MoSs package documentation

Wolfram Mathematica® 10.0 package for modular modelling of multibody systems

Renato Maia Matarazzo Orsino

November 13, 2015

Introduction

MoSs, acronym for **Mo**dular **Mo**delling of Multibody Systems Based on **S**ub**s**ystems Models, is a Mathematica Package developed by Renato Maia Matarazzo Orsino based on the modular modeling methodology for multibody systems presented in [**?**].

The package, developed in Wolfram Mathematica 10.0, aids in the implementation of a modular modelling algorithm in which, the user only needs to provide the mathematical models of subsystems of a multibody system (i.e., systems of differential-algebraic equations of motion of the subsystems when there are no constraints among them) and some description of the constraints among these subsystems (i.e., holonomic or non-holonomic constraint equations) to obtain the equations of motion of the whole system (satsifying all the existing physical constraints).

Consider a mechanical system $\mathscr{M}$ consisting of a finite set of constrained subsystems generally denoted by $\mathcal{S}_n$.

Define $\boldsymbol{q}_n^{\langle 0 \rangle}$ as the column-matrix of 0-th order generalized variables of $\mathcal{S}_n$ (which also can be called generalized coordinates of $\mathcal{S}_n$); $\boldsymbol{q}_n^{\langle 0 \rangle}$ represents a set of variables that is enought to parametrize the description of every configuration of this subsystem. That is, all positions and orientations of $\mathcal{S}_n$ when it is not constrained to any other subsystem, can be described as functions of $\boldsymbol{q}_n^{\langle 0 \rangle}$ and of geometrical of this subsystem. Analogously, define $\boldsymbol{q}_n^{\langle 1 \rangle}$ as the column-matrix of 1-st order generalized variables of $\mathcal{S}_n$ (which also can be called quasi-velocities of $\mathcal{S}_n$); $\boldsymbol{q}_n^{\langle 1 \rangle}$ represents a set of variables that is enough to parametrize, along with $\boldsymbol{q}_n^{\langle 0 \rangle}$, the description of any state of $\mathcal{S}_n$. All components velocities, angular velocities, linear and angular momenta of $\mathscr{S}_n$ as well as an expression for the kinetic energy of this subsystem when it is not constrained to any other subsystem can be described as functions of $\boldsymbol{q}^{\langle 0\rangle}$ and $\boldsymbol{q}^{\langle 1\rangle}$. Actually, $\boldsymbol{q}^{\langle 1\rangle}$ can be interpreted as a set of variables that replace $\dot{\boldsymbol{q}}_n^{\langle 0\rangle}$ in the description of any state of $\mathscr{S}_n$.

Generally, α-th order generalized variables ($\boldsymbol{q}_n^{\langle\alpha\rangle}$) can be similarly defined as being a set of variables that replace the time derivatives of $(\alpha-1)$-th order generalized variables ($\dot{\boldsymbol{q}}_n^{\langle\alpha-1\rangle}$) in the parametric description of some motion variable. Define also $\boldsymbol{q}_n^{\langle\!\langle\alpha\rangle\!\rangle}$ as the column-matrix constituted by all generalized variables of $\mathscr{S}_n$ up to α-th order ($\boldsymbol{q}_n^{\langle 0\rangle},\ldots,\boldsymbol{q}_n^{\langle\alpha\rangle}$).

Define also the column-matrix $\boldsymbol{u}_n$ consisting of some control inputs or external disturbances that influence on the components of active forces and torques of $\mathscr{S}_n$. Consider that the mathematical model of $\mathscr{S}_n$ is already known and given by the following system of equations:

$$\begin{cases} \dot{\boldsymbol{q}}_n^{\langle\kappa\rangle} = \underline{\dot{\boldsymbol{q}}}_n^{\langle\kappa\rangle}\big(t, \boldsymbol{q}_n^{\langle\!\langle\kappa+1\rangle\!\rangle}\big) \qquad \text{for} \qquad 0 \le \kappa \le \sigma-1 \\ \bar{\boldsymbol{q}}_n^{\langle\sigma\rangle} = \tilde{\boldsymbol{A}}_n\big(t, \boldsymbol{q}_n^{\langle\!\langle\sigma-1\rangle\!\rangle}\big)\, \boldsymbol{q}_n^{\langle\sigma\rangle} + \tilde{\boldsymbol{b}}_n^{\langle\sigma-1\rangle}\big(t, \boldsymbol{q}_n^{\langle\!\langle\sigma-1\rangle\!\rangle}\big) = \boldsymbol{0} \\ \bar{\boldsymbol{d}}_n^{\langle\sigma\rangle}(t, \boldsymbol{q}_n^{\langle\!\langle\sigma\rangle\!\rangle}, \boldsymbol{u}_n) = \boldsymbol{0} \end{cases} \tag{1}$$

Define $\boldsymbol{q}^{\langle\alpha\rangle}$ and $\boldsymbol{q}^{\langle\!\langle\alpha\rangle\!\rangle}$ as the block-column-matrices constituted respectively by the $\boldsymbol{q}_n^{\langle\alpha\rangle}$ and $\boldsymbol{q}_n^{\langle\!\langle\alpha\rangle\!\rangle}$ of all the subsystems $\mathscr{S}_n$. Suppose that all the constraints among the subsystems can be described by equations of the form:

$$\bar{\bar{\boldsymbol{q}}}^{\langle\sigma\rangle} = \sum_n \tilde{\tilde{\boldsymbol{A}}}_n(t, \boldsymbol{q}^{\langle\!\langle\sigma-1\rangle\!\rangle})\, \boldsymbol{q}_n^{\langle\sigma\rangle} + \tilde{\tilde{\boldsymbol{b}}}^{\langle\sigma-1\rangle}(t, \boldsymbol{q}^{\langle\!\langle\sigma-1\rangle\!\rangle}) = \boldsymbol{0} \tag{2}$$

Suppose without loss of generality that the subsystems $\mathscr{S}_n$ of $\mathscr{M}$ are indexed by consecutive positive integers, i.e., $n \in \{1, 2, \ldots, \nu_{\mathscr{S}}\}$. In this case the jacobian of the constraint equations that most be satisfied in order to a motion be compatible with both internal constraints of the subsystems and external constraints among subsystems is given by:

$$\boldsymbol{A} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \ldots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \ldots & \tilde{\boldsymbol{A}}_{\nu_{\mathscr{S}}} \\ \tilde{\tilde{\boldsymbol{A}}}_1 & \ldots & \tilde{\tilde{\boldsymbol{A}}}_{\nu_{\mathscr{S}}} \end{bmatrix}$$

Let $\tilde{\boldsymbol{C}}_n$ denote an orthogonal complement of $\tilde{\boldsymbol{A}}_n$. Depending on the methodology used to derive the mathematical model of $\mathscr{S}_n$, some expression for $\tilde{\boldsymbol{C}}_n$ may already be known. Define the matrix $\tilde{\tilde{\boldsymbol{A}}}$ by the expression:

$$\tilde{\tilde{\boldsymbol{A}}} = \left[\begin{array}{cccc} \tilde{\tilde{\boldsymbol{A}}}_1 \tilde{\boldsymbol{C}}_1 & \tilde{\tilde{\boldsymbol{A}}}_2 \tilde{\boldsymbol{C}}_2 & \ldots & \tilde{\tilde{\boldsymbol{A}}}_{\nu_{\mathscr{S}}} \tilde{\boldsymbol{C}}_{\nu_{\mathscr{S}}} \end{array} \right]$$

Define $\tilde{\boldsymbol{d}}^{\langle\sigma\rangle}$ as the block-column-matrix constituted by the $\tilde{\boldsymbol{d}}_n^{\langle\sigma\rangle}$ of all the subsystems $\mathscr{S}_n$ and let $\tilde{\tilde{\boldsymbol{C}}}$ be an orthogonal complement of $\tilde{\tilde{\boldsymbol{A}}}$. It can be stated that, the equations of motion of system $\mathscr{M}$, compatible with all its physical constraints, are given by [?]:

$$\begin{cases} \dot{\boldsymbol{q}}_n^{\langle\kappa\rangle} = \underline{\dot{\boldsymbol{q}}}_n^{\langle\kappa\rangle}\left(t, \boldsymbol{q}_n^{\langle\langle\kappa+1\rangle\rangle}\right), \text{ for } 0 \leq \kappa \leq \sigma - 1, \forall\, n \\ \bar{\boldsymbol{q}}_n^{\langle\sigma\rangle} = \tilde{\boldsymbol{A}}_n\, \boldsymbol{q}_n^{\langle\sigma\rangle} + \tilde{\boldsymbol{b}}_n^{\langle\sigma-1\rangle} = \boldsymbol{0}, \forall\, n \\ \bar{\bar{\boldsymbol{q}}}^{\langle\sigma\rangle} = \sum_n \tilde{\tilde{\boldsymbol{A}}}_n\, \boldsymbol{q}_n^{\langle\sigma\rangle} + \tilde{\tilde{\boldsymbol{b}}}^{\langle\sigma-1\rangle} = \boldsymbol{0} \\ \bar{\bar{\boldsymbol{d}}}^{\langle\sigma\rangle} = \tilde{\tilde{\boldsymbol{C}}}^{\mathsf{T}}\, \bar{\boldsymbol{d}}^{\langle\sigma\rangle} = \boldsymbol{0} \end{cases} \tag{3}$$

Package MoSs consists of functions developed in Wolfram Mathematica 10.0 that enable the implementation of the algorithm for obtaining the system of equations (3) from already known expressions for (1) and (2).

Commented example

Spherical pendulum modelling

Figure 1 illustrates the use of the package MoSs for obtaining a mathematical model of a spherical pendulum.

Basically, a spherical pendulum $\mathscr{S}$ can be conceived as a rigid body $\mathcal{B}$ whose centre of mass is constrained to move in a spherical surface. Consider that the centre of the sphere remains fixed with respect to an inertial reference frame $\mathcal{N}$. In order to use the modular modelling algorithm for this system, consider it as composed by two subsystems: $\mathscr{N}$ consisting of the inertial reference frame $\mathcal{N}$ and $\mathscr{B}$ consisting of the rigid body $\mathcal{B}$.

Define a coordinate system fixed to $\mathcal{N}$ such that its origin is the centre of the spherical surface and the z-axis is vertical pointing upwards. Let $(p_{\mathcal{P},\mathrm{x}}, p_{\mathcal{P},\mathrm{y}}, p_{\mathcal{P},\mathrm{z}})$ denote the coordinates of the centre of mass of $\mathcal{B}$ in this coordinate system. The (external) constraint between systems $\mathscr{N}$ and $\mathscr{S}$ can be expressed by the following equation:

$$p_{\mathcal{B},\mathrm{x}}^2 + p_{\mathcal{B},\mathrm{y}}^2 + p_{\mathcal{B},\mathrm{z}}^2 - \bar{a}^2 = 0$$

Therefore, the mathematical model of $\mathscr{B}$ can be given by:

```
NewtonEuler["ℬ", "Position␣Only", "-z"]
```

and $\mathscr{S}$ can be defined as a system composed by the subsystems $\mathscr{N}$ (included by default) and $\mathscr{B}$ whose external constraints are defined by the following order 0 invariant:

```
"*q"["0"]->{Subscript[p, "ℬ", "x"][t]^2 + Subscript[p, "ℬ", "y"][t]^2
  + Subscript[p, "ℬ", "z"][t]^2 - ā^2}
```

This is the strategy for modelling $\mathscr{P}$ presented in Figure 1.

```
Sy11 = MoSs["𝒮", {NewtonEuler["ℬ", "Position", "-z"]}];
Sy11["*q"["0"]] = {p"ℬ","x"[t]^2 + p"ℬ","y"[t]^2 + p"ℬ","z"[t]^2 - ā^2};
Sy11["Replacement Rules"] = {
   p"ℬ","x"[t]^2 + p"ℬ","y"[t]^2 + p"ℬ","z"[t]^2 → ā^2
  };
Sy11["q#"["0"]] = {p"ℬ","x"[t], p"ℬ","y"[t]};
Sy11["q#"["1"]] = {v"ℬ","x"[t], v"ℬ","y"[t], ω"ℬ","x"[t], ω"ℬ","y"[t], ω"ℬ","z"[t]};
Sy11["Explicit EOM"] = "Yes";
Sy11 = MoSs @ Sy11;

Sy11["_f"] // TableForm
Sy11["S"] // SMatrixForm
```

$$\dot{v}_{\mathscr{B},x}[t] \to \frac{p_{\mathscr{B},x}[t]\left(g\, p_{\mathscr{B},z}[t]-v_{\mathscr{B},x}[t]^2-v_{\mathscr{B},y}[t]^2-v_{\mathscr{B},z}[t]^2\right)}{\bar{a}^2}$$

$$\dot{v}_{\mathscr{B},y}[t] \to \frac{p_{\mathscr{B},y}[t]\left(g\, p_{\mathscr{B},z}[t]-v_{\mathscr{B},x}[t]^2-v_{\mathscr{B},y}[t]^2-v_{\mathscr{B},z}[t]^2\right)}{\bar{a}^2}$$

$$\dot{v}_{\mathscr{B},z}[t] \to -\frac{g\left(p_{\mathscr{B},x}[t]^2+p_{\mathscr{B},y}[t]^2\right)+p_{\mathscr{B},z}[t]\left(v_{\mathscr{B},x}[t]^2+v_{\mathscr{B},y}[t]^2+v_{\mathscr{B},z}[t]^2\right)}{\bar{a}^2}$$

$$\dot{\omega}_{\mathscr{B},x}[t] \to \frac{\left(I_{\mathscr{B},y}-I_{\mathscr{B},z}\right)\omega_{\mathscr{B},y}[t]\,\omega_{\mathscr{B},z}[t]}{I_{\mathscr{B},x}}$$

$$\dot{\omega}_{\mathscr{B},y}[t] \to \frac{\left(-I_{\mathscr{B},x}+I_{\mathscr{B},z}\right)\omega_{\mathscr{B},x}[t]\,\omega_{\mathscr{B},z}[t]}{I_{\mathscr{B},y}}$$

$$\dot{\omega}_{\mathscr{B},z}[t] \to \frac{\left(I_{\mathscr{B},x}-I_{\mathscr{B},y}\right)\omega_{\mathscr{B},x}[t]\,\omega_{\mathscr{B},y}[t]}{I_{\mathscr{B},z}}$$

$$\left(\begin{array}{c|ccccc}
 & v_{\mathscr{B},x} & v_{\mathscr{B},y} & \omega_{\mathscr{B},x} & \omega_{\mathscr{B},y} & \omega_{\mathscr{B},z} \\
\hline
v_{\mathscr{B},x} & 1 & 0 & 0 & 0 & 0 \\
v_{\mathscr{B},y} & 0 & 1 & 0 & 0 & 0 \\
v_{\mathscr{B},z} & -\frac{p_{\mathscr{B},x}[t]}{p_{\mathscr{B},z}[t]} & -\frac{p_{\mathscr{B},y}[t]}{p_{\mathscr{B},z}[t]} & 0 & 0 & 0 \\
\omega_{\mathscr{B},x} & 0 & 0 & 1 & 0 & 0 \\
\omega_{\mathscr{B},y} & 0 & 0 & 0 & 1 & 0 \\
\omega_{\mathscr{B},z} & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

Figure 1: Model of a spherical pendulum obtained using MoSs package

However, noticing that the term $v_{\mathcal{B},x}^2 + v_{\mathcal{B},y}^2 + v_{\mathcal{B},z}^2$ appears in all the dynamic equations related to the translational motion, a new quasi-velocity $k_{\mathcal{B}}$ can be defined, such that:

$$v_{\mathcal{B},x}^2 + v_{\mathcal{B},y}^2 + v_{\mathcal{B},z}^2 - k_{\mathcal{B}} = 0$$

This is done in the example shown in Figure 2.

```
Sy21 = MoSs["S", {NewtonEuler["B", "Position", "-z"]}];
Sy21["q"]["1"]] = {k"B"[t]};
Sy21["*q"]["0"]] = {p"B","x"[t]^2 + p"B","y"[t]^2 + p"B","z"[t]^2 - ā^2};
Sy21["*q"]["1"]] = {v"B","x"[t]^2 + v"B","y"[t]^2 + v"B","z"[t]^2 - k"B"[t]};
Sy21["Replacement Rules"] = {
    p"B","x"[t]^2 + p"B","y"[t]^2 + p"B","z"[t]^2 → ā^2,
    p"B","x"[t] p"B","x"'[t] + p"B","y"[t] p"B","y"'[t] + p"B","z"[t] p"B","z"'[t] → 0,
    p"B","x"[t]^2 + p"B","y"[t]^2 → ā^2 - p"B","z"[t]^2,
    p"B","x"[t] p"B","x"'[t] + p"B","y"[t] p"B","y"'[t] → -p"B","z"[t] p"B","z"'[t],
    v"B","x"[t]^2 + v"B","y"[t]^2 + v"B","z"[t]^2 → k"B"[t]
  };
Sy21["Explicit EOM"] = "Yes";
Sy21 = MoSs @ Sy21;

Sy21["_f"] // TableForm
Sy21["S"] // SMatrixForm
```

$$\dot{k}_{\mathcal{B}}[t] \to -2\,\bar{g}\,v_{\mathcal{B},z}[t]$$

$$\dot{v}_{\mathcal{B},x}[t] \to \frac{p_{\mathcal{B},x}[t]\left(-k_{\mathcal{B}}[t]+g\,p_{\mathcal{B},z}[t]\right)}{\bar{a}^2}$$

$$\dot{v}_{\mathcal{B},y}[t] \to \frac{p_{\mathcal{B},y}[t]\left(-k_{\mathcal{B}}[t]+g\,p_{\mathcal{B},z}[t]\right)}{\bar{a}^2}$$

$$\dot{v}_{\mathcal{B},z}[t] \to -\frac{\bar{a}^2\,g+p_{\mathcal{B},z}[t]\left(k_{\mathcal{B}}[t]-g\,p_{\mathcal{B},z}[t]\right)}{\bar{a}^2}$$

$$\dot{\omega}_{\mathcal{B},x}[t] \to \frac{\left(I_{\mathcal{B},y}-I_{\mathcal{B},z}\right)\omega_{\mathcal{B},y}[t]\,\omega_{\mathcal{B},z}[t]}{I_{\mathcal{B},x}}$$

$$\dot{\omega}_{\mathcal{B},y}[t] \to \frac{\left(-I_{\mathcal{B},x}+I_{\mathcal{B},z}\right)\omega_{\mathcal{B},x}[t]\,\omega_{\mathcal{B},z}[t]}{I_{\mathcal{B},y}}$$

$$\dot{\omega}_{\mathcal{B},z}[t] \to \frac{\left(I_{\mathcal{B},x}-I_{\mathcal{B},y}\right)\omega_{\mathcal{B},x}[t]\,\omega_{\mathcal{B},y}[t]}{I_{\mathcal{B},z}}$$

	$\tilde{q}_{\mathcal{S},1}$	$\tilde{q}_{\mathcal{S},2}$	$\tilde{q}_{\mathcal{S},3}$	$\tilde{q}_{\mathcal{S},4}$	$\tilde{q}_{\mathcal{S},5}$
$k_{\mathcal{B}}$	0	0	0	$\frac{2\left(-p_{\mathcal{B},z}[t]\,v_{\mathcal{B},x}[t]+p_{\mathcal{B},x}[t]\,v_{\mathcal{B},z}[t]\right)}{p_{\mathcal{B},x}[t]}$	$\frac{2\left(-p_{\mathcal{B},y}[t]\,v_{\mathcal{B},x}[t]+p_{\mathcal{B},x}[t]\,v_{\mathcal{B},y}[t]\right)}{p_{\mathcal{B},x}[t]}$
$v_{\mathcal{B},x}$	0	0	0	$-\frac{p_{\mathcal{B},z}[t]}{p_{\mathcal{B},x}[t]}$	$-\frac{p_{\mathcal{B},y}[t]}{p_{\mathcal{B},x}[t]}$
$v_{\mathcal{B},y}$	0	0	0	0	1
$v_{\mathcal{B},z}$	0	0	0	1	0
$\omega_{\mathcal{B},x}$	0	0	1	0	0
$\omega_{\mathcal{B},y}$	0	1	0	0	0
$\omega_{\mathcal{B},z}$	1	0	0	0	0

Figure 2: Model of a spherical pendulum obtained using MoSs package: definition of a new quasi-velocity

Another variant of the model can be obtained when $(p_{\mathcal{B},x}, p_{\mathcal{B},y}, p_{\mathcal{B},z})$ are parametrized in terms of spherical coordinates, leading to the most conventional version of the spherical pendulum equations of motion. This is done in the example shown in Figure 3.

```
Sy31 = MoSs["𝒮", {NewtonEuler["ℬ", "Position", "-z"]}];
Sy31["q"["0"]] = {ϕ[t], θ[t]};
Sy31["q"["1"]] = {ϕ'[t], θ'[t]};
Sy31["q#"["1"]] = {ϕ'[t], θ'[t], ω"ℬ","x"[t], ω"ℬ","y"[t], ω"ℬ","z"[t]};
Sy31["*q"["0"]] = {
   p"ℬ","x"[t] - ā Sin[θ[t]] Cos[ϕ[t]],
   p"ℬ","y"[t] - ā Sin[θ[t]] Sin[ϕ[t]],
   p"ℬ","z"[t] + ā Cos[θ[t]]};
Sy31["Explicit EOM"] = "Yes";
Sy31 = MoSs @ Sy31;

Sy31["_f"] // TableForm
Sy31["S"] // SMatrixForm
```

$$\dot{v}_{\mathcal{B},x}[t] \to -c_{\phi[t]}\, s_{\theta[t]} \left(c_{\theta[t]}\, \bar{g} + \bar{a}\left(\dot{\theta}[t]^2 + s^2_{\theta[t]}\, \dot{\phi}[t]^2\right)\right)$$

$$\dot{v}_{\mathcal{B},y}[t] \to -s_{\theta[t]}\, s_{\phi[t]} \left(c_{\theta[t]}\, \bar{g} + \bar{a}\left(\dot{\theta}[t]^2 + s^2_{\theta[t]}\, \dot{\phi}[t]^2\right)\right)$$

$$\dot{v}_{\mathcal{B},z}[t] \to -\bar{g}\, s^2_{\theta[t]} + c_{\theta[t]}\, \bar{a}\left(\dot{\theta}[t]^2 + s^2_{\theta[t]}\, \dot{\phi}[t]^2\right)$$

$$\dot{\omega}_{\mathcal{B},x}[t] \to \frac{(I_{\mathcal{B},y} - I_{\mathcal{B},z})\, \omega_{\mathcal{B},y}[t]\, \omega_{\mathcal{B},z}[t]}{I_{\mathcal{B},x}}$$

$$\dot{\omega}_{\mathcal{B},y}[t] \to \frac{(-I_{\mathcal{B},x} + I_{\mathcal{B},z})\, \omega_{\mathcal{B},x}[t]\, \omega_{\mathcal{B},z}[t]}{I_{\mathcal{B},y}}$$

$$\dot{\omega}_{\mathcal{B},z}[t] \to \frac{(I_{\mathcal{B},x} - I_{\mathcal{B},y})\, \omega_{\mathcal{B},x}[t]\, \omega_{\mathcal{B},y}[t]}{I_{\mathcal{B},z}}$$

$$\ddot{\theta}[t] \to s_{\theta[t]} \left(-\frac{\bar{g}}{\bar{a}} + c_{\theta[t]}\, \dot{\phi}[t]^2\right)$$

$$\ddot{\phi}[t] \to -\frac{2\, c_{\theta[t]}\, \dot{\theta}[t]\, \dot{\phi}[t]}{s_{\theta[t]}}$$

	$\omega_{\mathcal{B},x}$	$\omega_{\mathcal{B},y}$	$\omega_{\mathcal{B},z}$	$\dot{\theta}$	$\dot{\phi}$
$v_{\mathcal{B},x}$	0	0	0	$c_{\theta[t]}\, c_{\phi[t]}\, \bar{a}$	$-\bar{a}\, s_{\theta[t]}\, s_{\phi[t]}$
$v_{\mathcal{B},y}$	0	0	0	$c_{\theta[t]}\, \bar{a}\, s_{\phi[t]}$	$c_{\phi[t]}\, \bar{a}\, s_{\theta[t]}$
$v_{\mathcal{B},z}$	0	0	0	$\bar{a}\, s_{\theta[t]}$	0
$\omega_{\mathcal{B},x}$	1	0	0	0	0
$\omega_{\mathcal{B},y}$	0	1	0	0	0
$\omega_{\mathcal{B},z}$	0	0	1	0	0
$\dot{\theta}$	0	0	0	1	0
$\dot{\phi}$	0	0	0	0	1

Figure 3: Model of a spherical pendulum obtained using MoSs package: model in spherical coordinates

Double pendulum modelling

The example shown in Figure 4 explores an alternative use of the syntax of the function `MoSs` and the to model a planar double pendulum. Also linearized equations of motion are obtained by the use of the function `LinearizeSystem`.

The strategy consists of defining a multibody system $\mathscr{D}$ consisting of two subsystems, 1 and 2, each one consisting of a free rigid body in a gravitational field (that has "−z" direction). New angular coordinates θ_1 and θ_2, as well as the quasi-velocities $\dot{\theta}_1$ and $\dot{\theta}_2$, are defined to parametrize the description of the position coordinates of the centres of mass of each of these rigid bodies. Such parametrical descriptions lead to order 0 invariants. Finally the reference state of the system is defined and the linearization procedure can be applied, leading to the linearized explicit equations of motion shown in Figure 4.

```
SyDp = MoSs[{"𝒟", "Double Pendulum"}, {NewtonEuler[1, "Position", "-z"], NewtonEuler[2, "Position", "-z"]}];
SyDp["q"["0"]] = SyDp["q#"["0"]] = {θ1[t], θ2[t]};
SyDp["q"["1"]] = SyDp["q#"["1"]] = {θ1'[t], θ2'[t], ω1,"x"[t], ω1,"y"[t], ω1,"z"[t], ω2,"x"[t], ω2,"y"[t], ω2,"z"[t]};
SyDp["*q"["0"]] = {
    p1,"x"[t] - ā1 Sin[θ1[t]],
    p1,"y"[t],
    p1,"z"[t] + ā1 Cos[θ1[t]],
    p2,"x"[t] - ā1 Sin[θ1[t]] - ā2 Sin[θ2[t]],
    p2,"y"[t],
    p2,"z"[t] + ā1 Cos[θ1[t]] + ā2 Cos[θ2[t]]
   };
SyDp[1]["Reference Motion"] = {p1,"z"[t] → -ā1};
SyDp[2]["Reference Motion"] = {p2,"z"[t] → -ā1 - ā2};
SyDp["Explicit Linearized EOM"] = "Yes";
SyDp = MoSs @ SyDp;
LSyDp = LinearizeSystem @ SyDp;

LSyDp["_f"] // TableForm
```

$$\dot{v}_{1,x}[t] \to \frac{g\left(-(\bar{m}_1+\bar{m}_2)\,\theta_1[t]+\bar{m}_2\,\theta_2[t]\right)}{\bar{m}_1}$$

$$\dot{v}_{1,y}[t] \to 0$$

$$\dot{v}_{1,z}[t] \to 0$$

$$\dot{v}_{2,x}[t] \to -\bar{g}\,\theta_2[t]$$

$$\dot{v}_{2,y}[t] \to 0$$

$$\dot{v}_{2,z}[t] \to 0$$

$$\dot{\omega}_{1,x}[t] \to 0$$

$$\dot{\omega}_{1,y}[t] \to 0$$

$$\dot{\omega}_{1,z}[t] \to 0$$

$$\dot{\omega}_{2,x}[t] \to 0$$

$$\dot{\omega}_{2,y}[t] \to 0$$

$$\dot{\omega}_{2,z}[t] \to 0$$

$$\ddot{\theta}_1[t] \to \frac{g\left(-(\bar{m}_1+\bar{m}_2)\,\theta_1[t]+\bar{m}_2\,\theta_2[t]\right)}{\bar{a}_1\,\bar{m}_1}$$

$$\ddot{\theta}_2[t] \to \frac{g\,(\bar{m}_1+\bar{m}_2)\,(\theta_1[t]-\theta_2[t])}{\bar{a}_2\,\bar{m}_1}$$

Figure 4: Model of a double pendulum obtained using MoSs package

1 Modular Modelling

```
MoSs[xSystem_, xSubSystems_List:{}]:=
 Module[{xIn, xOut, xRules, xKeys, xA, xTimer},
  xTimer = AbsoluteTime[];
  xIn = xSystem;
  xOut = If[AssociationQ[xIn], xIn, Association[]];
  Quiet @ (
   xOut["System Label"] = xIn /. {
    xX_List /; Length[xX] >= 1 -> xX[[1]],
    xX_Association -> xX["System Label"]
    };
   xOut["Subsystems Labels"] = xIn /. {
    xX_Association /; KeyExistsQ[xX,"Subsystems Labels"] -> xX["Subsystems Labels"],
    xX_ -> {}
    };
   xOut["Description"] = xIn /. {
    xX_List /; And[Length[xX] >= 2,StringQ[xX[[2]]]] -> xX[[2]],
    xX_Association /; And[KeyExistsQ[xX,"Description"], StringQ[xX["Description"]]] -> xX["Description"],
    xX_ -> ""
    };
   xOut["Naming Rules"] = xIn /. {
    xX_List /; Length[xX] >= 3 -> xX[[3]],
    xX_Association /; KeyExistsQ[xX, "Naming Rules"] -> xX["Naming Rules"],
    xX_ -> {}
    };
   xOut["Replacement Rules"] = xIn /. {
    xX_List /; Length[xX] >= 4 -> xX[[4]],
    xX_Association /; KeyExistsQ[xX, "Replacement Rules"] -> xX["Replacement Rules"],
    xX_ -> {}
    };
```

```
    );

  Quiet @ (
    xIn = (xSubSystems[[#]]) /. {
      xX_List /; AssociationQ[xX[[1]]] -> xX[[1]]
      };
    xRules["Naming␣Rules"] = Join[
      (xSubSystems[[#]]) /. {
      xX_List /; And[Length[xX] >= 2, Or[ListQ[xX[[2]]], AssociationQ[xX[[2]]]]] -> Normal @ (xX[[2]]),
      xX_ -> {}
      },
      xOut["Naming␣Rules"]
      ];
    xRules["Replacement␣Rules"] = Join[
      (xSubSystems[[#]]) /. {
      xX_List /; And[Length[xX] >= 3,Or[ListQ[xX[[3]]],AssociationQ[xX[[3]]]]] -> Normal @ (xX[[3]]),
      xX_ -> {}
      },
      xOut["Replacement␣Rules"]
      ];
    xIn = SRename[xIn, xRules["Naming␣Rules"], xRules["Replacement␣Rules"]];
    xOut["Subsystems␣Labels"] = Union[xOut["Subsystems␣Labels"], {xIn["System␣Label"]}];
    xOut[xIn["System␣Label"]] = xIn;
    xOut["Replacement␣Rules"] = Union[xOut["Replacement␣Rules"], xRules["Replacement␣Rules"]];
    )& /@ Range @ (Length @ xSubSystems);

  xIn = xOut;
  xOut["q:Order"] = If[KeyExistsQ[xIn, "q:Order"],
    xIn["q:Order"],
```

```
    Max @ (((xIn[#]["q:Order"])& /@ xIn["Subsystems␣Labels"]) //. Missing[xX__] -> {})
    ];
  (If[xIn[#]["q:Order"] < xOut["q:Order"],
    xIn[#]["q:Order"] = xOut["q:Order"]]; xOut[#] = MoSs @ xIn[#];
    )& /@ xIn["Subsystems␣Labels"];
  xOut["Reference␣Motion"] = Union @@ {
    (xIn["Reference␣Motion"] //. Missing[xX__] -> {}),
    Union @@ ((xOut[#]["Reference␣Motion"] //. Missing[xX__] -> {})& /@
      xIn["Subsystems␣Labels"])
    };
  If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["‘‘:‘‘:Subsystems:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
      xOut["System␣Label"]]]
    ];

  xKeys = Part[#, 1]& /@ Union @ (Flatten @ {
    (Select[Keys @ xIn, Part[#, 0] === "q"&]),
    (Select[Keys @ xIn[#], Part[#, 0] === "q"&])& /@ xIn["Subsystems␣Labels"]
    });
  Function[{xKey},
    xIn["q+"[xKey]] = Complement[
      xIn["q"[xKey]] //. Missing[xX__] -> {},
      Union @@ (Function[{xSub}, xIn[xSub]["q"[xKey]] //. Missing[xX__] -> {}] /@ xIn["Subsystems␣Labels"])
      ];
    ] /@ xKeys;
  xKeys = Part[#, 1]& /@ Union @ (Flatten @ {
    Select[Keys @ xIn, And[Part[#, 0] === "q+", Not @ (xIn[#] === {})]&]
    });
  If[xKeys === {},
    (*-TRUE-*)
```

```
    (xOut["q+"[ToString @ #]] = {})& /@ Range[0, Max[2, xOut["q:Order"]]],
    (*-FALSE-*)
    (xOut["q+"[#]] = xIn["q+"[#]])& /@ xKeys;
    xOut["q:Def:Order"] = If[KeyExistsQ[xIn,"q:Def:Order"],
      xIn["q:Def:Order"],
      Max @ ToExpression @ Flatten @ (StringSplit[#, {":", "|"}]& /@ xKeys)
      ];
    (xOut["q+"[ToString @ #]] = D[
      xOut["q+"[ToString @ xOut["q:Def:Order"]]],
      {t, (# - xOut["q:Def:Order"])}
      ])& /@ Range[xOut["q:Def:Order"] + 1, Max[2, xOut["q:Order"]]];
    ];
  (xOut["q"[#]] = Union[
    xOut["q+"[#]] //. Missing[xX__] -> {},
    Union @@ (Function[{xSub}, xOut[xSub]["q"[#]] //. Missing[xX__] -> {}] /@ xOut["Subsystems Labels"])
    ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
  xKeys = Union[
    ReplaceRepeated[#, {{xX_,xY_} :> (ToString[xX] <> "|" <> ToString[xY])}]& @
      (Select[Flatten[#, 1], (Part[#, 1] > Part[#, 2])&]& @ (Outer[List, #, #]))
    ]& @ Range[0, Max[2, xOut["q:Order"]]];
  (xOut["q"[#]] = D[
    xOut["q"[Part[#, 2]]] //. Missing[xX__] -> {},
    {t, ((ToExpression @ Part[#, 1]) - (ToExpression @ Part[#, 2]))}
    ]& @ StringSplit[#, {":", "|"}];
  xOut["q+"[#]] = D[
    xOut["q+"[Part[#, 2]]] //. Missing[xX__] -> {},
    {t, ((ToExpression @ Part[#, 1]) - (ToExpression @ Part[#, 2]))}
    ]& @ StringSplit[#, {":", "|"}];
    )& /@ xKeys;
  If[xOut["Debug Mode"] === "On",
```

```
    Print[StringForm["``:``:q:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
      xOut["System␣Label"]]]
    ];

  xKeys = Part[#,1]& /@ Union @ (Flatten @ {
    (Select[Keys @ xIn, Part[#, 0] === "*c"&]),
    (Select[Keys @ xIn[#], Part[#, 0] === "*c"&])& /@ xIn["Subsystems␣Labels"]
    });
  Function[xKey,
    xOut["*c"[xKey]] = (Union @ (Flatten @ ({
      xIn["*c"[xKey]],
      Function[{xSub}, xIn[xSub]["*c"[xKey]]] /@ xIn["Subsystems␣Labels"]} //. Missing[xX__] -> {})))
    ]/@ xKeys;
  (xOut["*c"[#]] = {})& /@ Complement[ToString /@ Range[0, xOut["q:Order"]], xKeys];

  xRules = Union @ (Flatten @
    ({(* xIn["_c"], *) Function[{xSub}, xIn[xSub]["_c"]] /@ xIn["Subsystems␣Labels"]} //. Missing[xX__] -> {})
    );
  ( xOut["_q"[(ToString @ #)<> "|" <> (ToString @ (# - 1))]] = Union @ (Flatten @ ({
      xIn["_q"[(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
      Function[{xSub}, xIn[xSub]["_q"[(ToString @ #)<> "|"<>(ToString @ (#-1))]]] /@ xIn["Subsystems␣Labels"]
      } //. Missing[xX__] -> {})
      );
    If[Not @ (Complement[
      xOut["q"[(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
      xOut["q"[ToString @ #]], First /@ xRules,
      First /@ xOut["_q"[(ToString @ #) <> "|" <> (ToString @ (# - 1))]]
```

```
      ] === {}),
      xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (#-1))]] = Union @@ {
        xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (#-1))]],
        (Simplify @ Flatten @ (Quiet @ Solve[
          (# == 0)& /@ (RedundantElim @((RedundantElim @( Union @@ {
            D[xOut["*c"][ToString @ (#-1)]],t],
            xOut["*c"][ToString @ #]]
            } //. xRules)) //. xRules)),
          Complement[
            xOut["q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
            xOut["q"][ToString @ #]], First/@ xRules,
            First /@ xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (#-1))]]
            ]
          ])) //. xRules
        }
      ];
    xRules = Union @@ {
      xRules,
      xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]]
      };
    )& /@ Range[1, xOut["q:Order"]];
  xOut["_c"] = Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @
    (Union[xRules, xIn["_c"] //. Missing[xX__] -> {}]);
  xOut["Replacement Rules"] = Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @
    (Union[#, # /. xOut["_c"]]& @ xIn["Replacement Rules"]);
  If[xOut["Debug Mode"] === "On",
    Print[StringForm["``:``:_q:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
      xOut["System Label"]]]
    ];

  xA = {};
```

```
    xKeys = (ToString @ xOut["q:Order"]);
    If[Not @ (xOut["q+"[xKeys]] === {}),
      If[KeyExistsQ[xIn, "f"],
        AppendTo[xA, SPart[xIn["f"], xOut["q+"[xKeys]]]],
        AppendTo[xA, SPart[0, xOut["q+"[xKeys]]]]
        ]
      ];
    If[KeyExistsQ[xOut[#], "*f"],
      AppendTo[xA, xOut[#]["*f"]]
      ]& /@ xOut["Subsystems␣Labels"];
    xOut["*f"] = xOut["f"] = SAssemble @@ (RedundantElim @ xA);
    If[xOut["Debug␣Mode"] === "On",
      Print[StringForm["``:``:f:OK",
        NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
        xOut["System␣Label"]]]
      ];

    xKeys = Part[#, 1]& /@ (Select[Keys @ xIn, Or[Part[#, 0] === "*q", Part[#, 0] === "*q+"]&]);
    If[xKeys === {},
      (*-TRUE-*)
      (xOut["*q+"[#]] = {})& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]),
      (*-FALSE-*)
      xOut["*q:Order"] = If[KeyExistsQ[xIn, "*q:Order"], xIn["*q:Order"], xOut["q:Order"]];
      (xOut["*q+"[#]] = Complement[
        Union[xIn["*q"[#]] //. Missing[xX__] -> {}, xIn["*q+"[#]] //. Missing[xX__] -> {}],
        Union @@ (Function[{xSub}, xIn[xSub]["*q"[#]] //. Missing[xX__] -> {}] /@ xIn["Subsystems␣Labels"])
        ])& /@ xKeys;
      If[Not @ (xOut["*q?"] === "No"),
        (*-TRUE-*)
        (xOut["*q+"[#]] = Union[
```

```
     xOut["*q+"[#]],
     RedundantElim @ ((xOut["*c"[#]] //. Missing[xX__] -> {}) //. xOut["_c"])
     ])& /@ xKeys;
    (xOut["*q+"[ToString @ #]] = (RedundantElim @ ((Union @@ {
     D[xOut["*q+"[ToString @ (# - 1)]] //. Missing[xX__] -> {}, t],
     xOut["*q+"[ToString @ #]] //. Missing[xX__] -> {}
     }) //. xOut["_c"]))
     )& /@ Range[1, Max[2, xOut["q:Order"]]],
    (*-FALSE-*)
    (xOut["*q+"[ToString @ #]] = ((Union @@ {
     D[xOut["*q+"[ToString @ (# - 1)]] //. Missing[xX__] -> {}, t]
     }) //. xOut["_c"])
     )& /@ Range[xOut["*q:Order"] + 1, 2]
    ]
   ];
  (xOut["*q"[#]] = Union[
   xOut["*q+"[#]] //. Missing[xX__] -> {},
   Union @@ (Function[{xSub}, xIn[xSub]["*q"[#]] //. Missing[xX__] -> {}] /@ xIn["Subsystems␣Labels"])
   ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
  If[xOut["Debug␣Mode"] === "On",
   Print[StringForm["``:``:*q:OK",
    NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
    xOut["System␣Label"]]]
   ];

  xA = {};
  xKeys = (ToString @ xOut["q:Order"]);
  If[KeyExistsQ[xOut, "*q+"[xKeys]],
   If[And[KeyExistsQ[xOut, "q+"[xKeys]], Not @ (xOut["q+"[xKeys]] === {})],
    AppendTo[xA, Jacobi[xOut["*q+"[xKeys]], xOut["q+"[xKeys]]]]
    ];
```

```
    If[And[KeyExistsQ[xOut[#], "q"[xKeys]], Not @ (xOut[#]["q"[xKeys]] === {})],
      If[KeyExistsQ[xOut[#], "S"],
        AppendTo[xA, Jacobi[xOut["*q+"[xKeys]], xOut[#]["q"[xKeys]]] ~SDot~ xOut[#]["S"]],
        AppendTo[xA, Jacobi[xOut["*q+"[xKeys]], xOut[#]["q"[xKeys]]]]
        ]
      ]& /@ xIn["Subsystems␣Labels"];
    xOut["B"] = SAssemble @@ xA
    ];
  If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["``:``:B:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
      xOut["System␣Label"]]]
    ];

  If[Not @ (xOut["C?"] === "No"),
    If[xOut["B"]["Matrix"] === {},
      xOut["C"] = SAssemble[1, xOut["B"]["Column␣Labels"]],
      If[KeyExistsQ[xOut, "q#"[xKeys]],
        xOut["C"] = OrthogonalComplement[xOut["B"], xOut["q#"[xKeys]]],
        xOut["C"] = OrthogonalComplement[xOut["B"], ToString @ xOut["System␣Label"]]
        ]
      ];
    If[Not @ (xOut["S?"] === "No"),
      xA = {};
      If[KeyExistsQ[xOut[#], "S"], AppendTo[xA, xOut[#]["S"]]]& /@ xIn["Subsystems␣Labels"];
      If[xA === {},
        xOut["S"] = xOut["C"],
        If[Not @ (xOut["q+"[xKeys]] === {}),
          AppendTo[xA, SAssemble[1, xOut["q+"[xKeys]]]]
          ];
        xOut["S"] = (SAssemble @@ xA) ~SDot~ xOut["C"]
```

```
          ];
        ];
      If[xOut["Debug␣Mode"] === "On",
        Print[StringForm["``:``:C:OK",
          NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
          xOut["System␣Label"]]]
        ];
      ];

    If[And[KeyExistsQ[xOut, "f"], KeyExistsQ[xOut, "C"]],
      xOut["*f"] = STranspose[xOut["C"]] ~SDot~ xOut["f"];
      If[xOut["Explicit␣EOM"] === "Yes",
        (xOut["_f"] = SReplaceFullSimplify[
          Solve[(# == 0)& /@ Flatten @ (Union @@ {SAssemble[xOut["*f"]]["Matrix"], xOut["*q"][#]}),
            xOut["q"[#]]],
          xOut["Replacement␣Rules"]
          ])& @ (ToString @ (Max[2, xOut["q:Order"]]));
        If[xOut["Debug␣Mode"] === "On",
          Print[StringForm["``:``:_f:OK",
          NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
          xOut["System␣Label"]]]
          ];
        ];
      ];
    If[xOut["Debug␣Mode"] === "On",
      Print[StringForm["``:``:*f:OK",
        NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
        xOut["System␣Label"]]]
      ];

    If[xOut["Timer"] === "On",
      Print[StringForm["``:``:OK",
        NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
        xOut["System␣Label"]]]
```

```
    ];
  xOut
  ]
```

MoSs is a function that implements the modular modelling algorithm. Once enough information is provided (models of subsystems and descriptions of external constraint equations), its output is an Association element representing the complete model of a multibody system (dynamic equations and ν°-th order constraint equations). Two syntaxes are admissible for this function:

- MoSs[S]
  S must be an Association element representing a multibody system. If S is already a complete model, then the output of this function will be S.

- MoSs[S,Ss]
  S can be:
  (a) An Association element representing a multibody system.
  (b) A String element representing the label of output multibody system.
  (c) Or a List element with up to 4 elements:
     i. The first element is a String element representing the label of multibody system.
     ii. The second element (optional) is a String element providing a description of the system.
     iii. The third element (optional) is a List of replacement rules for nomenclature, applicable both to the keys and values of Association elements within the scope of the function.
     iv. The fourth element (optional) is a List of replacement rules to be applicable to the values of Association elements within the scope of the function.

  Ss is a List element providing the models of the subsystems of this system. The elements of Ss can be:
  (a) Association elements representing multibody systems which are subsystems of the output system.
  (b) List elements with up to 3 elements:

i. The first element is `Association` element representing a multibody system which is a subsystem of the output system.

ii. The second element (optional) a `List` of replacement rules for nomenclature, applicable both to the keys and values of `Association` elements related to the associated subsystem within the scope of the function.

iii. The third element (optional) is a `List` of replacement rules to be applicable to the values of `Association` elements within the scope of the function.

Some keys in `S` can have its values setted to control the execution of the internal algorithms of the function `LinearizeSystem`. These keys are:

- `"Debug␣Mode"`: whenever its value is `"On"` messages indicating the progress of the execution of the internal algorithms are shown.

- `"Timer"`: whenever its value is `"On"` a message shows the total computation time of the function.

- "$\bar{\boldsymbol{q}}$?": whenever its value is `"No"` it means that the algorithm must not complete the list of constraint equations (i.e., all the forms of the constraint equations necessary for the correct execution of the modular modelling algorithm were already provided).

- "$\bar{\boldsymbol{C}}$?": whenever its value is `"No"` the algorithm for calculating the matrix $\tilde{\tilde{\boldsymbol{C}}}$ is not executed.

- `"Explicit␣EOM"`: whenever its value is `"Yes"`, explicit forms of the differential equations of motion (EOM) are shown, i.e., the system of EOM is presented in the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x})$.

2 Linearization of equations of motion

2.1 Reference motion

```
ReferenceMotion[xSystem_, xReferenceValues_: {}] :=
  Module[{xOut, xKeys, xVariables},
    xKeys = Part[#,1]& /@ Union @ (Flatten @ {
```

```
    (Select[Keys @ xSystem, Part[#, 0] === "q"&]),
    (Select[Keys @ xSystem[#], Part[#, 0] === "q"&])& /@ xSystem["Subsystems Labels"]
    });
  xVariables = Union @@ (Function[xKey, (Union @@ ({
    xSystem["q"[xKey]],
    Union @@ (Function[xSub, xSystem[xSub]["q"[xKey]]] /@ xSystem["Subsystems Labels"])
    } //. Missing[xX__]-> {}
    ))] /@ xKeys);
  xOut = Association @ (Flatten @ Outer[(#1-> #2)&, (Superscript[#, \[EmptySmallCircle]])& /@
    (xVariables /. SymbolReplacements), {0}]);
  AssociateTo[xOut, (Superscript[First[#], \[EmptySmallCircle]] -> Last[#])& /@
    (xReferenceValues /. SymbolReplacements)];
  xOut // Normal
  ]
```

ReferenceMotion identifies all the generalized variables in a mathematical model and creates a List of replacement rules for the reference values of these variables. Two syntaxes are admissible for this function:

- ReferenceMotion[S]: simply set all the reference values of all the generalized variables of system S to zero.
- ReferenceMotion[S,L]: L is a List of replacement rules for reference values of some of the generalized variables provided by the user; in this case, the output is a List consisting of the union of L with another List setting null reference values for all the variables that are not in L.

2.2 Linearized model

```
LinearizeSystem[xSystem_, xLinSubsystemsModels_: Association[], xExtraReferenceMotion_: {}, xExtraRules_: {}] :=
 Module[{xIn, xOut, xReferenceMotion, xKeys, xA, xTimer},
  xTimer = AbsoluteTime[];
```

```
  xIn = xOut = MoSs @ xSystem;
  (xOut[#] = xLinSubsystemsModels[#])& /@
    Intersection[xOut["Subsystems␣Labels"], Keys[xLinSubsystemsModels]];
  (xOut[#] = LinearizeSystem[xIn[#], Association[], xExtraReferenceMotion, xExtraRules])& /@
    Complement[xOut["Subsystems␣Labels"], Keys[xLinSubsystemsModels]];
  xOut["Reference␣Motion"] = Union @@ {
    xExtraReferenceMotion,
    (xIn["Reference␣Motion"] //. Missing[xX__] -> {}),
    Union @@ ((xLinSubsystemsModels[#]["Reference␣Motion"] //. Missing[xX__] -> {})& /@
      Intersection[xOut["Subsystems␣Labels"], Keys[xLinSubsystemsModels]])
    };
  xReferenceMotion = ReferenceMotion[xIn, xOut["Reference␣Motion"]];
  If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["``:``:ReferenceMotion:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01],{5,2}],
      xOut["System␣Label"]]]
    ];

  xKeys = First /@ (Select[Keys @ xIn, Part[#,0] === "q#"&]);
  xOut["q#:Def:Order"] = If[KeyExistsQ[xIn,"q#:Def:Order"],
    xIn["q#:Def:Order"],
    Max @ ToExpression @ Flatten @ (StringSplit[#, {":", "|"}]& /@ xKeys)
    ];
  (xOut["q#"[ToString @ #]]= D[
    xOut["q#"[ToString @ xOut["q#:Def:Order"]]],
    {t,(#-xOut["q#:Def:Order"])}
    ])& /@ Complement[Range[0, Max[2, xOut["q:Order"]]], Range[0, xOut["q#:Def:Order"]]];
  xKeys = Union[ReplaceRepeated[#, {{xA_, xB_} :> (ToString[xA] <> "|" <> ToString[xB])}]& @
```

```
    (Select[Flatten[#,1],(Part[#,1]>Part[#,2])&]& @(Outer[List,#,#]))
    ]& @ Range[0, Max[2, xOut["q:Order"]]];
  (xOut["q#"[#]]= D[
    xOut["q#"[Part[#,2]]],
    {t,((ToExpression @ Part[#,1]) - (ToExpression @ Part[#,2]))}
    ]& @ StringSplit[#, {":", "|"}]
  )& /@ xKeys;

  xKeys = Part[#, 1]& /@ Union @ (Select[Keys @ xIn, Part[#, 0] === "*q+"&]);
  Module[{xRules},
    (xOut["*q+"[#]] = RedundantElim @ (Linearize[xIn["*q+"[#]] (* //. xIn["_c"] *),
      xReferenceMotion] //. xExtraRules))& /@ xKeys;
    xRules = (((#-> 0)& /@ Expand @ RedundantElim @ ((Union @@ (xOut["*q+"[#]]& /@ xKeys))
         //. {xX_[t]-> 0} //. xExtraRules)) /. {({} -> 0) -> {}});
    (xOut["*q+"[#]] = RedundantElim @ ((Expand @ xOut["*q+"[#]]) //. xRules //. xExtraRules))& /@ xKeys;
    (xOut["*q+"[#]] = {})& /@ Complement[ToString /@ Range[0, Max[2, xOut["q:Order"]]], xKeys];
    xOut["_c"] = Union @@ (((xOut[#]["_c"])& /@ xOut["Subsystems Labels"]) //. Missing[xX__]-> {});
    xOut["_c"] = Union @@ {
      xOut["_c"],
      Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @ xRules
      };
    xOut["_c"] = Union @@ {
      xOut["_c"],
      ((#-> 0)& /@ (RedundantElim @ (
        (Linearize[xIn["_c"] /. {(xX_ -> xY_) -> (xX - xY)}, xReferenceMotion]
        //.xExtraRules) //.xOut["_c"]
        )))
      };
```

```
    (* (xOut["*q+"[#]] = RedundantElim @ (xOut["*q+"[#]] //. xOut["_c"] //. xExtraRules))& /@ xKeys; *)
    (xOut["*q"[#]] = Union[
      xOut["*q+"[#]] //. Missing[xX__] -> {},
      Union @@ (Function[{xSub}, xOut[xSub]["*q"[#]] //. Missing[xX__] -> {}] /@ xIn["Subsystems␣Labels"])
      ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
    ];
  If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["``:``:*q:OK",
      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
      xOut["System␣Label"]]]
    ];

  xKeys = Part[#,1]& /@ Union @ (Select[Keys @ xIn, Part[#,0] === "_q" &]);
  Module[{xFirst, xLast},
    xFirst = Linearize[(First /@ xIn["_q"[#]]), xReferenceMotion] //. xOut["_c"] //. xExtraRules;
    xLast = Linearize[(Last /@ xIn["_q"[#]]), xReferenceMotion] //. xOut["_c"] //. xExtraRules;
    xOut["_q"[#]] = Select[MapThread[(#1 - (#1 //. {xX_[t]-> 0})) -> (#2 - (#2 //. {xX_[t]-> 0}))&,
      {xFirst, xLast}, 1], (Not @ (First[#] - Last[#] === 0))&];
    xOut["_c"] = Select[Union @@ {
      xOut["_c"], xOut["_q"[#]],
      Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @ MapThread[#1 -> #2&,
        {xFirst, xLast} //. {xX_[t] -> 0}, 1]
      }, (Not @ (First[#] - Last[#] === 0))&];
    ]& /@ xKeys;
  xOut["Test␣Parameters"] = (xIn["Test␣Parameters"] //. Missing[xX__] -> {});
  Module[{xEquations, xTemp},
    xEquations = RedundantElim @ (xOut["*q+"[#]] //. xOut["_c"]);
```

```
    If[Or[xEquations === {}, SetComplement[xIn["q"][#]], xOut["q#"][#]]] === {}],
      (*-TRUE-*)
      xOut["_q"][#]] = (* xOut["_q"[#]] //. Missing[xX__]-> *) {},
      (*-FALSE-*)
      If[Not @ (KeyExistsQ[xOut, "_q?Size"]),
        (*-TRUE-*)
        xOut["_q"][#]] = Function[{xX},
          MapThread[(#1-> #2)&, {
            xX,
            Flatten @ (-LinearSolve @@ Reverse @ CoefficientArrays[xEquations, xX])
            }, 1]
          ] @ SetComplement[Intersection[xIn["q"][#]], GetVariables @ xEquations], xOut["q#"][#]]] //.xExtraRules,
        (*-FALSE-*)
        If[(ToExpression @ #) <= xOut["q:Order"],
          (*-TRUE-*)
          {xOut["_q"][#]], xTemp} = LSSolver[xEquations, xIn["q"][#]], xOut["q#"][#]],
            xOut["_c"], Union[xOut["Replacement␣Rules"], xExtraRules],
            xOut["_q?Size"], xOut["Test␣Parameters"],
            xIn["C:Symmetry"] //. Missing[xX__] -> Automatic
            ];
          xOut["Test␣Parameters"] = Union[xOut["Test␣Parameters"], xTemp],
          (*-FALSE-*)
          xOut["_q"][#]] = Expand @ (D[
            xOut["_q"[ToString @ xOut["q:Order"]]],
            {t, (ToExpression @ #) - xOut["q:Order"]}
            ] //. xOut["_c"])
          ]
        ]
      ];
```

```
   xOut["_c"] = Complement[Union @@ {xOut["_c"], xOut["_q"][#]]}, {0 -> 0}];
   ]& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
  If[xOut["Debug␣Mode"] === "On",
   Print[StringForm["``:``:_c:OK",
    NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
    xOut["System␣Label"]]]
   ];

  xKeys = (ToString @ xOut["q:Order"]);
  If[KeyExistsQ[xIn, "B"],
   xOut["B"] = SApply[
    Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //. xExtraRules)&,
    xIn["B"]
    ];
   If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["``:``:B:OK",
     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
     xOut["System␣Label"]]]
    ];
   If[And[KeyExistsQ[xIn, "C"], Complement[xIn["C"]["Column␣Labels"], xIn["q#"[xKeys]]] === {}],
    xOut["C"] = SApply[
     Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //. xExtraRules)&,
     xIn["C"]
     ],
    xOut["C"] = LSLinearizedOrthogonalComplement[
     xOut["B"],
     xOut["q#"[xKeys]],
     xOut["_c"],
     xIn["C:Symmetry"] //. Missing[xX__] -> Automatic,
     xOut["Test␣Parameters"]
     ]
```

```
        ];
      If[And[KeyExistsQ[xIn, "S"],
          Complement[xIn["S"]["Column␣Labels"], xIn["q#"[xKeys]]] ===
              {}],
        xOut["S"] = SApply[
          Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //.
              xExtraRules)&,
          xIn["S"]
          ],
        If[Not @ (xIn["S?"] === "No"),
          xA = {};
          If[KeyExistsQ[xOut[#], "S"], AppendTo[xA, xOut[#]["S"]]]& /@
              xIn["Subsystems␣Labels"];
          If[xA === {},
            xOut["S"] = xOut["C"],
            If[Not @ (xOut["q+"[xKeys]] === {}),
              AppendTo[xA, SAssemble[1, xOut["q+"[xKeys]]]]
              ];
            xOut["S"] = Linearize @ ((SAssemble @@ xA) ~SDot~ xOut["C"])
            ]
          ]
        ];
      If[xOut["Debug␣Mode"] === "On",
        Print[StringForm["‘‘:‘‘:C:OK",
          NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
          xOut["System␣Label"]]]
        ];
      ];

    xA = {};
    If[Not @ (xOut["q+"[xKeys]] === {}),
      If[KeyExistsQ[xIn, "f"],
        AppendTo[xA, SApply[
          (* Collect[ *)Simplify @ (Linearize[#, xReferenceMotion] //.
              xOut["_c"] //. xExtraRules)(* ,
```

```
          Union @@ (xOut["q"[#]]& /@ (ToString /@ Range[xOut["q:Order"], Max[2, xOut["q:Order"]]])),
          Simplify] *)&,
        SPart[xIn["f"], xOut["q+"[xKeys]]]
        ]],
      AppendTo[xA, SPart[0, xOut["q+"[xKeys]]]]
      ]
    ];
  If[KeyExistsQ[xOut[#], "*f"],
    AppendTo[xA, SApply[
      (# //. xOut["_c"] //. xExtraRules)&,
      xOut[#]["*f"]]]
    ]& /@ xOut["Subsystems␣Labels"];
  xOut["*f"] = xOut["f"] = SAssemble @@ (RedundantElim @ xA);

  If[And[KeyExistsQ[xOut, "f"], KeyExistsQ[xOut, "C"]],
    xOut["*f"] = SApply[Linearize, STranspose[xOut["C"]] ~SDot~ xOut["f"]];
    If[Or[xOut["Explicit␣EOM"] === "Yes", xOut["Explicit␣Linearized␣EOM"] === "Yes"],
      (xOut["_f"] = SReplaceFullSimplify[
        Solve[(# == 0)& /@ Flatten @ (Union @@ {SAssemble[xOut["*f"]]["Matrix"], xOut["*q"[#]]}),
          xOut["q"[#]]],
        xOut["Replacement␣Rules"]
        ])& @ (ToString @ (Max[2, xOut["q:Order"]]));
      If[xOut["Debug␣Mode"] === "On",
        Print[StringForm["``:``:_f:OK",
          NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
          xOut["System␣Label"]]]
        ];
      ];
    ];
  If[xOut["Debug␣Mode"] === "On",
    Print[StringForm["``:``:*f:OK",
```

```
    NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
    xOut["System␣Label"]]]
  ];

xOut["*f:q"] = Union @ (GetVariables @ xOut["*f"]);
xOut["System␣Parameters"] = Union @@ {
  Union @@ (xOut[#]["System␣Parameters"]&/@xOut["Subsystems␣Labels"])
      ,
  RedundantElim @ (Quiet @ GetAllVariables[ Join @@ {
    xOut["*f"]["Matrix"],
    Join @@ (xOut["*q"[#]]&/@(ToString/@Range[0,xOut["q:Order"]]))}]
       //. xX_[t]-> 0)
  };

If[xOut["Timer"] === "On",
  Print[StringForm["``:``:OK",
    NumberForm[Round[AbsoluteTime[]-xTimer,0.01],{5,2}],
    xOut["System␣Label"]]]
  ];
xOut
];
```

LinearizeSystem obtains the linearized version of a model given its nonlinear version. The syntax for this function is LinearizeSystem[S,LM,R,X]:

- S is an Association element representing a nonlinear mathematical model (e.g.: any output of MoSs)
- LM is an optional argument, *whose default value is an empty* Association, that may be an Association element whose values correspond to linearized models of some of the subsystems of the system (whenever linearized models for subsystems are already known, it makes the linearization algorithm faster).
- R is an optional argument, *whose default value is an empty* List, that may be a List element of extra replacement rules setting non-zero reference values of some of the generalized variables of the model.
- X is an optional argument, *whose default value is an empty* List, that may be a

`List` element of replacement rules for other symbolic variables in the linearized model (affects only the values in the output `Association` element, not its keys).

Some keys in `S` can have its values setted to control the execution of the internal algorithms of the function `LinearizeSystem`. These keys are:

- `"Debug␣Mode"`: whenever its value is `"On"` messages indicating the progress of the execution of the internal algorithms are shown.
- `"Timer"`: whenever its value is `"On"` a message shows the total computation time of the function.
- `"Explicit␣Linearized␣EOM"`: whenever its value is `"Yes"`, explicit forms of the differential equations of motion (EOM) are shown, i.e., the system of EOM is presented in the form $\dot{x} = f(t, x)$.

3 Auxiliar parameters evaluation

```
ParametersEval[xSystem_Association, xPhysicalParameters_, xExtraRules_:<||>] :=
 Module[{xAuxiliarParameters,xInvariants,xVariables,xCoeffA,xVarA,xCoeffC,xVarC},
  xAuxiliarParameters = {};
  (
   xInvariants = DeleteCases[Flatten @ CoefficientArrays[#, GetVariables[#]] & @
     (xSystem["*q"[#]] //. xSystem["_c"] //. xExtraRules //. xPhysicalParameters //. xAuxiliarParameters),
    _?NumericQ];
   xVariables = Union @ GetAllVariables[xInvariants];
   If[ Not @ (xVariables === {}),
    xAuxiliarParameters = Union[
     xAuxiliarParameters,
     MapThread[#1 -> #2 &, {
      xVariables,
      -LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants, xVariables])
```

```
          }, 1]
        ]
      ];
  )& /@ (ToString /@ Range[0, xSystem["q:Order"]]);

  {xCoeffA,xVarA} = SMatrixCoefficientArrays @ (xSystem["B"]);
  {xCoeffC,xVarC} = SMatrixCoefficientArrays @ (xSystem["C"]);
  xInvariants = Expand /@ RedundantElim @ (Expand /@ (Flatten @
    (SAssemble[xCoeffA[1] ~SDot~ xCoeffC[1]]["Matrix"]) //. xExtraRules //. xPhysicalParameters));
  xVariables = Union @ GetAllVariables[xInvariants];
  xAuxiliarParameters = Union[
    xAuxiliarParameters,
    MapThread[#1-> #2&, {
      xVariables,
      - LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants, xVariables])
      }, 1]
    ];
  (
    xInvariants = Expand /@ RedundantElim @ (Chop @ (Expand /@ (Flatten @
      (SAssemble[xCoeffA[1] ~SDot~ xCoeffC[#], xCoeffA[#] ~SDot~ xCoeffC[1]]["Matrix"])
      //. xExtraRules //. xPhysicalParameters //. xAuxiliarParameters)));
    xVariables = Union @ GetAllVariables[xInvariants];
    If[Not @ (xInvariants === {}),
      xAuxiliarParameters = Union[
        xAuxiliarParameters,
        MapThread[#1-> #2&, {
          xVariables,
          - LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants, xVariables])
          }, 1]
```

```
        ]
      ];
  )& /@ xVarC;
  Association[xPhysicalParameters, xAuxiliarParameters]
  ]
```

ParametersEval evaluates eventual auxiliar symbolic parameters in the linearized expressions of matrix $\tilde{\boldsymbol{C}}$ (due to the use of least squares algorithm for the calculations of orthogonal complements). Its syntax is ParametersEval[S,P,X]:

- S is an Association element representing the model of the system.
- P is a List element of replacement rules for the values of the physical parameters of the system.
- X is an optional List element (*whose default value is an empty* List) for declaring extra replacement rules.

4 Newton-Euler equations

```
NewtonEuler[xLabel_, xPositionOrientationDescription_String: "None",
 xGravitationalField_: "Default", xInertiaSymmetry_: "Central",
 xExternalActiveTorque_List: {0,0,0}, xExternalActiveForce_List: {0,0,0}] :=
 Module[{xOut},
  xOut = <|
   "System Label" -> xLabel,
   "Description" -> "Newton-Euler equations of the free rigid body " <> (ToString @ xLabel),
   "q:Order" -> 1
   |>;
  xOut["q"]["1"]] = xOut["q#"]["1"]] = {
   Subscript[v, xLabel, "x"][t],
   Subscript[v, xLabel, "y"][t],
   Subscript[v, xLabel, "z"][t],
   Subscript[\[Omega], xLabel, "x"][t],
   Subscript[\[Omega], xLabel, "y"][t],
```

```
    Subscript[\[Omega], xLabel, "z"][t]
    };

  If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription), ___~~"POSITION"~~___],
    xOut["q"["0"]] = {
      Subscript[p, xLabel, "x"][t],
      Subscript[p, xLabel, "y"][t],
      Subscript[p, xLabel, "z"][t]
      };
    xOut["*c"["1"]] = {
      Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
      Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
      Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]
      };
    xOut["_q"["1|0"]] = {
      Subscript[p, xLabel, "x"]'[t] -> Subscript[v, xLabel, "x"][t],
      Subscript[p, xLabel, "y"]'[t] -> Subscript[v, xLabel, "y"][t],
      Subscript[p, xLabel, "z"]'[t] -> Subscript[v, xLabel, "z"][t]
      };
    ];

  If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription), ___~~"QUATERNION"~~___],
    xOut["q"["0"]] = {
      Subscript[p, xLabel, "x"][t],
      Subscript[p, xLabel, "y"][t],
      Subscript[p, xLabel, "z"][t],
      Subscript[q, xLabel, "x"][t],
      Subscript[q, xLabel, "y"][t],
      Subscript[q, xLabel, "z"][t],
      Subscript[q, xLabel, "t"][t]
      };
    xOut[(ToString @ \[ScriptCapitalN]) <> "|" <> (ToString @ xLabel)] = QuatToRot @@ {
```

```
      Subscript[q, xLabel, "x"][t],
      Subscript[q, xLabel, "y"][t],
      Subscript[q, xLabel, "z"][t],
      Subscript[q, xLabel, "t"][t]
      };
    xOut["_c"] = {
      Subscript[q, xLabel, "t"][t]^2 + Subscript[q, xLabel, "x"][t]^2 +
          Subscript[q, xLabel, "y"][t]^2 + Subscript[q, xLabel, "z"][t
         ]^2 -> 1,
      1/2 Subscript[q, xLabel, "t"][t]^2 + 1/2 Subscript[q, xLabel, "x"
         ][t]^2 + 1/2 Subscript[q, xLabel, "y"][t]^2 + 1/2 Subscript[q,
          xLabel, "z"][t]^2 -> 1/2,
      1-(Subscript[q, xLabel, "x"][t]^2 + Subscript[q, xLabel, "y"][t
         ]^2 + Subscript[q, xLabel, "z"][t]^2) -> Subscript[q, xLabel,
         "t"][t]^2
      };
    xOut["*c"["0"]] = {
      -1 + Subscript[q, xLabel, "t"][t]^2 + Subscript[q, xLabel, "x"][t
         ]^2 + Subscript[q, xLabel, "y"][t]^2 + Subscript[q, xLabel, "z
         "][t]^2
      };
    xOut["*c"["1"]] = {
      Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
      Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
      Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t],
      Subscript[\[Omega], xLabel, "z"][t] + 2 Subscript[q, xLabel, "z"
         ][t] Subscript[q, xLabel, "t"]'[t] + 2 Subscript[q, xLabel, "y
         "][t] Subscript[q, xLabel, "x"]'[t]-2 Subscript[q, xLabel, "x"
         ][t] Subscript[q, xLabel, "y"]'[t]-2 Subscript[q, xLabel, "t"
         ][t] Subscript[q, xLabel, "z"]'[t],
      Subscript[\[Omega], xLabel, "y"][t] + 2 Subscript[q, xLabel, "y"
         ][t] Subscript[q, xLabel, "t"]'[t]-2 Subscript[q, xLabel, "z"
         ][t] Subscript[q, xLabel, "x"]'[t]-2 Subscript[q, xLabel, "t"
         ][t] Subscript[q, xLabel, "y"]'[t] + 2 Subscript[q, xLabel, "x
         "][t] Subscript[q, xLabel, "z"]'[t],
```

```
      Subscript[\[Omega], xLabel, "x"][t] + 2 Subscript[q, xLabel, "x"
         ][t] Subscript[q, xLabel, "t"]'[t]-2 Subscript[q, xLabel, "t"
         ][t] Subscript[q, xLabel, "x"]'[t] + 2 Subscript[q, xLabel, "z
         "][t] Subscript[q, xLabel, "y"]'[t]-2 Subscript[q, xLabel, "y"
         ][t] Subscript[q, xLabel, "z"]'[t]
      };
    xOut["_q"["1|0"]] = {
      Subscript[p, xLabel, "x"]'[t] -> Subscript[v, xLabel, "x"][t],
      Subscript[p, xLabel, "y"]'[t] -> Subscript[v, xLabel, "y"][t],
      Subscript[p, xLabel, "z"]'[t] -> Subscript[v, xLabel, "z"][t],
      Subscript[q, xLabel, "t"]'[t] -> 1/2 (-Subscript[q, xLabel, "x"][
         t] Subscript[\[Omega], xLabel, "x"][t]-Subscript[q, xLabel, "y
         "][t] Subscript[\[Omega], xLabel, "y"][t]-Subscript[q, xLabel,
          "z"][t] Subscript[\[Omega], xLabel, "z"][t]),
      Subscript[q, xLabel, "x"]'[t] -> 1/2 (Subscript[q, xLabel, "t"][t
         ] Subscript[\[Omega], xLabel, "x"][t] + Subscript[q, xLabel, "
         z"][t] Subscript[\[Omega], xLabel, "y"][t]-Subscript[q, xLabel
         , "y"][t] Subscript[\[Omega], xLabel, "z"][t]),
      Subscript[q, xLabel, "y"]'[t] -> 1/2 (-Subscript[q, xLabel, "z"][
         t] Subscript[\[Omega], xLabel, "x"][t] + Subscript[q, xLabel,
         "t"][t] Subscript[\[Omega], xLabel, "y"][t] + Subscript[q,
         xLabel, "x"][t] Subscript[\[Omega], xLabel, "z"][t]),
      Subscript[q, xLabel, "z"]'[t] -> 1/2 (Subscript[q, xLabel, "y"][t
         ] Subscript[\[Omega], xLabel, "x"][t]-Subscript[q, xLabel, "x"
         ][t] Subscript[\[Omega], xLabel, "y"][t] + Subscript[q, xLabel
         , "t"][t] Subscript[\[Omega], xLabel, "z"][t])
      };
    ];

  If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription),___
      ~~"EULER␣ANGLES"~~___] ,
    xOut["q"["0"]] = {
      Subscript[p, xLabel, "x"][t],
      Subscript[p, xLabel, "y"][t],
      Subscript[p, xLabel, "z"][t],
```

```
      Subscript[\[Psi], xLabel][t],
      Subscript[\[Phi], xLabel][t],
      Subscript[\[Theta], xLabel][t]
      };
    xOut[(ToString @ \[ScriptCapitalN]) <> "|" <> (ToString @ xLabel)]
      =
      (Rotation @@ (Characters @ (First @ StringSplit[
        xPositionOrientationDescription,{":", "|", "␣"}])))[Subscript
        [\[Psi], xLabel][t],Subscript[\[Phi], xLabel][t],Subscript[\[
        Theta], xLabel][t]];
    If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription),___
      ~~"REDUNDANT"~~___] ,
      (*-TRUE-*)
      xOut["q"["1"]] = {
        Subscript[v, xLabel, "x"][t],
        Subscript[v, xLabel, "y"][t],
        Subscript[v, xLabel, "z"][t],
        Subscript[\[Omega], xLabel, "x"][t],
        Subscript[\[Omega], xLabel, "y"][t],
        Subscript[\[Omega], xLabel, "z"][t],
        Subscript[\[Psi], xLabel]'[t],
        Subscript[\[Phi], xLabel]'[t],
        Subscript[\[Theta], xLabel]'[t]
        };
      xOut["q#"["1"]] = {
        Subscript[v, xLabel, "x"][t],
        Subscript[v, xLabel, "y"][t],
        Subscript[v, xLabel, "z"][t],
        Subscript[\[Psi], xLabel]'[t],
        Subscript[\[Phi], xLabel]'[t],
        Subscript[\[Theta], xLabel]'[t]
        };
      xOut["*c"["1"]] = {
        Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
        Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
```

```
      Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]
      };
    xOut["*q"["1"]] = Union @@ {
      ({Subscript[\[Omega], xLabel, "x"][t],Subscript[\[Omega],
         xLabel, "y"][t],Subscript[\[Omega], xLabel, "z"][t]}
        -(AngularVelocity @ xOut[(ToString @ \[ScriptCapitalN]) <> "|
          " <> (ToString @ xLabel)]))
      },
    (*-FALSE-*)
    xOut["*c"["1"]] = Union @@ {
      {Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
      Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
      Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]},
      ({Subscript[\[Omega], xLabel, "x"][t],Subscript[\[Omega],
         xLabel, "y"][t],Subscript[\[Omega], xLabel, "z"][t]}
        -(AngularVelocity @ xOut[(ToString @ \[ScriptCapitalN]) <> "|
          " <> (ToString @ xLabel)]))
      }
    ];
  ];

Module[{xI,xg},
  xI["Spherical"] = xI["S"] = ({
    {Subscript[OverBar[\[CapitalIota]], xLabel], 0, 0},
    {0, Subscript[OverBar[\[CapitalIota]], xLabel], 0},
    {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel]}
    });
  xI["Cylindrical␣x"] = xI["Cx"] = ({
    {Subscript[OverBar[\[CapitalIota]], xLabel, "a"], 0, 0},
    {0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0},
    {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"]}
    });
  xI["Cylindrical␣y"] = xI["Cy"] = ({
    {Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0, 0},
    {0, Subscript[OverBar[\[CapitalIota]], xLabel, "a"], 0},
```

```
      {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"]}
      });
    xI["Cylindrical␣z"] = xI["Cz"] = ({
      {Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0, 0},
      {0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0},
      {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "a"]}
      });
    xI["Central"] = xI["xyz"] = xI["C"] = ({
      {Subscript[OverBar[\[CapitalIota]], xLabel, "x"], 0, 0},
      {0, Subscript[OverBar[\[CapitalIota]], xLabel, "y"], 0},
      {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "z"]}
      });
    xI["xy␣Plane"] = xI["xy"] = xI["z"] = ({
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xx"], Subscript[
         OverBar[\[CapitalIota]], xLabel, "xy"], 0},
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xy"], Subscript[
         OverBar[\[CapitalIota]], xLabel, "yy"], 0},
      {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "zz"]}
      });
    xI["xz␣Plane"] = xI["xz"] = xI["y"] = ({
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xx"], 0, Subscript[
         OverBar[\[CapitalIota]], xLabel, "xz"]},
      {0, Subscript[OverBar[\[CapitalIota]], xLabel, "yy"], 0},
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xz"], 0, Subscript[
         OverBar[\[CapitalIota]], xLabel, "zz"]}
      });
    xI["yz␣Plane"] = xI["yz"] = xI["x"] = ({
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xx"], 0, 0},
      {0, Subscript[OverBar[\[CapitalIota]], xLabel, "yy"], Subscript[
         OverBar[\[CapitalIota]], xLabel, "yz"]},
      {0, Subscript[OverBar[\[CapitalIota]], xLabel, "yz"], Subscript[
         OverBar[\[CapitalIota]], xLabel, "zz"]}
      });
    xI[xX_]:= ({
```

```
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xx"], Subscript[OverBar[\[CapitalIota]], xLabel, "xy"], Subscript[OverBar[\[CapitalIota]], xLabel, "xz"]},
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xy"], Subscript[OverBar[\[CapitalIota]], xLabel, "yy"], Subscript[OverBar[\[CapitalIota]], xLabel, "yz"]},
      {Subscript[OverBar[\[CapitalIota]], xLabel, "xz"], Subscript[OverBar[\[CapitalIota]], xLabel, "yz"], Subscript[OverBar[\[CapitalIota]], xLabel, "zz"]}
    });

    xg["Default"] = OverBar[g]{Sin[OverBar[\[Xi]]], 0, Cos[OverBar[\[Xi]]]};
    xg["None"] = {0,0,0};
    xg["x"] = OverBar[g]{1,0,0};
    xg["-x"] = OverBar[g]{-1,0,0};
    xg["y"] = OverBar[g]{0,1,0};
    xg["-y"] = OverBar[g]{0,-1,0};
    xg["z"] = OverBar[g]{0,0,1};
    xg["-z"] = OverBar[g]{0,0,-1};
    xg[xL_List]:= xL;
    xg[xX_]:= OverBar[g]{Sin[xX],0,Cos[xX]};

    xOut["*f"] = xOut["f"] = <|
      "Matrix" -> Join @@ {
        - Subscript[OverBar[m], xLabel] (D[#,t]& /@
          {Subscript[v, xLabel, "x"][t], Subscript[v, xLabel, "y"][t], Subscript[v, xLabel, "z"][t]})
        + Subscript[OverBar[m], xLabel]xg[xGravitationalField]
        + xExternalActiveForce,
        - xI[xInertiaSymmetry].(D[#,t]& /@
          {Subscript[\[Omega], xLabel, "x"][t],Subscript[\[Omega], xLabel, "y"][t],Subscript[\[Omega], xLabel, "z"][t]})
        - {Subscript[\[Omega], xLabel, "x"][t], Subscript[\[Omega], xLabel, "y"][t], Subscript[\[Omega], xLabel, "z"][t]} ~Cross
```

```
            ~
           (xI[xInertiaSymmetry].{Subscript[\[Omega], xLabel, "x"][t],
              Subscript[\[Omega], xLabel, "y"][t], Subscript[\[Omega],
              xLabel, "z"][t]})
         + xExternalActiveTorque
         },
       "Row␣Labels" -> {
         Subscript[v, xLabel, "x"][t],
         Subscript[v, xLabel, "y"][t],
         Subscript[v, xLabel, "z"][t],
         Subscript[\[Omega], xLabel, "x"][t],
         Subscript[\[Omega], xLabel, "y"][t],
         Subscript[\[Omega], xLabel, "z"][t]
         }
       |>;
     ];

  xOut
  ]
```

`NewtonEuler` provides the Newton-Euler equations based model of a single free rigid-body. The syntax for this function is `NewtonEuler[L,PO,GF,IS,T,F]`:

- `L` is a label for identifying the system (typically a `String` element).
- `PO` is an optional argument for choosing the generalized coordinates for describing position and orientation of the rigid body. Its possible values are the following (non case sensitive) `Strings`:
  - `"None"` (*default value*): defines no generalized coordinates.
  - `"Position"` or `"Position␣only"`: defines 3 generalized coordinates only - 3 Cartesian coordinates of the centre of mass of the rigid body (with respect to a coordinate system fixed to an inertial reference frame); no coordinates are defined for the orientation description.
  - `"Quaternion"`: defines a set of 7 generalized coordinates - 3 Cartesian coordinates of the centre of mass with respect to a coordinate system fixed to an inertial reference frame and 4 quaternion components for describing the

orientation of a coordinate system attached to the inertial reference frame with respect to the one fixed to an inertial reference frame.

  - `"xyx␣Euler␣Angles"`, `"xyz␣Euler␣Angles"`, `"zyx␣Euler␣Angles"`, etc.: defines a set of 6 generalized coordinates - 3 Cartesian coordinates of the centre of mass with respect to a coordinate system fixed to an inertial reference frame and 3 Euler angles for describing the orientation of a coordinate system attached to the inertial reference frame with respect to the one fixed to an inertial reference frame; the convention adopted to define the Euler angles must be set by the first 3 characters of the `String`.
  - `"xyx␣Euler␣Angles␣Redundant"`, `"xyz␣Euler␣Angles␣Redundant"`, `"zyx␣Euler␣Angles␣Redundant"`, etc.: does the same as the previous case, but also defines as quasi-velocities the time derivatives of the Euler angles (thus, the set of quasi-velocities will be redundant consisting of 3 components of velocity of the centre of mass, 3 components of the angular velocity of the rigid body with respect to an inertial reference frame and 3 time derivatives of Euler angles).

- `GF` is an optional argument for defining the gravitational field. Its possible values are ($\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are the unity vectors of the coordinate system fixed to an inertial reference frame):
  - `"Default"` (*default value*): $\mathbf{g} = \bar{g}(\sin\bar{\xi}\,\hat{\mathbf{x}} + \cos\bar{\xi}\,\hat{\mathbf{z}})$
  - `"None"`: $\mathbf{g} = \mathbf{0}$
  - `"x"`: $\mathbf{g} = \bar{g}\,\hat{\mathbf{x}}$
  - `"-x"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{x}}$
  - `"y"`: $\mathbf{g} = \bar{g}\,\hat{\mathbf{y}}$
  - `"-y"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{y}}$
  - `"z"`: $\mathbf{g} = \bar{g}\,\hat{\mathbf{z}}$
  - `"-z"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{z}}$
  - Any 3 elements `List` setting the components $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ of $\mathbf{g}$.

- `IS` is an optional argument for defining the inertia symmetry of the rigid body. Its possible values are:

    - `"Central"` (*default value*): the inertia tensor with respect to the centre of mass is represented by a diagonal matrix.
    - `"Spherical"`: the inertia tensor with respect to the centre of mass is represented by a multiple of the identity matrix.
    - `"Cylindrical␣x"` or `"Cx"`: the inertia tensor with respect to the centre of mass is represented by a diagonal matrix in which the entries associated to $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are equal.
    - `"Cylindrical␣y"` or `"Cy"`: the inertia tensor with respect to the centre of mass is represented by a diagonal matrix in which the entries associated to $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are equal.
    - `"Cylindrical␣z"` or `"Cz"`: the inertia tensor with respect to the centre of mass is represented by a diagonal matrix in which the entries associated to $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are equal.
    - `"-x"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{x}}$
    - `"y"`: $\mathbf{g} = \bar{g}\,\hat{\mathbf{y}}$
    - `"-y"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{y}}$
    - `"z"`: $\mathbf{g} = \bar{g}\,\hat{\mathbf{z}}$
    - `"-z"`: $\mathbf{g} = -\bar{g}\,\hat{\mathbf{z}}$
    - Any 3 elements `List` setting the components $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ of $\mathbf{g}$.

- `T` is an optional argument for setting the 3 components, with respect to a coordinate system fixed to the body, of any external torque actuating in the rigid body. Its default value is `{0,0,0}`.

- `F` is an optional argument for setting the 3 components, with respect to a coordinate system fixed to an inertial reference frame, of any external force actuating in the rigid body. Its default value is `{0,0,0}`.

Index