

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 **Modular Modelling of Multibody Systems Based on Subsystems 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 (satisfying all the existing physical constraints).

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

Define $\mathbf{q}_n^{(0)}$ 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); $\mathbf{q}_n^{(0)}$ represents a set of variables that is enough 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 $\mathbf{q}_n^{(0)}$ and of geometrical of this subsystem. Analogously, define $\mathbf{q}_n^{(1)}$ 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); $\mathbf{q}_n^{(1)}$ represents a set of variables that is enough to parametrize, along with $\mathbf{q}_n^{(0)}$, the description of any state of \mathcal{S}_n . All components

velocities, angular velocities, linear and angular momenta of \mathcal{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 $\mathbf{q}^{(0)}$ and $\mathbf{q}^{(1)}$. Actually, $\mathbf{q}^{(1)}$ can be interpreted as a set of variables that replace $\dot{\mathbf{q}}_n^{(0)}$ in the description of any state of \mathcal{S}_n .

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

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

$$\begin{cases} \dot{\mathbf{q}}_n^{(\kappa)} = \dot{\mathbf{q}}_n^{(\kappa)}(t, \mathbf{q}_n^{\llbracket \kappa+1 \rrbracket}) & \text{for } 0 \leq \kappa \leq \sigma - 1 \\ \bar{\mathbf{q}}_n^{(\sigma)} = \tilde{\mathbf{A}}_n(t, \mathbf{q}_n^{\llbracket \sigma-1 \rrbracket}) \mathbf{q}_n^{(\sigma)} + \tilde{\mathbf{b}}_n^{(\sigma-1)}(t, \mathbf{q}_n^{\llbracket \sigma-1 \rrbracket}) = \mathbf{0} \\ \bar{\mathbf{d}}_n^{(\sigma)}(t, \mathbf{q}_n^{\llbracket \sigma \rrbracket}, \mathbf{u}_n) = \mathbf{0} \end{cases} \quad (1)$$

Define $\mathbf{q}^{(\alpha)}$ and $\mathbf{q}^{\llbracket \alpha \rrbracket}$ as the block-column-matrices constituted respectively by the $\mathbf{q}_n^{(\alpha)}$ and $\mathbf{q}_n^{\llbracket \alpha \rrbracket}$ of all the subsystems \mathcal{S}_n . Suppose that all the constraints among the subsystems can be described by equations of the form:

$$\bar{\mathbf{q}}^{(\sigma)} = \sum_n \tilde{\mathbf{A}}_n(t, \mathbf{q}^{\llbracket \sigma-1 \rrbracket}) \mathbf{q}_n^{(\sigma)} + \tilde{\mathbf{b}}^{(\sigma-1)}(t, \mathbf{q}^{\llbracket \sigma-1 \rrbracket}) = \mathbf{0} \quad (2)$$

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

$$\mathbf{A} = \begin{bmatrix} \tilde{\mathbf{A}}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \tilde{\mathbf{A}}_{\nu_{\mathcal{S}}} \\ \tilde{\tilde{\mathbf{A}}}_1 & \dots & \tilde{\tilde{\mathbf{A}}}_{\nu_{\mathcal{S}}} \end{bmatrix}$$

Let $\tilde{\tilde{\mathbf{C}}}_n$ denote an orthogonal complement of $\tilde{\mathbf{A}}_n$. Depending on the methodology used

to derive the mathematical model of \mathcal{S}_n , some expression for $\tilde{\mathbf{C}}_n$ may already be known. Define the matrix $\tilde{\mathbf{A}}$ by the expression:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \tilde{\mathbf{A}}_1 \tilde{\mathbf{C}}_1 & \tilde{\mathbf{A}}_2 \tilde{\mathbf{C}}_2 & \dots & \tilde{\mathbf{A}}_{\nu_{\mathcal{S}}} \tilde{\mathbf{C}}_{\nu_{\mathcal{S}}} \end{bmatrix}$$

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

$$\begin{cases} \dot{\mathbf{q}}_n^{(\kappa)} = \dot{\mathbf{q}}_n^{(\kappa)}(t, \mathbf{q}_n^{\langle\kappa+1\rangle}), \text{ for } 0 \leq \kappa \leq \sigma - 1, \forall n \\ \bar{\mathbf{q}}_n^{(\sigma)} = \tilde{\mathbf{A}}_n \mathbf{q}_n^{(\sigma)} + \tilde{\mathbf{b}}_n^{(\sigma-1)} = \mathbf{o}, \forall n \\ \bar{\mathbf{q}}^{(\sigma)} = \sum \tilde{\mathbf{A}}_n \mathbf{q}_n^{(\sigma)} + \tilde{\mathbf{b}}^{(\sigma-1)} = \mathbf{o} \\ \bar{\mathbf{d}}^{(\sigma)} = \tilde{\mathbf{C}}^{\top} \mathbf{d}^{(\sigma)} = \mathbf{o} \end{cases} \quad (3)$$

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 \mathcal{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: \mathcal{N} consisting of the inertial reference frame \mathcal{N} and \mathcal{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 $(p_{\mathcal{B},x}, p_{\mathcal{B},y}, p_{\mathcal{B},z})$ denote the coordinates of the centre of mass of \mathcal{B} in this coordinate system. The (external) constraint between systems \mathcal{N} and \mathcal{S} can be expressed by the following equation:

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

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

```
1 NewtonEuler["B", "PositionOnly", "-z"]
```

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

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

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

```
Syll = MoSs["S", {NewtonEuler["B", "Position", "-z"]}];
Syll["*q"["0"]] = {p["s", "x"][t]^2 + p["s", "y"][t]^2 + p["s", "z"][t]^2 - a^2};
Syll["Replacement Rules"] = {
  p["s", "x"][t]^2 + p["s", "y"][t]^2 + p["s", "z"][t]^2 -> a^2
};
Syll["q#"["0"]] = {p["s", "x"][t], p["s", "y"][t]};
Syll["q#"["1"]] = {v["s", "x"][t], v["s", "y"][t], w["s", "x"][t], w["s", "y"][t], w["s", "z"][t]};
Syll["Explicit EOM"] = "Yes";
Syll = MoSs @ Syll;

Syll["_f"] // TableForm
Syll["S"] // SMatrixForm


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


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


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


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


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


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


|                              | $\mathbf{v}_{\mathcal{B},x}$                         | $\mathbf{v}_{\mathcal{B},y}$                         | $\omega_{\mathcal{B},x}$ | $\omega_{\mathcal{B},y}$ | $\omega_{\mathcal{B},z}$ |
|------------------------------|------------------------------------------------------|------------------------------------------------------|--------------------------|--------------------------|--------------------------|
| $\mathbf{v}_{\mathcal{B},x}$ | 1                                                    | 0                                                    | 0                        | 0                        | 0                        |
| $\mathbf{v}_{\mathcal{B},y}$ | 0                                                    | 1                                                    | 0                        | 0                        | 0                        |
| $\mathbf{v}_{\mathcal{B},z}$ | $-\frac{p_{\mathcal{B},x}[t]}{p_{\mathcal{B},z}[t]}$ | $-\frac{p_{\mathcal{B},y}[t]}{p_{\mathcal{B},z}[t]}$ | 0                        | 0                        | 0                        |
| $\omega_{\mathcal{B},x}$     | 0                                                    | 0                                                    | 1                        | 0                        | 0                        |
| $\omega_{\mathcal{B},y}$     | 0                                                    | 0                                                    | 0                        | 1                        | 0                        |
| $\omega_{\mathcal{B},z}$     | 0                                                    | 0                                                    | 0                        | 0                        | 1                        |


```

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

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

$$v_{B,x}^2 + v_{B,y}^2 + v_{B,z}^2 - k_B = 0$$

This is done in the example shown in Figure 2.

```

Sy21 = MoSs["S", {NewtonEuler["B", "Position", "-z"]}];
Sy21["q"["1"]] = {kB["t"]};
Sy21["*q"["0"]] = {pB["x"["t"]]^2 + pB["y"["t"]]^2 + pB["z"["t"]]^2 - a2};
Sy21["*q"["1"]] = {vB["x"["t"]]^2 + vB["y"["t"]]^2 + vB["z"["t"]]^2 - kB["t"]};
Sy21["Replacement Rules"] = {
  pB["x"["t"]]^2 + pB["y"["t"]]^2 + pB["z"["t"]]^2 → a2,
  pB["x"["t"]] pB["x"["t"]] + pB["y"["t"]] pB["y"["t"]] + pB["z"["t"]] pB["z"["t"]] → 0,
  pB["x"["t"]]^2 + pB["y"["t"]]^2 → a2 - pB["z"["t"]]^2,
  pB["x"["t"]] pB["x"["t"]] + pB["y"["t"]] pB["y"["t"]] → -pB["z"["t"]] pB["z"["t"]],
  vB["x"["t"]]^2 + vB["y"["t"]]^2 + vB["z"["t"]]^2 → kB["t"]
};
Sy21["Explicit EOM"] = "Yes";
Sy21 = MoSs @ Sy21;

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

kB["t"] → -2 g vB,z["t"]
vB,x["t"] →  $\frac{p_{B,x}["t"] (-k_B["t"] + g p_{B,z}["t"])}{a^2}$ 
vB,y["t"] →  $\frac{p_{B,y}["t"] (-k_B["t"] + g p_{B,z}["t"])}{a^2}$ 
vB,z["t"] →  $-\frac{a^2 g + p_{B,z}["t"] (k_B["t"] - g p_{B,z}["t"])}{a^2}$ 
ωB,x["t"] →  $\frac{(I_{B,y} - I_{B,z}) \omega_{B,y}["t"] \omega_{B,z}["t"]}{I_{B,x}}$ 
ωB,y["t"] →  $\frac{(-I_{B,x} + I_{B,z}) \omega_{B,x}["t"] \omega_{B,z}["t"]}{I_{B,y}}$ 
ωB,z["t"] →  $\frac{(I_{B,x} - I_{B,y}) \omega_{B,x}["t"] \omega_{B,y}["t"]}{I_{B,z}}$ 

```

	$\tilde{q}_{S,1}$	$\tilde{q}_{S,2}$	$\tilde{q}_{S,3}$	$\tilde{q}_{S,4}$	$\tilde{q}_{S,5}$
k_B	0	0	0	$2(-p_{B,z}["t"] v_{B,x}["t"] + p_{B,x}["t"] v_{B,z}["t"])$	$2(-p_{B,y}["t"] v_{B,x}["t"] + p_{B,x}["t"] v_{B,y}["t"])$
$v_{B,x}$	0	0	0	$\frac{p_{B,x}["t"]} {p_{B,x}["t"]}$	$\frac{p_{B,x}["t"]} {p_{B,x}["t"]}$
$v_{B,y}$	0	0	0	0	1
$v_{B,z}$	0	0	0	1	0
$\omega_{B,x}$	0	0	1	0	0
$\omega_{B,y}$	0	1	0	0	0
$\omega_{B,z}$	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 $(p_{B,x}, p_{B,y}, p_{B,z})$ 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"]] = { $\phi[t]$ ,  $\theta[t]$ };
Sy31["q"["1"]] = { $\phi'[t]$ ,  $\theta'[t]$ };
Sy31["q#"["1"]] = { $\phi'[t]$ ,  $\theta'[t]$ ,  $\omega_{\mathcal{B}}, "x"[t]$ ,  $\omega_{\mathcal{B}}, "y"[t]$ ,  $\omega_{\mathcal{B}}, "z"[t]$ };
Sy31["*q"["0"]] = {
   $p_{\mathcal{B}}, "x"[t] - \bar{a} \sin[\theta[t]] \cos[\phi[t]]$ ,
   $p_{\mathcal{B}}, "y"[t] - \bar{a} \sin[\theta[t]] \sin[\phi[t]]$ ,
   $p_{\mathcal{B}}, "z"[t] + \bar{a} \cos[\theta[t]]$ };
Sy31["Explicit EOM"] = "Yes";
Sy31 = MoSs @ Sy31;

Sy31["_f"] // TableForm
Sy31["_S"] // SMatrixForm

```

$$\begin{aligned}
\dot{\mathbf{v}}_{\mathcal{B},x}[t] &\rightarrow -\mathbf{c}_{\theta[t]} \mathbf{s}_{\theta[t]} \left(\mathbf{c}_{\theta[t]} \bar{g} + \bar{a} \left(\dot{\theta}[t]^2 + \mathbf{s}_{\theta[t]}^2 \dot{\phi}[t]^2 \right) \right) \\
\dot{\mathbf{v}}_{\mathcal{B},y}[t] &\rightarrow -\mathbf{s}_{\theta[t]} \mathbf{s}_{\phi[t]} \left(\mathbf{c}_{\theta[t]} \bar{g} + \bar{a} \left(\dot{\theta}[t]^2 + \mathbf{s}_{\theta[t]}^2 \dot{\phi}[t]^2 \right) \right) \\
\dot{\mathbf{v}}_{\mathcal{B},z}[t] &\rightarrow -\bar{g} \mathbf{s}_{\theta[t]}^2 + \mathbf{c}_{\theta[t]} \bar{a} \left(\dot{\theta}[t]^2 + \mathbf{s}_{\theta[t]}^2 \dot{\phi}[t]^2 \right) \\
\dot{\omega}_{\mathcal{B},x}[t] &\rightarrow \frac{(\mathcal{I}_{\mathcal{B},y} - \mathcal{I}_{\mathcal{B},z}) \omega_{\mathcal{B},y}[t] \omega_{\mathcal{B},z}[t]}{\mathcal{I}_{\mathcal{B},x}} \\
\dot{\omega}_{\mathcal{B},y}[t] &\rightarrow \frac{(-\mathcal{I}_{\mathcal{B},x} + \mathcal{I}_{\mathcal{B},z}) \omega_{\mathcal{B},x}[t] \omega_{\mathcal{B},z}[t]}{\mathcal{I}_{\mathcal{B},y}} \\
\dot{\omega}_{\mathcal{B},z}[t] &\rightarrow \frac{(\mathcal{I}_{\mathcal{B},x} - \mathcal{I}_{\mathcal{B},y}) \omega_{\mathcal{B},x}[t] \omega_{\mathcal{B},y}[t]}{\mathcal{I}_{\mathcal{B},z}} \\
\ddot{\theta}[t] &\rightarrow \mathbf{s}_{\theta[t]} \left(-\frac{\bar{g}}{\bar{a}} + \mathbf{c}_{\theta[t]} \dot{\phi}[t]^2 \right) \\
\ddot{\phi}[t] &\rightarrow -\frac{2 \mathbf{c}_{\theta[t]} \dot{\theta}[t] \dot{\phi}[t]}{\mathbf{s}_{\theta[t]}}
\end{aligned}$$

	$\omega_{\mathcal{B},x}$	$\omega_{\mathcal{B},y}$	$\omega_{\mathcal{B},z}$	$\dot{\theta}$	$\dot{\phi}$
$\mathbf{v}_{\mathcal{B},x}$	0	0	0	$\mathbf{c}_{\theta[t]} \mathbf{c}_{\phi[t]} \bar{a}$	$-\bar{a} \mathbf{s}_{\theta[t]} \mathbf{s}_{\phi[t]}$
$\mathbf{v}_{\mathcal{B},y}$	0	0	0	$\mathbf{c}_{\theta[t]} \bar{a} \mathbf{s}_{\phi[t]}$	$\mathbf{c}_{\phi[t]} \bar{a} \mathbf{s}_{\theta[t]}$
$\mathbf{v}_{\mathcal{B},z}$	0	0	0	$\bar{a} \mathbf{s}_{\theta[t]}$	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` 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 \mathcal{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"]"] = { $\theta_1[t]$ ,  $\theta_2[t]$ };
SyDp["q"["1"]] = SyDp["q#["1"]"] = { $\theta_1'[t]$ ,  $\theta_2'[t]$ ,  $\omega_{1,x}[t]$ ,  $\omega_{1,y}[t]$ ,  $\omega_{1,z}[t]$ ,  $\omega_{2,x}[t]$ ,  $\omega_{2,y}[t]$ ,  $\omega_{2,z}[t]$ };
SyDp["*q"["0"]] = {
  p1,"x"[t] -  $\bar{a}_1 \sin[\theta_1[t]]$ ,
  p1,"y"[t],
  p1,"z"[t] +  $\bar{a}_1 \cos[\theta_1[t]]$ ,
  p2,"x"[t] -  $\bar{a}_1 \sin[\theta_1[t]] - \bar{a}_2 \sin[\theta_2[t]]$ ,
  p2,"y"[t],
  p2,"z"[t] +  $\bar{a}_1 \cos[\theta_1[t]] + \bar{a}_2 \cos[\theta_2[t]]$ 
};
SyDp[1]["Reference Motion"] = {p1,"z"[t]  $\rightarrow -\bar{a}_1$ };
SyDp[2]["Reference Motion"] = {p2,"z"[t]  $\rightarrow -\bar{a}_1 - \bar{a}_2$ };
SyDp["Explicit Linearized EOM"] = "Yes";
SyDp = MoSs @ SyDp;
LSyDp = LinearizeSystem @ SyDp;

LSyDp["_f"] // TableForm

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


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


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


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


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


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


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


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


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


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


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


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


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


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


```

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

1 Modular Modelling

```
1 MoSs[xSystem_, xSubSystems_List:{}] :=
2   Module[{xIn, xOut, xRules, xKeys, xA, xTimer},
3     xTimer = AbsoluteTime[];
4     xIn = xSystem;
5     xOut = If[AssociationQ[xIn], xIn, Association[]];
6     Quiet @ (
7       xOut["System_Label"] = xIn /. {
8         xX_List /; Length[xX] >= 1 -> xX[[1]],
9         xX_Association -> xX["System_Label"]
10      };
11      xOut["Subsystems_Labels"] = xIn /. {
12        xX_Association /; KeyExistsQ[xX, "Subsystems_Labels"] -> xX["
13        Subsystems_Labels"],
14        xX_ -> {}
15      };
16      xOut["Description"] = xIn /. {
17        xX_List /; And[Length[xX] >= 2, StringQ[xX[[2]]]] -> xX[[2]],
18        xX_Association /; And[KeyExistsQ[xX, "Description"], StringQ[xX["
19        Description"]]] -> xX["Description"],
20        xX_ -> ""
21      };
22      xOut["Naming_Rules"] = xIn /. {
23        xX_List /; Length[xX] >= 3 -> xX[[3]],
24        xX_Association /; KeyExistsQ[xX, "Naming_Rules"] -> xX["Naming_
25        Rules"],
26        xX_ -> {}
27      };
28      xOut["Replacement_Rules"] = xIn /. {
29        xX_List /; Length[xX] >= 4 -> xX[[4]],
30        xX_Association /; KeyExistsQ[xX, "Replacement_Rules"] -> xX["
31        Replacement_Rules"],
32        xX_ -> {}
33      };
34    )
35  }
```



```

30 );
31
32 Quiet @ (
33   xIn = (xSubSystems[[#]]) /. {
34     xX_List /; AssociationQ[xX[[1]]] -> xX[[1]]
35   };
36   xRules["Naming_Rules"] = Join[
37     (xSubSystems[[#]]) /. {
38       xX_List /; And[Length[xX] >= 2, Or[ListQ[xX[[2]]], AssociationQ[
39         xX[[2]]]] -> Normal @ (xX[[2]]),
40       xX_ -> {}
41     },
42     xOut["Naming_Rules"]
43   ];
44   xRules["Replacement_Rules"] = Join[
45     (xSubSystems[[#]]) /. {
46       xX_List /; And[Length[xX] >= 3, Or[ListQ[xX[[3]]], AssociationQ[xX
47         [[3]]]] -> Normal @ (xX[[3]]),
48       xX_ -> {}
49     },
50     xOut["Replacement_Rules"]
51   ];
52   xIn = SRename[xIn, xRules["Naming_Rules"], xRules["Replacement_
53     Rules"]];
54   xOut["Subsystems_Labels"] = Union[xOut["Subsystems_Labels"], {xIn["
55     System_Label"]}];
56   xOut[xIn["System_Label"]] = xIn;
57   xOut["Replacement_Rules"] = Union[xOut["Replacement_Rules"], xRules
58     ["Replacement_Rules"]];
59   )& /@ Range @ (Length @ xSubSystems);
60
61 xIn = xOut;
62 xOut["q:Order"] = If[KeyExistsQ[xIn, "q:Order"],
63   xIn["q:Order"],

```

```

59      Max @ (((xIn[#]["q:Order"])& /@ xIn["Subsystems_Labels"])) //.
      Missing[xX_] -> {}
60  ];
61  (If[xIn[#]["q:Order"] < xOut["q:Order"],
62      xIn[#]["q:Order"] = xOut["q:Order"]]; xOut[#] = MoSs @ xIn[#];
63      )& /@ xIn["Subsystems_Labels"];
64  xOut["Reference_Motion"] = Union @@ {
65      (xIn["Reference_Motion"] //. Missing[xX_] -> {}),
66      Union @@ ((xOut[#]["Reference_Motion"] //. Missing[xX_] -> {})& /@
67          xIn["Subsystems_Labels"])
68  };
69  If[xOut["Debug_Mode"] === "On",
70      Print[StringForm["'':':Subsystems:OK",
71          NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
72          xOut["System_Label"]]]
73  ];
74
75  xKeys = Part[#, 1]& /@ Union @ (Flatten @ {
76      (Select[Keys @ xIn, Part[#, 0] === "q"&]),
77      (Select[Keys @ xIn[#], Part[#, 0] === "q"&])& /@ xIn["Subsystems_
      Labels"]
78  });
79  Function[{xKey},
80      xIn["q+"[xKey]] = Complement[
81          xIn["q"[xKey]] //. Missing[xX_] -> {},
82          Union @@ (Function[{xSub}, xIn[xSub]["q"[xKey]] //. Missing[xX_]
              -> {}] /@ xIn["Subsystems_Labels"])
83      ];
84      ] /@ xKeys;
85  xKeys = Part[#, 1]& /@ Union @ (Flatten @ {
86      Select[Keys @ xIn, And[Part[#, 0] === "q+", Not @ (xIn[#] === {})]
87          ]&
88      });
89  If[xKeys === {},
      (*-TRUE-*)

```

```

90 (xOut["q+"[ToString @ #]] = {})& /@ Range[0, Max[2, xOut["q:Order"
91   ]]],
92 (*-FALSE-*)
93 (xOut["q+"[#]] = xIn["q+"[#]])& /@ xKeys;
94 xOut["q:Def:Order"] = If[KeyExistsQ[xIn,"q:Def:Order"],
95   xIn["q:Def:Order"],
96   Max @ ToExpression @ Flatten @ (StringSplit[#, {":", "|"}]& /@
97     xKeys)
98   ];
99 (xOut["q+"[ToString @ #]] = D[
100   xOut["q+"[ToString @ xOut["q:Def:Order"]]],
101   {t, (# - xOut["q:Def:Order"])}
102   ])& /@ Range[xOut["q:Def:Order"] + 1, Max[2, xOut["q:Order"]]];
103 ];
104 (xOut["q"[#]] = Union[
105   xOut["q+"[#]] /. Missing[xX_] -> {},
106   Union @@ (Function[{xSub}, xOut[xSub]["q"[#]] /. Missing[xX_] ->
107     {}] /@ xOut["Subsystems_Labels"])
108   ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
109 xKeys = Union[
110   ReplaceRepeated[#, {{xX_,xY_} :> (ToString[xX] <> "|" <> ToString[
111     xY])}]& @
112   (Select[Flatten[#, 1], (Part[#, 1] > Part[#, 2])&]& @ (Outer[List
113     , #, #]))
114   ]& @ Range[0, Max[2, xOut["q:Order"]]];
115 (xOut["q"[#]] = D[
116   xOut["q"[Part[#, 2]]] /. Missing[xX_] -> {},
117   {t, ((ToExpression @ Part[#, 1]) - (ToExpression @ Part[#, 2]))}
118   ]& @ StringSplit[#, {":", "|"}];
119 xOut["q+"[#]] = D[
120   xOut["q+"[Part[#, 2]]] /. Missing[xX_] -> {},
121   {t, ((ToExpression @ Part[#, 1]) - (ToExpression @ Part[#, 2]))}
122   ]& @ StringSplit[#, {":", "|"}];
123 )& /@ xKeys;
124 If[xOut["Debug_Mode"] === "On",

```

```

120     Print[StringForm["':q:OK",
121         NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
122         xOut["System_Label"]]]
123 ];
124
125 xKeys = Part[#,1]& /@ Union @ (Flatten @ {
126     (Select[Keys @ xIn, Part[#, 0] == "*c"&]),
127     (Select[Keys @ xIn[#, Part[#, 0] == "*c"&])& /@ xIn["Subsystems_
128         Labels"]
129     });
130 Function[xKey,
131     xOut["*c"[xKey]] = (Union @ (Flatten @ ({
132         xIn["*c"[xKey]],
133         Function[{xSub}, xIn[xSub]["*c"[xKey]] /@ xIn["Subsystems_Labels
134             "]] /. Missing[xX_] -> {})))
135     ] /@ xKeys;
136 (xOut["*c"[#]] = {})& /@ Complement[ToString /@ Range[0, xOut["q:
137     Order"]], xKeys];
138
139 xRules = Union @ (Flatten @
140     ({(* xIn["_c"], *) Function[{xSub}, xIn[xSub]["_c"]] /@ xIn["
141         Subsystems_Labels"]} /. Missing[xX_] -> {}))
142 );
143 ( xOut["_q"][(ToString @ #)<> "|" <> (ToString @ (# - 1))]] = Union @
144     (Flatten @ ({
145         xIn["_q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
146         Function[{xSub}, xIn[xSub]["_q"][(ToString @ #)<> "|" <> (ToString @
147             (#-1))]]] /@ xIn["Subsystems_Labels"]
148     } /. Missing[xX_] -> {}))
149 );
150 If[Not @ (Complement[
151     xOut["q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
152     xOut["q"][ToString @ #]], First /@ xRules,
153     First /@ xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))
154     ]]

```

```

148 ] === {}),
149 xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (#-1))]] = Union
    @@ {
150 xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (#-1))]],
151 (Simplify @ Flatten @ (Quiet @ Solve[
152   (# == 0)& /@ (RedundantElim @((RedundantElim @( Union @@ {
153     D[xOut["*c"][ToString @ (#-1)]]],t],
154     xOut["*c"][ToString @ #]]
155     } // . xRules)) // . xRules)),
156 Complement[
157   xOut["q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]],
158   xOut["q"][ToString @ #]], First/@ xRules,
159   First /@ xOut["_q"][(ToString @ #) <> "|" <> (ToString @
    (#-1))]]
160 ]
161 ])) // . xRules
162 }
163 ];
164 xRules = Union @@ {
165   xRules,
166   xOut["_q"][(ToString @ #) <> "|" <> (ToString @ (# - 1))]]
167 };
168 )& /@ Range[1, xOut["q:Order"]];
169 xOut["_c"] = Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @
170 (Union[xRules, xIn["_c"] // . Missing[xX_] -> {}]);
171 xOut["ReplacementRules"] = Union[#, # /. {(xA_ -> xB_) -> (-xA -> -
    xB)}]& @
172 (Union[#, # /. xOut["_c"]]& @ xIn["ReplacementRules"]);
173 If[xOut["DebugMode"] === "On",
174   Print[StringForm["'':':_q:OK",
175     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
176     xOut["SystemLabel"]]]]
177 ];
178
179 xA = {};

```

```

180 xKeys = (ToString @ xOut["q:Order"]);
181 If[Not @ (xOut["q+[xKeys]" === {}),
182   If[KeyExistsQ[xIn, "f"],
183     AppendTo[xA, SPart[xIn["f"], xOut["q+[xKeys]"]]],
184     AppendTo[xA, SPart[0, xOut["q+[xKeys]"]]]
185   ]
186 ];
187 If[KeyExistsQ[xOut[#], "*f"],
188   AppendTo[xA, xOut[#]["*f"]]
189   ]& /@ xOut["Subsystems_Labels"];
190 xOut["*f"] = xOut["f"] = SAssemble @@ (RedundantElim @ xA);
191 If[xOut["Debug_Mode"] === "On",
192   Print[StringForm["':':f:OK",
193     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
194     xOut["System_Label"]]]
195 ];
196
197 xKeys = Part[#, 1]& /@ (Select[Keys @ xIn, Or[Part[#, 0] === "*q",
198   Part[#, 0] === "*q+"&]]);
199 If[xKeys === {},
200   (*-TRUE-*)
201   (xOut["*q+[#]" = {})& /@ (ToString /@ Range[0, Max[2, xOut["q:
202     Order"]]]),
203   (*-FALSE-*)
204   xOut["*q:Order"] = If[KeyExistsQ[xIn, "*q:Order"], xIn["*q:Order"],
205     xOut["q:Order"]];
206   (xOut["*q+[#]" = Complement[
207     Union[xIn["*q+[#]" /. Missing[xX_] -> {}, xIn["*q+[#]" /.
208       Missing[xX_] -> {}],
209     Union @@ (Function[{xSub}, xIn[xSub]["*q+[#]" /. Missing[xX_]
210       -> {}] /@ xIn["Subsystems_Labels"])
211   ])& /@ xKeys;
212 If[Not @ (xOut["*q?"] === "No"),
213   (*-TRUE-*)
214   (xOut["*q+[#]" = Union[

```

```

210      xOut["*q+"[#]],
211      RedundantElim @ ((xOut["*c"[#]] //.{Missing[xX_] -> {}}) //.
      xOut["_c"])
212    ]& /@ xKeys;
213    (xOut["*q+"[ToString @ #]] = (RedundantElim @ ((Union @@ {
214      D[xOut["*q+"[ToString @ (# - 1)]] //.{Missing[xX_] -> {}} , t],
215      xOut["*q+"[ToString @ #]] //.{Missing[xX_] -> {}}
216    }) //.{xOut["_c"]}))
217    )& /@ Range[1, Max[2, xOut["q:Order"]]],
218    (*-FALSE-*)
219    (xOut["*q+"[ToString @ #]] = ((Union @@ {
220      D[xOut["*q+"[ToString @ (# - 1)]] //.{Missing[xX_] -> {}} , t]
221    }) //.{xOut["_c"]}))
222    )& /@ Range[xOut["*q:Order"] + 1, 2]
223  ]
224 ];
225 (xOut["*q"[#]] = Union[
226   xOut["*q+"[#]] //.{Missing[xX_] -> {}} ,
227   Union @@ (Function[{xSub}, xIn[xSub]["*q"[#]] //.{Missing[xX_] ->
     {}}] /@ xIn["Subsystems_Labels"])
228 ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
229 If[xOut["Debug_Mode"] == "On",
230   Print[StringForm["'':':*q:OK",
231     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
232     xOut["System_Label"]]]
233 ];
234
235 xA = {};
236 xKeys = (ToString @ xOut["q:Order"]);
237 If[KeyExistsQ[xOut, "*q+"[xKeys]],
238   If[And[KeyExistsQ[xOut, "q+"[xKeys]], Not @ (xOut["q+"[xKeys]] ==
     {})],
239     AppendTo[xA, Jacobi[xOut["*q+"[xKeys]], xOut["q+"[xKeys]]]]
240 ];

```

```

241     If[And[KeyExistsQ[xOut[#], "q"[xKeys]], Not @ (xOut[#]["q"[xKeys]]
242         === {})],
243     If[KeyExistsQ[xOut[#], "S"],
244         AppendTo[xA, Jacobi[xOut["*q"+[xKeys]], xOut[#]["q"[xKeys]]] ~
245             SDot~ xOut[#]["S"]],
246         AppendTo[xA, Jacobi[xOut["*q"+[xKeys]], xOut[#]["q"[xKeys]]]]
247     ]
248     ]& /@ xIn["Subsystems_Labels"];
249     xOut["B"] = SAssemble @@ xA
250 ];
251 If[xOut["Debug_Mode"] === "On",
252     Print[StringForm["':':B:OK",
253         NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
254         xOut["System_Label"]]]
255 ];
256 If[Not @ (xOut["C?"] === "No"),
257     If[xOut["B"]["Matrix"] === {},
258         xOut["C"] = SAssemble[1, xOut["B"]["Column_Labels"]],
259         If[KeyExistsQ[xOut, "q#[xKeys]],
260             xOut["C"] = OrthogonalComplement[xOut["B"], xOut["q#[xKeys]]],
261             xOut["C"] = OrthogonalComplement[xOut["B"], ToString @ xOut["
262                 System_Label"]]
263         ]
264     ];
265 If[Not @ (xOut["S?"] === "No"),
266     xA = {};
267     If[KeyExistsQ[xOut[#], "S"], AppendTo[xA, xOut[#]["S"]]]& /@ xIn[
268         "Subsystems_Labels"];
269     If[xA === {},
270         xOut["S"] = xOut["C"],
271         If[Not @ (xOut["q"+[xKeys]] === {}),
272             AppendTo[xA, SAssemble[1, xOut["q"+[xKeys]]]]
273         ];
274     xOut["S"] = (SAssemble @@ xA) ~SDot~ xOut["C"]

```



```

272 ];
273 ];
274 If[xOut["DebugMode"] === "On",
275   Print[StringForm["':':C:OK",
276     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
277     xOut["SystemLabel"]]]
278 ];
279 ];
280
281 If[And[KeyExistsQ[xOut, "f"], KeyExistsQ[xOut, "C"]],
282   xOut["*f"] = STranspose[xOut["C"]] ~SDot~ xOut["f"];
283   If[xOut["ExplicitEOM"] === "Yes",
284     (xOut["_f"] = SReplaceFullSimplify[
285       Solve[(# == 0)& /@ Flatten @ (Union @@ {SAssemble[xOut["*f"]][
286         Matrix"], xOut["*q"[#]]})),
287       xOut["q"[#]]],
288       xOut["ReplacementRules"]
289     ]& @ (ToString @ (Max[2, xOut["q:Order"]])));
290   If[xOut["DebugMode"] === "On",
291     Print[StringForm["':':_f:OK",
292       NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
293       xOut["SystemLabel"]]]
294 ];
295 ];
296 If[xOut["DebugMode"] === "On",
297   Print[StringForm["':':*f:OK",
298     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
299     xOut["SystemLabel"]]]
300 ];
301
302 If[xOut["Timer"] === "On",
303   Print[StringForm["':':OK",
304     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
305     xOut["SystemLabel"]]]

```

```

306     ];
307     xOut
308 ]

```

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:

- `"DebugMode"`: 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.
- `"q̄?"`: 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).
- `"C̃?"`: whenever its value is `"No"` the algorithm for calculating the matrix $\tilde{\mathbf{C}}$ is not executed.
- `"ExplicitEOM"`: 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 $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$.

2 Linearization of equations of motion

2.1 Reference motion

```

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

```

```

4   (Select[Keys @ xSystem, Part[#, 0] === "q"&]),
5   (Select[Keys @ xSystem[#, Part[#, 0] === "q"&])& /@ xSystem["
      Subsystems_Labels"]
6   });
7   xVariables = Union @@ (Function[xKey, (Union @@ ({
8     xSystem["q"[xKey]],
9     Union @@ (Function[xSub, xSystem[xSub]["q"[xKey]]) /@ xSystem["
      Subsystems_Labels"])]
10  } /. Missing[xX_] -> {}
11  ))] /@ xKeys);
12  xOut = Association @ (Flatten @ Outer[(#1 -> #2)&, (Superscript[#, \[
      EmptySmallCircle]])& /@
13    (xVariables /. SymbolReplacements), {0}]);
14  AssociateTo[xOut, (Superscript[First[#], \[EmptySmallCircle]] ->
      Last[#])& /@
15    (xReferenceValues /. SymbolReplacements)];
16  xOut // Normal
17  ]

```

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

```

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

```

```

4  xIn = xOut = MoSs @ xSystem;
5  (xOut[#] = xLinSubsystemsModels[#])& /@
6  Intersection[xOut["Subsystems_Labels"], Keys[xLinSubsystemsModels
    ]];
7  (xOut[#] = LinearizeSystem[xIn[#], Association[],
    xExtraReferenceMotion, xExtraRules])& /@
8  Complement[xOut["Subsystems_Labels"], Keys[xLinSubsystemsModels]];
9  xOut["Reference_Motion"] = Union @@ {
10  xExtraReferenceMotion,
11  (xIn["Reference_Motion"] /. Missing[xX_] -> {}),
12  Union @@ ((xLinSubsystemsModels[#]["Reference_Motion"] /. Missing[
    xX_] -> {})& /@
13  Intersection[xOut["Subsystems_Labels"], Keys[xLinSubsystemsModels
    ]])
14  };
15  xReferenceMotion = ReferenceMotion[xIn, xOut["Reference_Motion"]];
16  If[xOut["Debug_Mode"] === "On",
17  Print[StringForm["':ReferenceMotion:OK",
18  NumberForm[Round[AbsoluteTime[] - xTimer, 0.01],{5,2}],
19  xOut["System_Label"]]]
20  ];
21
22  xKeys = First /@ (Select[Keys @ xIn, Part[#,0] === "q#&"]);
23  xOut["q#:Def:Order"] = If[KeyExistsQ[xIn,"q#:Def:Order"],
24  xIn["q#:Def:Order"],
25  Max @ ToExpression @ Flatten @ (StringSplit[#, {":", "|"}]& /@
    xKeys)
26  ];
27  (xOut["q#" [ToString @ #]] = D[
28  xOut["q#" [ToString @ xOut["q#:Def:Order"]]],
29  {t, (#-xOut["q#:Def:Order"])}
30  ])& /@ Complement[Range[0, Max[2, xOut["q:Order"]]], Range[0, xOut
    ["q#:Def:Order"]]];
31  xKeys = Union[ReplaceRepeated[#, {{xA_, xB_} :> (ToString[xA] <> "|"
    <> ToString[xB])}]& @

```

```

32     (Select[Flatten[#,1],(Part[#,1]>Part[#,2])&& @ (Outer[List, #, #]))
33     ]& @ Range[0, Max[2, xOut["q:Order"]]]];
34 (xOut["q#"[#]] = D[
35     xOut["q#"[Part[#,2]]],
36     {t, ((ToExpression @ Part[#,1]) - (ToExpression @ Part[#,2]))}
37     ]& @ StringSplit[#, {":", "|"}]
38 )& /@ xKeys;
39
40 xKeys = Part[#, 1]& /@ Union @ (Select[Keys @ xIn, Part[#, 0] === "*"
41     q+"&]);
42 Module[{xRules},
43     (xOut["*q+"[#]] = RedundantElim @ (Linearize[xIn["*q+"[#]] (* //.
44         xIn["_c"] *),
45         xReferenceMotion] //. xExtraRules))& /@ xKeys;
46     xRules = (((#-> 0)& /@ Expand @ RedundantElim @ ((Union @@ (xOut["*
47         q+"[#]]& /@ xKeys))
48         //. {xX_[t]-> 0} //. xExtraRules)) /. {{{} -> 0} -> {}});
49     (xOut["*q+"[#]] = RedundantElim @ ((Expand @ xOut["*q+"[#]]) //.
50         xRules //. xExtraRules))& /@ xKeys;
51     (xOut["*q+"[#]] = {})& /@ Complement[ToString /@ Range[0, Max[2,
52         xOut["q:Order"]]]], xKeys];
53     xOut["_c"] = Union @@ ((xOut[#]["_c"])& /@ xOut["Subsystems_Labels
54         "] // Missing[xX_]-> {});
55     xOut["_c"] = Union @@ {
56         xOut["_c"],
57         Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @ xRules
58     };
59     xOut["_c"] = Union @@ {
60         xOut["_c"],
61         ((#-> 0)& /@ (RedundantElim @ (
62             (Linearize[xIn["_c"] /. {(xX_ -> xY_) -> (xX - xY)},
63             xReferenceMotion]
64             //. xExtraRules) //. xOut["_c"]
65         )))
66     };

```

```

60      (* (xOut["*q+"[#]] = RedundantElim @ (xOut["*q+"[#]] //.{xOut["_c
        "] //.{xExtraRules}))& /@ xKeys; *)
61      (xOut["*q"[#]] = Union[
62          xOut["*q+"[#]] //.{Missing[xX_] -> {}},
63          Union @@ (Function[{xSub}, xOut[xSub]["*q"[#]] //.{Missing[xX_]
            -> {}}] /@ xIn["Subsystems_Labels"]))
64      ])& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
65      ];
66      If[xOut["Debug_Mode"] === "On",
67          Print[StringForm["':':*q:OK",
68              NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
69              xOut["System_Label"]]]
70      ];
71
72      xKeys = Part[#,1]& /@ Union @ (Select[Keys @ xIn, Part[#,0] === "_q"
        &]);
73      Module[{xFirst, xLast},
74          xFirst = Linearize[(First /@ xIn["_q"[#]]), xReferenceMotion] //.
            xOut["_c"] //.{xExtraRules};
75          xLast = Linearize[(Last /@ xIn["_q"[#]]), xReferenceMotion] //.
            xOut["_c"] //.{xExtraRules};
76          xOut["_q"[#]] = Select[MapThread[(#1 - (#1 //.{xX_[t]-> 0})) ->
            (#2 - (#2 //.{xX_[t]-> 0}))&,
77              {xFirst, xLast}, 1], (Not @ (First[#] - Last[#] === 0))&];
78          xOut["_c"] = Select[Union @@ {
79              xOut["_c"], xOut["_q"[#]],
80              Union[#, # /. {(xA_ -> xB_) -> (-xA -> -xB)}]& @ MapThread[#1 ->
                #2&,
81                  {xFirst, xLast} //.{xX_[t] -> 0}, 1]
82              }, (Not @ (First[#] - Last[#] === 0))&];
83          ]& /@ xKeys;
84          xOut["Test_Parameters"] = (xIn["Test_Parameters"] //.{Missing[xX_]
            -> {}});
85      Module[{xEquations, xTemp},
86          xEquations = RedundantElim @ (xOut["*q+"[#]] //.{xOut["_c"]});

```

```

87 If[Or[xEquations === {}, SetComplement[xIn["q"[#]], xOut["q#"[#]]]
    === {}],
88   (*-TRUE-*)
89   xOut["_q"[#]] = (* xOut["_q"[#]] //. Missing[xX__]-> *) {},
90   (*-FALSE-*)
91   If[Not @ (KeyExistsQ[xOut, "_q?Size"]),
92     (*-TRUE-*)
93     xOut["_q"[#]] = Function[{xX},
94       MapThread[(#1-> #2)&, {
95         xX,
96         Flatten @ (-LinearSolve @@ Reverse @ CoefficientArrays[
97           xEquations, xX])
98       ] @ SetComplement[Intersection[xIn["q"[#]], GetVariables @
99         xEquations], xOut["q#"[#]]] //. xExtraRules,
100     (*-FALSE-*)
101     If[(ToExpression @ #) <= xOut["q:Order"],
102       (*-TRUE-*)
103       {xOut["_q"[#]], xTemp} = LSSolver[xEquations, xIn["q"[#]],
104         xOut["q#"[#]],
105         xOut["_c"], Union[xOut["ReplacementRules"], xExtraRules],
106         xOut["_q?Size"], xOut["TestParameters"],
107         xIn["C:Symmetry"] //. Missing[xX__] -> Automatic
108       ];
109       xOut["TestParameters"] = Union[xOut["TestParameters"],
110         xTemp],
111       (*-FALSE-*)
112       xOut["_q"[#]] = Expand @ (D[
113         xOut["_q" [ToString @ xOut["q:Order"]]],
114         {t, (ToExpression @ #) - xOut["q:Order"]}
115       ] // . xOut["_c"])
116     ]
117   ]
118 ];

```



```

116     xOut["_c"] = Complement[Union @@ {xOut["_c"], xOut["_q"[#]]}, {0 ->
117         0}];
118     ]& /@ (ToString /@ Range[0, Max[2, xOut["q:Order"]]]);
119 If[xOut["DebugMode"] === "On",
120     Print[StringForm["'':':_c:OK",
121         NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
122         xOut["SystemLabel"]]]
123 ];
124 xKeys = (ToString @ xOut["q:Order"]);
125 If[KeyExistsQ[xIn, "B"],
126     xOut["B"] = SApply[
127         Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //.
128             xExtraRules)&,
129         xIn["B"]
130     ];
131 If[xOut["DebugMode"] === "On",
132     Print[StringForm["'':':B:OK",
133         NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
134         xOut["SystemLabel"]]]
135 ];
136 If[And[KeyExistsQ[xIn, "C"], Complement[xIn["C"] ["ColumnLabels"],
137     xIn["q#" [xKeys]]] === {}],
138     xOut["C"] = SApply[
139         Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //.
140             xExtraRules)&,
141         xIn["C"]
142     ],
143     xOut["C"] = LSLinearizedOrthogonalComplement[
144         xOut["B"],
145         xOut["q#" [xKeys]],
146         xOut["_c"],
147         xIn["C:Symmetry"] //. Missing[xX_] -> Automatic,
148         xOut["TestParameters"]
149     ]

```

```

147 ];
148 If[And[KeyExistsQ[xIn, "S"],
149     Complement[xIn["S"]["ColumnLabels"], xIn["q#" [xKeys]]] ===
150     {}],
151     xOut["S"] = SApply[
152         Simplify @ (Linearize[#, xReferenceMotion] //. xOut["_c"] //.
153             xExtraRules)&,
154         xIn["S"]
155     ],
156     If[Not @ (xIn["S?"] === "No"),
157         xA = {};
158         If[KeyExistsQ[xOut[#], "S"], AppendTo[xA, xOut[#]["S"]]]& /@
159             xIn["SubsystemsLabels"];
160         If[xA === {},
161             xOut["S"] = xOut["C"],
162             If[Not @ (xOut["q+" [xKeys]] === {}),
163                 AppendTo[xA, SAssemble[1, xOut["q+" [xKeys]]]]
164             ];
165             xOut["S"] = Linearize @ ((SAssemble @@ xA) ~SDot~ xOut["C"])
166         ]
167     ];
168     If[xOut["DebugMode"] === "On",
169         Print[StringForm["':':C:OK",
170             NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
171             xOut["SystemLabel"]]]
172 ];
173 xA = {};
174 If[Not @ (xOut["q+" [xKeys]] === {}),
175     If[KeyExistsQ[xIn, "f"],
176         AppendTo[xA, SApply[
177             (* Collect[ *)Simplify @ (Linearize[#, xReferenceMotion] //.
178                 xOut["_c"] //. xExtraRules)(* ,

```

```

178      Union @@ (xOut["q"#[#]]& /@ (ToString /@ Range[xOut["q:Order
179      ], Max[2, xOut["q:Order"]]])),
180      Simplify] *)&,
181      SPart[xIn["f"], xOut["q+"[xKeys]]]
182      ]],
183      AppendTo[xA, SPart[0, xOut["q+"[xKeys]]]]
184      ];
185      If[KeyExistsQ[xOut[#], "f"],
186      AppendTo[xA, SApply[
187      (# /. xOut["_c"] /. xExtraRules)&,
188      xOut[#]["*f"]]]
189      ]& /@ xOut["Subsystems_Labels"];
190      xOut["*f"] = xOut["f"] = SASsemble @@ (RedundantElim @ xA);
191
192      If[And[KeyExistsQ[xOut, "f"], KeyExistsQ[xOut, "C"]],
193      xOut["*f"] = SApply[Linearize, STranspose[xOut["C"]] ~SDot~ xOut["f
194      "]];
195      If[Or[xOut["Explicit_EOM"] === "Yes", xOut["Explicit_Linearized_EOM
196      "] === "Yes"],
197      (xOut["_f"] = SReplaceFullSimplify[
198      Solve[(# == 0)& /@ Flatten @ (Union @@ {SASsemble[xOut["*f"]]["
199      Matrix"], xOut["*q"#[#]]})),
200      xOut["q"#[#]]],
201      xOut["Replacement_Rules"]
202      ])& @ (ToString @ (Max[2, xOut["q:Order"]]]));
203      If[xOut["Debug_Mode"] === "On",
204      Print[StringForm["'':':_f:OK",
205      NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
206      xOut["System_Label"]]]
207      ];
208      ];
209      ];
210      If[xOut["Debug_Mode"] === "On",
211      Print[StringForm["'':':*f:OK",

```

```

209     NumberForm[Round[AbsoluteTime[] - xTimer, 0.01], {5, 2}],
210     xOut["System_Label"]]]
211 ];
212
213 xOut["*f:q"] = Union @ (GetVariables @ xOut["*f"]);
214 xOut["System_Parameters"] = Union @@ {
215     Union @@ (xOut[#]["System_Parameters"]&/@xOut["Subsystems_Labels"])
216     ,
217     RedundantElim @ (Quiet @ GetAllVariables[ Join @@ {
218         xOut["*f"]["Matrix"],
219         Join @@ (xOut["*q"][#]&/@(ToString/@Range[0,xOut["q:Order"]]))}]
220         //. xX_[t]-> 0)
221     };
222
223 If[xOut["Timer"] === "On",
224     Print[StringForm["':':OK",
225         NumberForm[Round[AbsoluteTime[]-xTimer,0.01],{5,2}],
226         xOut["System_Label"]]]]
227 ];

```

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:

- "DebugMode": 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.
- "ExplicitLinearizedEOM": 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 $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$.

3 Auxiliar parameters evaluation

```

1 ParametersEval[xSystem_Association, xPhysicalParameters_, xExtraRules_:
  <|>] :=
2 Module[{xAuxiliarParameters, xInvariants, xVariables, xCoeffA, xVarA,
  xCoeffC, xVarC},
3   xAuxiliarParameters = {};
4   (
5     xInvariants = DeleteCases[Flatten @ CoefficientArrays[#,
      GetVariables[#]] & @
6     (xSystem["*q"[#]] //. xSystem["_c"] //. xExtraRules //.
      xPhysicalParameters //. xAuxiliarParameters),
7     _?NumericQ];
8   xVariables = Union @ GetAllVariables[xInvariants];
9   If[ Not @ (xVariables === {}),
10    xAuxiliarParameters = Union[
11      xAuxiliarParameters,
12      MapThread[#1 -> #2 &, {
13        xVariables,
14        -LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants,
          xVariables])

```

```

15     }, 1]
16   ]
17 ];
18 )& /@ (ToString /@ Range[0, xSystem["q:Order"]]);
19
20 {xCoeffA,xVarA} = SMatrixCoefficientArrays @ (xSystem["B"]);
21 {xCoeffC,xVarC} = SMatrixCoefficientArrays @ (xSystem["C"]);
22 xInvariants = Expand /@ RedundantElim @ (Expand /@ (Flatten @
23   (SAssemble[xCoeffA[1] ~SDot~ xCoeffC[1]]["Matrix"]) //. xExtraRules
24     //. xPhysicalParameters));
25 xVariables = Union @ GetAllVariables[xInvariants];
26 xAuxiliarParameters = Union[
27   xAuxiliarParameters,
28   MapThread[#1-> #2&, {
29     xVariables,
30     - LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants,
31       xVariables])
32   }, 1]
33 ];
34 (
35   xInvariants = Expand /@ RedundantElim @ (Chop @ (Expand /@ (Flatten
36     @
37     (SAssemble[xCoeffA[1] ~SDot~ xCoeffC[#], xCoeffA[#] ~SDot~
38       xCoeffC[1]]["Matrix"])
39     //. xExtraRules //. xPhysicalParameters //. xAuxiliarParameters))
40   );
41   xVariables = Union @ GetAllVariables[xInvariants];
42   If[Not @ (xInvariants === {}),
43     xAuxiliarParameters = Union[
44       xAuxiliarParameters,
45       MapThread[#1-> #2&, {
46         xVariables,
47         - LeastSquares @@ (Reverse @ CoefficientArrays[xInvariants,
48           xVariables])
49       }, 1]

```

```

44     ]
45 ];
46 )& /@ xVarC;
47 Association[xPhysicalParameters, xAuxiliarParameters]
48 ]

```

`ParametersEval` evaluates eventual auxiliar symbolic parameters in the linearized expressions of matrix $\tilde{\mathbf{C}}$ (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

```

1 NewtonEuler[xLabel_, xPositionOrientationDescription_String: "None",
2   xGravitationalField_: "Default", xInertiaSymmetry_: "Central",
3   xExternalActiveTorque_List: {0,0,0}, xExternalActiveForce_List:
4     {0,0,0}] :=
5   Module[{xOut},
6     xOut = <|
7       "System_Label" -> xLabel,
8       "Description" -> "Newton-Euler_equations_of_the_free_rigid_body_"
9         <> (ToString @ xLabel),
10      "q:Order" -> 1
11    |>;
12    xOut["q"]["1"] = xOut["q#"]["1"] = {
13      Subscript[v, xLabel, "x"][t],
14      Subscript[v, xLabel, "y"][t],
15      Subscript[v, xLabel, "z"][t],
16      Subscript[\[Omega], xLabel, "x"][t],
17      Subscript[\[Omega], xLabel, "y"][t],

```

```

16     Subscript[\[Omega], xLabel, "z"][t]
17 };
18
19 If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription), ___
    ~~"POSITION"~~___],
20     xOut["q"]["0"] = {
21         Subscript[p, xLabel, "x"][t],
22         Subscript[p, xLabel, "y"][t],
23         Subscript[p, xLabel, "z"][t]
24     };
25     xOut["*c"]["1"] = {
26         Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
27         Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
28         Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]
29     };
30     xOut["_q"]["1|0"] = {
31         Subscript[p, xLabel, "x"]'[t] -> Subscript[v, xLabel, "x"][t],
32         Subscript[p, xLabel, "y"]'[t] -> Subscript[v, xLabel, "y"][t],
33         Subscript[p, xLabel, "z"]'[t] -> Subscript[v, xLabel, "z"][t]
34     };
35 ];
36
37 If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription), ___
    ~~"QUATERNION"~~___],
38     xOut["q"]["0"] = {
39         Subscript[p, xLabel, "x"][t],
40         Subscript[p, xLabel, "y"][t],
41         Subscript[p, xLabel, "z"][t],
42         Subscript[q, xLabel, "x"][t],
43         Subscript[q, xLabel, "y"][t],
44         Subscript[q, xLabel, "z"][t],
45         Subscript[q, xLabel, "t"][t]
46     };
47     xOut[(ToString @ \[ScriptCapitalN]) <> "|" <> (ToString @ xLabel)]
    = QuatToRot @@ {

```



```

48     Subscript[q, xLabel, "x"][t],
49     Subscript[q, xLabel, "y"][t],
50     Subscript[q, xLabel, "z"][t],
51     Subscript[q, xLabel, "t"][t]
52 };
53 xOut["_c"] = {
54     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,
55     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,
56     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
57 };
58 xOut["*c"]["0"] = {
59     -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
60 };
61 xOut["*c"]["1"] = {
62     Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
63     Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
64     Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t],
65     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],
66     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],

```

```

67      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]
68      };
69      xOut["_q"]["1|0"] = {
70      Subscript[p, xLabel, "x"]'[t] -> Subscript[v, xLabel, "x"][t],
71      Subscript[p, xLabel, "y"]'[t] -> Subscript[v, xLabel, "y"][t],
72      Subscript[p, xLabel, "z"]'[t] -> Subscript[v, xLabel, "z"][t],
73      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]),
74      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]),
75      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]),
76      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])
77      };
78      ];
79
80      If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription),___
        ~~"EULER_ANGLES"~~___] ,
81      xOut["q"]["0"] = {
82      Subscript[p, xLabel, "x"][t],
83      Subscript[p, xLabel, "y"][t],
84      Subscript[p, xLabel, "z"][t],

```

```

85     Subscript[\[Psi], xLabel][t],
86     Subscript[\[Phi], xLabel][t],
87     Subscript[\[Theta], xLabel][t]
88 };
89 xOut[(ToString @ \[ScriptCapitalN]) <> "|" <> (ToString @ xLabel)]
    =
90 (Rotation @@ (Characters @ (First @ StringSplit[
    xPositionOrientationDescription,{":", "|", "\_"}])))[Subscript
    [\[Psi], xLabel][t],Subscript[\[Phi], xLabel][t],Subscript[\[
    Theta], xLabel][t]];
91 If[StringMatchQ[(ToUpperCase @ xPositionOrientationDescription),___
    ~~"REDUNDANT"~~___] ,
92     (*-TRUE-*)
93     xOut["q"["1"]] = {
94         Subscript[v, xLabel, "x"][t],
95         Subscript[v, xLabel, "y"][t],
96         Subscript[v, xLabel, "z"][t],
97         Subscript[\[Omega], xLabel, "x"][t],
98         Subscript[\[Omega], xLabel, "y"][t],
99         Subscript[\[Omega], xLabel, "z"][t],
100        Subscript[\[Psi], xLabel]'[t],
101        Subscript[\[Phi], xLabel]'[t],
102        Subscript[\[Theta], xLabel]'[t]
103    };
104 xOut["q#"["1"]] = {
105     Subscript[v, xLabel, "x"][t],
106     Subscript[v, xLabel, "y"][t],
107     Subscript[v, xLabel, "z"][t],
108     Subscript[\[Psi], xLabel]'[t],
109     Subscript[\[Phi], xLabel]'[t],
110     Subscript[\[Theta], xLabel]'[t]
111 };
112 xOut["*c"["1"]] = {
113     Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
114     Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],

```

```

115     Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]
116 };
117 xOut["*q"["1"]] = Union @@ {
118     ({Subscript[\[Omega], xLabel, "x"][t],Subscript[\[Omega],
119         xLabel, "y"][t],Subscript[\[Omega], xLabel, "z"][t]}
120     -(AngularVelocity @ xOut[(ToString @ \[ScriptCapitalN]) <> "I
121         " <> (ToString @ xLabel)]))
122 },
123 (*-FALSE-*)
124 xOut["*c"["1"]] = Union @@ {
125     {Subscript[v, xLabel, "x"][t]-Subscript[p, xLabel, "x"]'[t],
126     Subscript[v, xLabel, "y"][t]-Subscript[p, xLabel, "y"]'[t],
127     Subscript[v, xLabel, "z"][t]-Subscript[p, xLabel, "z"]'[t]},
128     ({Subscript[\[Omega], xLabel, "x"][t],Subscript[\[Omega],
129         xLabel, "y"][t],Subscript[\[Omega], xLabel, "z"][t]}
130     -(AngularVelocity @ xOut[(ToString @ \[ScriptCapitalN]) <> "I
131         " <> (ToString @ xLabel)]))
132 }
133 ];
134 ];
135
136 Module[{xI,xg},
137     xI["Spherical"] = xI["S"] = ({
138         {Subscript[OverBar[\[CapitalIota]], xLabel], 0, 0},
139         {0, Subscript[OverBar[\[CapitalIota]], xLabel], 0},
140         {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel]}
141     });
142     xI["Cylindrical_⊥x"] = xI["Cx"] = ({
143         {Subscript[OverBar[\[CapitalIota]], xLabel, "a"], 0, 0},
144         {0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0},
145         {0, 0, Subscript[OverBar[\[CapitalIota]], xLabel, "r"]}
146     });
147     xI["Cylindrical_⊥y"] = xI["Cy"] = ({
148         {Subscript[OverBar[\[CapitalIota]], xLabel, "r"], 0, 0},
149         {0, Subscript[OverBar[\[CapitalIota]], xLabel, "a"], 0},

```

```

146     {0, 0, Subscript[OverBar\[CapitalIota]], xLabel, "r"]}]
147     });
148     xI["Cylindrical␣z"] = xI["Cz"] = ({
149     {Subscript[OverBar\[CapitalIota]], xLabel, "r"], 0, 0},
150     {0, Subscript[OverBar\[CapitalIota]], xLabel, "r"], 0},
151     {0, 0, Subscript[OverBar\[CapitalIota]], xLabel, "a"]}]
152     });
153     xI["Central"] = xI["xyz"] = xI["C"] = ({
154     {Subscript[OverBar\[CapitalIota]], xLabel, "x"], 0, 0},
155     {0, Subscript[OverBar\[CapitalIota]], xLabel, "y"], 0},
156     {0, 0, Subscript[OverBar\[CapitalIota]], xLabel, "z"]}]
157     });
158     xI["xy␣Plane"] = xI["xy"] = xI["z"] = ({
159     {Subscript[OverBar\[CapitalIota]], xLabel, "xx"], Subscript[
160     OverBar\[CapitalIota]], xLabel, "xy"], 0},
161     {Subscript[OverBar\[CapitalIota]], xLabel, "xy"], Subscript[
162     OverBar\[CapitalIota]], xLabel, "yy"], 0},
163     {0, 0, Subscript[OverBar\[CapitalIota]], xLabel, "zz"]}]
164     });
165     xI["xz␣Plane"] = xI["xz"] = xI["y"] = ({
166     {Subscript[OverBar\[CapitalIota]], xLabel, "xx"], 0, Subscript[
167     OverBar\[CapitalIota]], xLabel, "xz"]},
168     {0, Subscript[OverBar\[CapitalIota]], xLabel, "yy"], 0},
169     {Subscript[OverBar\[CapitalIota]], xLabel, "xz"], 0, Subscript[
170     OverBar\[CapitalIota]], xLabel, "zz"]}]
171     });
172     xI["yz␣Plane"] = xI["yz"] = xI["x"] = ({
173     {Subscript[OverBar\[CapitalIota]], xLabel, "xx"], 0, 0},
174     {0, Subscript[OverBar\[CapitalIota]], xLabel, "yy"], Subscript[
175     OverBar\[CapitalIota]], xLabel, "yz"]},
176     {0, Subscript[OverBar\[CapitalIota]], xLabel, "yz"], Subscript[
177     OverBar\[CapitalIota]], xLabel, "zz"]}]
178     });
179     xI[xX_] := ({

```

```

174      {Subscript[OverBar\[CapitalIota]], xLabel, "xx"}, Subscript[
      OverBar\[CapitalIota]], xLabel, "xy"}, Subscript[OverBar\[
      CapitalIota]], xLabel, "xz"]},
175      {Subscript[OverBar\[CapitalIota]], xLabel, "xy"}, Subscript[
      OverBar\[CapitalIota]], xLabel, "yy"}, Subscript[OverBar\[
      CapitalIota]], xLabel, "yz"}},
176      {Subscript[OverBar\[CapitalIota]], xLabel, "xz"}, Subscript[
      OverBar\[CapitalIota]], xLabel, "yz"}, Subscript[OverBar\[
      CapitalIota]], xLabel, "zz"]}
177  });
178
179  xg["Default"] = OverBar[g]{Sin[OverBar\[Xi]], 0, Cos[OverBar\[Xi
      ]]]};
180  xg["None"] = {0,0,0};
181  xg["x"] = OverBar[g]{1,0,0};
182  xg["-x"] = OverBar[g]{-1,0,0};
183  xg["y"] = OverBar[g]{0,1,0};
184  xg["-y"] = OverBar[g]{0,-1,0};
185  xg["z"] = OverBar[g]{0,0,1};
186  xg["-z"] = OverBar[g]{0,0,-1};
187  xg[xL_List]:= xL;
188  xg[xX_]:= OverBar[g]{Sin[xX],0,Cos[xX]};
189
190  xOut["*f"] = xOut["f"] = <|
191    "Matrix" -> Join @@ {
192      - Subscript[OverBar[m], xLabel] (D[#,t]& /@
193        {Subscript[v, xLabel, "x"][t], Subscript[v, xLabel, "y"][t],
          Subscript[v, xLabel, "z"][t]})
194      + Subscript[OverBar[m], xLabel]xg[xGravitationalField]
195      + xExternalActiveForce,
196      - xI[xInertiaSymmetry].(D[#,t]& /@
197        {Subscript\[Omega], xLabel, "x"[t],Subscript\[Omega],
          xLabel, "y"[t],Subscript\[Omega], xLabel, "z"[t]})
198      - {Subscript\[Omega], xLabel, "x"[t], Subscript\[Omega],
          xLabel, "y"[t], Subscript\[Omega], xLabel, "z"[t]} ~Cross

```

```

~
199      (xI[xInertiaSymmetry].{Subscript[\[Omega], xLabel, "x"][t],
      Subscript[\[Omega], xLabel, "y"][t], Subscript[\[Omega],
      xLabel, "z"][t]})
200      + xExternalActiveTorque
201      },
202      "RowLabels" -> {
203          Subscript[v, xLabel, "x"][t],
204          Subscript[v, xLabel, "y"][t],
205          Subscript[v, xLabel, "z"][t],
206          Subscript[\[Omega], xLabel, "x"][t],
207          Subscript[\[Omega], xLabel, "y"][t],
208          Subscript[\[Omega], xLabel, "z"][t]
209      }
210      |>;
211      ];
212
213      xOut
214      ]

```

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{y} and \hat{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{x} and \hat{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{x} and \hat{y} are equal.
 - "-x": $\mathbf{g} = -\bar{g} \hat{x}$
 - "y": $\mathbf{g} = \bar{g} \hat{y}$
 - "-y": $\mathbf{g} = -\bar{g} \hat{y}$
 - "z": $\mathbf{g} = \bar{g} \hat{z}$
 - "-z": $\mathbf{g} = -\bar{g} \hat{z}$
 - Any 3 elements List setting the components \hat{x} , \hat{y} and \hat{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

LinearizeSystem, 19

MoSs, 7

NewtonEuler, 30

ParametersEval, 28

ReferenceMotion, 18