjugate pair), corresponding to an oscillatory stable ($re(\lambda_i) < 0$) or unstable ($re(\lambda_i) > 0$) eigenmode
- null, corresponding to a rigid mode

From (56), a maximum of $2(n-m)$ eigenvectors v_i can then be found. Let us note that in case of multiple eigenvalues, only one eigenvector will be produced and thus the number of eigenvectors will be less than $2(n-m)$. Moreover, in case of complex conjugate eigenvalues, a pair of complex conjugate eigenvectors is produced, giving rise to a single (and not two) eigenmode.

[20] For large systems, advanced numerical algorithms are used to find the λ's (e.g., the "eig" function in MATLAB).

5.4.2 Eigenmodes

Referring to the solution (λ_i, v_i) of system (56), each of them corresponds to a so-called eigenmode (characterized by an eigenvalue and an eigenvector) of the system around its — static or steady-state — equilibrium configuration: this is the purpose of modal analysis. For oscillatory eigenmodes ($im(\lambda_i) \neq 0$), one can extract, from the eigenvalues:

- the undamped natural circular frequency: $\omega_0 = |\lambda_i| = \sqrt{(re(\lambda_i))^2 + (im(\lambda_i))^2}$ [rad/s];
- the undamped frequency: $\nu_0 = \frac{\omega_0}{2\pi}$ [Hz];
- the damping factor: $\zeta = 100 * \cos(\arctan(-\frac{im(\lambda_i)}{re(\lambda_i)}))$ [%];
- the natural circular frequency of the – damped – mode: $\omega = \omega_0\sqrt{1-\zeta^2}$ [rad/s];
- the frequency of the – damped – mode: $\nu = \frac{\omega}{2\pi}$ [Hz].

For any eigenmode i, the associated eigenvector v_i describes, via its $n-m$ components, the associated motion. With this respect, an eigenvector will be advantageously written in terms of:

- the module[21] r_i^j of each of its component j: $r_i^j = \sqrt{(re(v_i^j))^2 + (im(v_i^j))^2}, j = 1 : n-m$
- the phase[22] ϕ_i^j of each of its component j: $\phi_i^j = \arctan\frac{im(v_i^j)}{re(v_i^j)}$

Modal analysis has to be performed around an equilibrium configuration. Afterwards, the system linearization and modal analysis are done in the Modal module. Results are the eigenvalues (including frequency, damping factor, ...) and the associated eigenvectors whose 3D animation (in the MBsysPad program) is straitghtforward and allows the user to clearly visualize the system moving according to a given eigenmode.

[21] Being computed, each eigenvector is normalized with its maximum component set to 1 for the module

[22] 0 rad for the phase

5.5 Inverse Dynamics Problem

The *inverse dynamics* of a multibody system is the computation of the generalized joint forces (torques) Q applied in the joints for a given configuration $(q, \dot{q}, \ddot{q})$ or a desired trajectory $(q(t), \dot{q}(t), \ddot{q}(t))$ of the MBS. The implicit formulation[23] Inverse dynamics is usefull for instance to control robot manipulators by predicting the feed-foward torques, or to compute the joint net torques in the musculoskelettal model of the human body.

5.5.1 Unconstrained MBS

For unconstrained MBS, inverse dynamics can be directly obtained from the implicit form of the equations of motion (7). In the context of inverse dynamics, let us distinguish two contributions in the vector Q: the *passive* generalized joint forces $Q_p(q, \dot{q})$ (arising from internal joint stiffness, friction, etc.) and the *active* generalized joint forces Q_a (referring to the joint actuation). Generally speaking, each joint may involve these two components, the latter being the unknown of the *inverse dynamics* problem, i.e.:

$$Q_a = \phi(q, \dot{q}, \ddot{q}) - Q_p(q, \dot{q}) \tag{57}$$

in which ϕ contains the dynamics of the MBS (including external forces and torques) induced by the desired trajectory. For sake of clarity, the dynamic equations associated with the driven — or locked — coordinates q_{uc} have been excluded from (57).

This equation represents the inverse dynamics of an unconstrained MBS, in which all the non-driven joints are actuated. There are as many unknown Q_a as equations: overactuation or underactuation is thus not possible.

5.5.2 Constrained MBS

As for a unconstrained tree-like MBS, one can be interested in computing the joint actuation in a constrained MBS (i.e. with closed-loops) for a desired trajectory $(q(t), \dot{q}(t), \ddot{q}(t))$. This is the case of parallel manipulators for instance. Like for the direct dynamics, the coordinate partitioning is used to reduce the implicit form of the dynamic equations (30). For a fully actuated MBS with $n - m$ degrees of freedom, $n - m$ active joint forces are unknown.

For sake of clarity, let us first consider the case where the actuators are located in the independent joints (coordinates q_u)[24]. As regards the passive generalized forces, they can be applied anywhere, either in independent or in dependent joints. Hence, the inverse dynamics of a constrained MBS reads:

$$Q_a = Q_{u,a} = \phi_u - Q_{u,p} + B_{vu}^t(\phi_v - Q_{v,p}) \tag{58}$$

Generally speaking, the $n - m$ actuators can practically be located in any joints, independently from the initial coordinate partitioning. Hence, considering a partitioning $\{q_e, q_f\}$ in which subscripts e and f refer to the active and passive joints respectively, the previous equation becomes:

$$Q_a = Q_{e,a} = \phi_e - Q_{e,p} + B_{fe}^t(\phi_f - Q_{f,p}) \tag{59}$$

in which the existence of the coupling matrix $B_{fe} \triangleq -(J_f)^{-1} J_e$ imposes the sub-matrix J_f to be regular.

The implementation of the passive joint forces and torques must be done in the "user_JointForces" function as for the direct dynamics (see section 4.2).

[23] also called the inverse dynamic model.
[24] or q_{uu} if driven joints are present

5.6 Constraints solution (at position level)

As mentioned earlier, to solve the constraints (i.e., to close the loops for a given configuration), the Newton-Raphson (N.R. in short) iterative algorithm is used for successive estimations of q_v:

$$q_v^{k+1} = q_v^k - (J_v)^{-1}\, h\,|_{q_v=q_v^k} \tag{60}$$

Where k is the iteration step number, to find a solution q_v^* with a predefined threshold (ex.: $\epsilon = 1e-9$). The right-hand side of (60) is evaluated for $q_v = q_v^k$ and for values of q_u and q_c corresponding to the configuration of the system.

A particular attention must be paid to the initial value of q_v (i.e., $q_v^{k=0}$). In a time integration process for instance, for which the constraints must be solved by N.R. at each time step, the previous step values of q_v: $q_{v,t}^*$ will be automatically used as initial value of the next time step : $q_{v,t+1}^0$.

The N.R. method for solving the constraints is automatically and internally performed by all modules during the associated numerical process (coordinate partitioning, inverse dynamics, direct dynamics, equilibrium and modal analysis).

References

[1] J.-C. Samin and P. Fisette, *Symbolic Modeling of Multibody Systems*. Solid Mechanics and Its Applications, Springer Netherlands, 2003.

[2] N. Docquier, A. Poncelet, and P. Fisette, "ROBOTRAN: a powerful symbolic generator of multibody models," *Mechanical Sciences*, vol. 4, no. 1, pp. 199 – 219, 2013.

[3] R. A. Wehage and E. J. Haug, "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems," *Journal of Mechanical Design*, vol. 104, pp. 247–255, Jan. 1982.

A Link forces in vehicles problem

The link forces, which are mainly inspired by vehicle suspension problems, have been automated in ROBOTRAN as a practical means for reducing computational loads wherever possible. A typical example is illustrated in Fig. 19 which represents a five-bar left suspension system of a car.

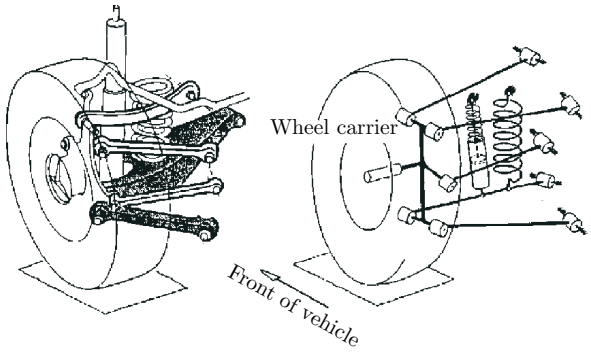


Figure 19: Five-bar suspension system

In this suspension, a rod is connected to the chassis via the suspension elements: a parallel spring/damper device. If the mass and inertia of these elements are negligible (with respect to the remainder of the vehicle), their dynamic effects can thus be simply modeled by means of forces respectively applied to the chassis ($\mathbf{F}$) and to a rod ($-\mathbf{F}$). The resulting multibody representation is sketched in Fig. 20.

From *internal* to *external* force formulation To make the following developments more legible, let us suppose that the link force constitutive law (ex. suspension spring) has a linear characteristic through a stiffness k and a neutral length Z_o. The component F_{link} of the vector force $\mathbf{F}_{link}$ due to the spring is assumed to be given by the user as:

$$F_{link} = +k(Z - Z_o) \tag{61}$$

where Z is the actual length of the spring. This vector force is applied to the material point of the chassis which serves as an attachment point of the spring (i.e., the anchor point of the link), and is aligned with the spring direction. Based on this constitutive equation, we can now compute the spring contribution to the resultant external vector force $\mathbf{frc}^{chas}$ and external torque $\mathbf{trq}^{chas}$ applied to the chassis. For this purpose, we denote by $\mathbf{x}_{spr}^{chas}(q)$ the absolute vector position of the attachment point of the spring on the chassis, and similarly by $\mathbf{x}_{spr}^{rod}(q)$ the vector position of its attachment point on the rod. The vector link force is thus given by:

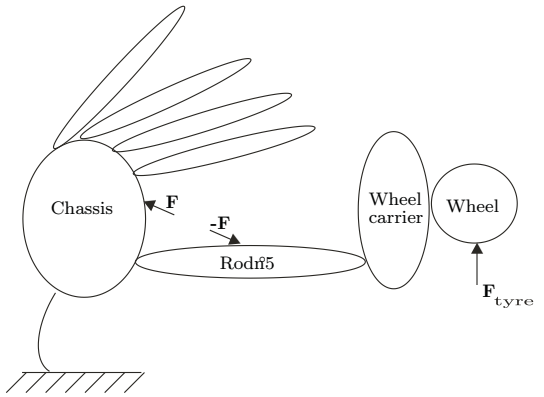


Figure 20: Multibody model of the suspension.

$$\mathbf{F}_{link} = k\,(Z - Z_0)\frac{\left(\mathbf{x}_{spr}^{rod} - \mathbf{x}_{spr}^{chas}\right)}{\parallel \mathbf{x}_{spr}^{rod} - \mathbf{x}_{spr}^{chas} \parallel} \tag{62}$$

and the contributions to $\mathbf{frc}^{chas}$ and $\mathbf{trq}^{chas}$ are given by:

$$\begin{aligned} \mathbf{frc}^{chas} &= \mathbf{F}_{link} \\ \mathbf{trq}^{chas} &= \left(\mathbf{x}_{spr}^{chas} - \mathbf{x}^{chas}\right) \times \mathbf{F}_{link} \end{aligned} \tag{63}$$

where $\mathbf{x}^{chas}$ denotes the position vector of the center of mass CM^{chas} of the body *chassis*. Similarly, the contributions to the external resultant force/torque on body *rod* read:

$$\begin{aligned} \mathbf{frc}^{rod} &= -\mathbf{F}_{link} \\ \mathbf{trq}^{rod} &= -\left(\mathbf{x}_{spr}^{rod} - \mathbf{x}^{rod}\right) \times \mathbf{F}_{link} \end{aligned} \tag{64}$$

In order to transform any constitutive equation of type (61) characterizing an internal link of the multibody system into *external forces/torques* $(\mathbf{frc}^i, \mathbf{trq}^i)$ applied to the involved bodies, we thus require, as functions of the joint generalized coordinates, the expressions of the position vectors of the two anchor points (in addition to the position vectors of the center of mass CM^i).

The method used above for the spring can also be applied in modeling a constitutive law which depends on the link velocity (ex. for the suspension damper). Assuming (again) a linear characteristic of the law:

$$F_{link} = c\,\dot{Z} \tag{65}$$

the vector damper link force is given by:

$$\mathbf{F}_{link} = c\,\dot{Z}\;\frac{\left(\mathbf{x}_{damp}^{rod} - \mathbf{x}_{damp}^{chas}\right)}{\parallel \mathbf{x}_{damp}^{rod} - \mathbf{x}_{damp}^{chas} \parallel} \tag{66}$$

which is easily transformed into external resultant forces/torques $(\mathbf{frc}^i, \mathbf{trq}^i)$ with formulae similar to (63) and (64). In addition to the anchor point position vectors, the user now needs, in order to compute $\dot{Z}$, the expressions of the velocities $\dot{\mathbf{x}}_{damp}^{chas}$ and $\dot{\mathbf{x}}_{damp}^{rod}$ in terms of the joint coordinates q and velocities $\dot{q}$.

B Linearization and Modal analysis for multiphysic models

When the constitutive ODE system (Section 4.7) are taken into account in the modal analysis, the linearization has to be performed on both the equations of motion and the constitutive ODE, which are denoted R_r (i.e., the residual form of the equations) and f_{usr} from equation (42). In addition, the linearization has to be performed with respect to the ODE states u_x but also more generally on the independent accelerations $\ddot{q}_u$ (to take for example inerter terms) and the state derivatives $\dot{u}_x$.

$$J_r\,\Delta\ddot{q}_u + S_r\Delta\dot{u}_x + G_r\,\Delta\dot{q}_u + K_r\,\Delta q_u + L_r\Delta u_x = 0 \tag{67}$$

$$\Delta\dot{u}_x = J_x\Delta\ddot{q}_u + G_x\Delta\dot{q}_u + K_x\Delta q_u + L_x\Delta u_x \tag{68}$$

Note that M_r has been replaced by J_r in the first equation to take into account non-linear force terms that would depend on joint accelerations (e.g., inerter and J-suspension). Let us define x as follows:

$$x = \begin{pmatrix} \Delta q_u \\ \Delta\dot{q}_u \\ \Delta u_x \end{pmatrix} \;\rightarrow\; \dot{x} = \begin{pmatrix} \Delta\dot{q}_u \\ \Delta\ddot{q}_u \\ \Delta\dot{u}_x \end{pmatrix}$$

System (67-68) can be written as:

$$\begin{pmatrix} I & 0 & 0 \\ 0 & J_r & S_r \\ 0 & J_x & I \end{pmatrix}\dot{x} + \begin{pmatrix} 0 & -I & 0 \\ K_r & G_r & L_r \\ K_x & G_x & L_x \end{pmatrix} x = 0 \tag{69}$$

Which can finally be written on the desired form (52) by numerically solving system (69).

$$\dot{x} = Ax \tag{70}$$

Finally, let us note that the cases $J_r \neq M_r$, $S_r \neq 0$ and $J_x \neq 0$ are not often encountered. Generally, the extra-diagonal terms simply disappear and the A matrix can be computed block by block.

$$A \triangleq -\begin{pmatrix} I & 0 & 0 \\ 0 & M_r & 0 \\ 0 & 0 & I \end{pmatrix}^{-1}\begin{pmatrix} 0 & -I & 0 \\ K_r & G_r & L_r \\ K_x & G_x & L_x \end{pmatrix} = \begin{pmatrix} 0 & I & 0 \\ -M_r^{-1}K_r & -M_r^{-1}G_r & -M_r^{-1}L_r \\ K_x & G_x & L_x \end{pmatrix} \tag{71}$$