OpenSYMORO: An open-source software package for Symbolic Modelling of Robots

Wisama Khalil, Aravindkumar Vijayalingam, Bogdan Khomutenko, Izzatbek Mukhanov, Philippe Lemoine, Gaël Ecorchard

OpenSYMORO: An open-source software package for Symbolic Modelling of Robots

Wisama Khalil, Aravindkumar Vijayalingam, Bogdan Khomutenko, Izzatbek Mukhanov,
Philippe Lemoine, Gaël Ecorchard
École Centrale de Nantes, IRCCyN UMR CNRS 6597
1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France

***Abstract*— This paper presents OpenSYMORO, an open-source software package for symbolic modelling of robots. This software package is based on previous work detailed in [1]. However, the package in [1] was developed using Wolfram Mathematica and hence required Mathematica license for use. OpenSYMORO is mainly developed using the `Python` programming language and the source code will be publicly available. The new version provides support to model robots with flexible joints, floating base and wheeled mobile robots. This is in addition to supporting serial, tree structure and closed-loop robots. A visualisation tool to view the structure of the robot is also included.**

I. Introduction

Symbolic modelling of robots is essential for analysis and design of robots and their controllers. Also symbolic modelling results in lower execution time and is more convenient for analysis. For instance, solution to the inverse geometric problem can be obtained by the use of differential model to compute an iterative numerical solution [2]. However, this results in long execution time and may not be practical. On the other hand the solution for the same inverse geometric problem can be obtained using closed-form symbolic models that can deliver all the possible solutions with a lower and predictable computation time.

The purpose of OpenSYMORO (henceforth referred to as SYMORO) software package is to compute the symbolic models of a robot that are needed for design, identification, control and simulation. Previously, such a package was presented in [1]. However, the package in [1] was developed using Wolfram Mathematica[1] and hence required Mathematica license for use. The new version of the package is developed using `Python` programming language and other third-party libraries – `SymPy`[2] and `NumPy`[3] that are available for use without any licensing requirements. As a result, the SYMORO package is available as an open-source project and the source code will be publicly available. Consequently the robotics community over the world can use this package and can take part in the development of new modules.

Additionally, the new version of SYMORO software package provides support to model robots with flexible joints, floating base and wheeled mobile robots. The new version also provides a visualisation tool to view the structure of the robot and the output of inverse geometric models can be obtained in a format that is compatible with `Matlab` and `Python` such that the generated equations can be directly used by these systems.

In the next section of this paper, the architecture and the different components of the software package are discussed. A short presentation on how to define a robot in SYMORO follows. Thereafter, the algorithms used to model robots with flexible joints, floating base and wheeled mobile robots are detailed.

II. Software Package Description

The SYMORO software package is made up of five different sub-packages. These sub-packages are – `symoroutils`, `pysymoro`, `symoroui`, `symoroviz` and `symorooptim`. These sub-packages may depend on another sub-package/s.

A. `symoroutils`

The `symoroutils` package contains the utility modules for mathematical functions and file handling. The file handling modules include the functions to perform parsing of files that are used to load robot data into the environment.

B. `pysymoro`

The algorithms that are used to generate the different models that can be obtained using SYMORO are contained in this package. The data structures that are used to store the robot data and models are also present in this package. This package in turn makes use of `SymPy` and `NumPy` to perform the symbolic and numeric computations.

C. `symoroui`

The `symoroui` package provides the graphical user interface for the SYMORO software. The user interface created using `wxPython`[4] is shown in Fig 3. This package permits to define the parameters that are needed in the calculation of different models. This interface is also used to execute any desired model.

[1] http://www.wolfram.com/mathematica/
[2] http://www.sympy.org/
[3] http://www.numpy.org/
[4] http://www.wxpython.org/

D. `symorooptim`

The purpose of this package is to generate optimised models in symbolic form. These optimised models reduce the total number of operations (addition and multiplication) required to compute the respective models by the use of intermediate variables. The output of the optimised models can be obtained in a format that is compatible with `C`, `Python`, `Matlab`, such that the generated model can be directly used by these systems.

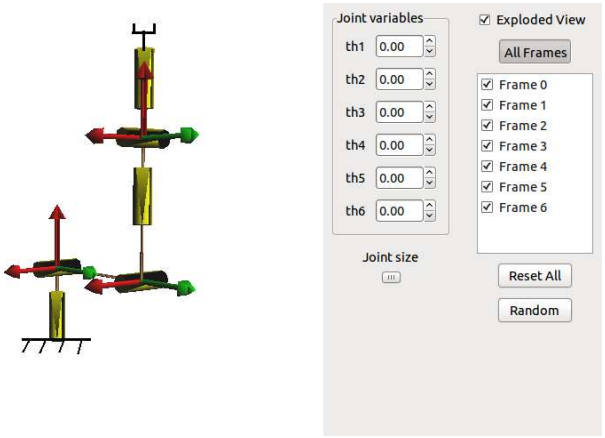

Fig. 1: Representation of Staubli RX-90 robot

E. `symoroviz`

This package developed using the Python port of `OpenGL` helps with visualisation of robots. As can be seen from Fig 1, in addition to drawing the joints of a robot this package also draws the frames corresponding to each joint according to the modified Denavit and Hartenberg method as proposed by Khalil and Kleinfinger [3]. This permits to verify if the user has properly defined the different frames attached to each of the links.

III. Robot Description and Menu Options

A new robot can be created by selecting `File -> New` and then specifying the basic properties of the robot (see Fig 2). At this step, number of links, number of joints and the robot structure type are specified. The robot structure can be serial, tree-structured, closed-loop robots with fixed base, wheeled mobile robot or floating base. Following this step, the parameters in Fig 3 are specified either symbolically or numerically.

A. Geometric parameters

The geometric parameters define the kinematic structure of the robot, joint types and the location of link frames with respect to its antecedent. As stated above, Khalil and Kleinfinger [3] notation is used to describe the geometric parameters. The coordinate frame `j` is assigned fixed with respect to link `j` such that z_j is along the axis of joint `j` and x_j is normal between z_j and one of the following axis. The geometry of the robot is described by the following parameters:

- γ_j, b_j, α_j, d_j, θ_j and r_j to define a frame `j` with respect to its antecedent frame $a(j)$. The terms γ_j and b_j are zero in a serial robot. The joint variable q_j is either θ_j or r_j depending on the joint type.
- σ_j defines the joint type – 0 for revolute, 1 for prismatic and 2 for a frame that has a constant relationship with it's antecedent. For revolute and prismatic joints, $\bar{\sigma}_j = 1 - \sigma_j$.
- μ_j indicates if a joint is active (actuated) or passive (non-actuated) – 1 for active and 0 for passive. For open-loop robots – serial and tree structure all the joints are supposed actuated.
- $a(j)$ to indicate the antecedent frame. Here "0" indicates the base.

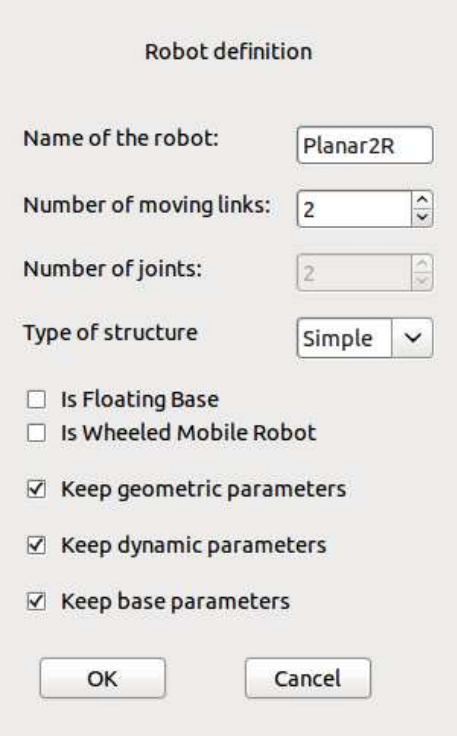

Fig. 2: Robot description dialog window

The geometric parameters are used to calculate the homogeneous transformation matrix between a frame `j` with respect to its antecedent `i = a(j)`. The transformation matrix is abbreviated by,

$$ {}^{i}\mathbf{T}_j = \begin{bmatrix} {}^{i}\mathbf{R}_j & {}^{i}\mathbf{P}_j \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{1} $$

where ${}^{i}\mathbf{R}_j$ is the (3x3) rotation matrix and ${}^{i}\mathbf{P}_j$ is the translation vector of frame `j` with respect to `i`.

B. Dynamic parameters and external forces

The dynamic parameters include the inertial and friction parameters of each link. They are:

- XX_j, XY_j, XZ_j, YY_j, YZ_j, ZZ_j are the elements of the inertia matrix $\bar{I}_{Oj}$ that defines the inertia of link `j` about frame `j`.

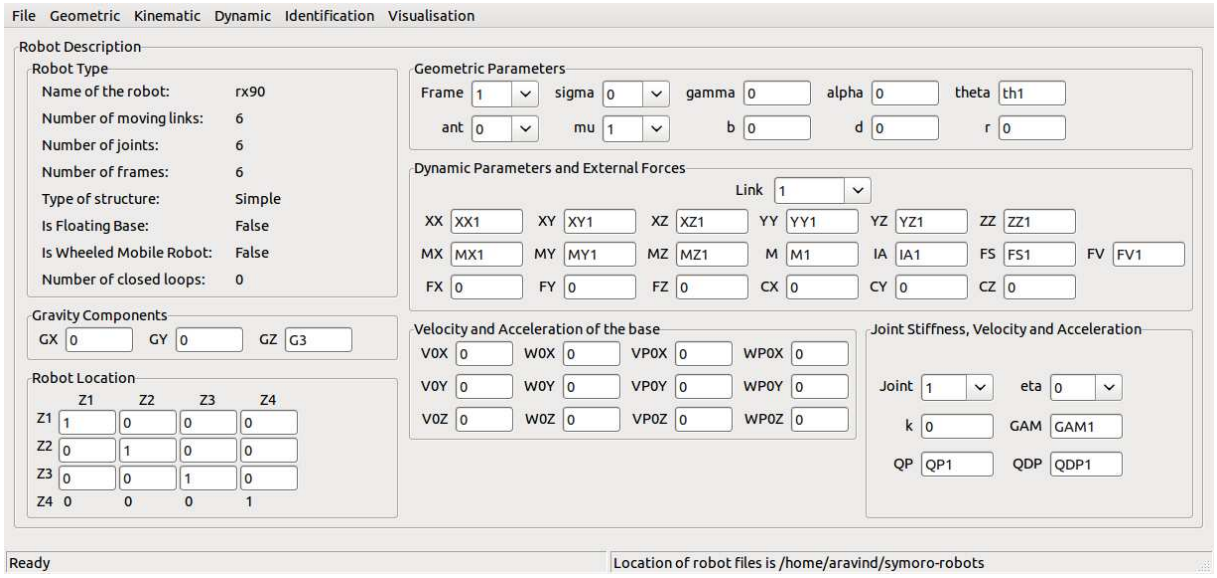

Fig. 3: Main window of SYMORO

- MX_j, MY_j, MZ_j are the elements defining the first moments of link j with respect to frame j and $\mathbf{MS}_j = [MX_j \quad MY_j \quad MZ_j]^T$.
- M_j is the mass of link j.
- Ia_j is the rotor inertia of actuator for joint j.
- Fc_j and Fv_j are the parameters of Coulomb friction and viscous friction respectively at joint j.
- f_{xj}, f_{yj}, f_{zj} are the forces $\mathbf{f}_{ej}$ exerted by the link j on the environment.
- m_{xj}, m_{yj}, m_{zj} are the moments $\mathbf{m}_{ej}$ exerted by the link j on the environment.

C. Miscellaneous parameters

The following parameters can also be defined for a robot:

- Location of the base of the robot with respect to a general fixed frame (matrix Z).
- Vector to indicate the acceleration due to gravity, $\mathbf{g} = [g_x \quad g_y \quad g_z]^T$.
- Velocity ($\mathbf{v}_0$, $\boldsymbol{\omega}_0$) and acceleration ($\dot{\mathbf{v}}_0$, $\dot{\boldsymbol{\omega}}_0$) of base of the robot.
- η_j defines if the joint is rigid or flexible – 0 for rigid and 1 for flexible.
- k_j to specify the joint stiffness in the case of flexible joints.
- Joint velocities (QP_j), accelerations (QDP_j) and torques (GAM_j). The user can specify special values such as 0 or 1 to obtain a customised model.

D. Menu Options

Once the robot is defined by the parameters mentioned above, the user can obtain the results for the options listed in the menu. The list of available options in SYMORO are shown in Fig 4. The user also has the possibility to run any of these options without using this interface.

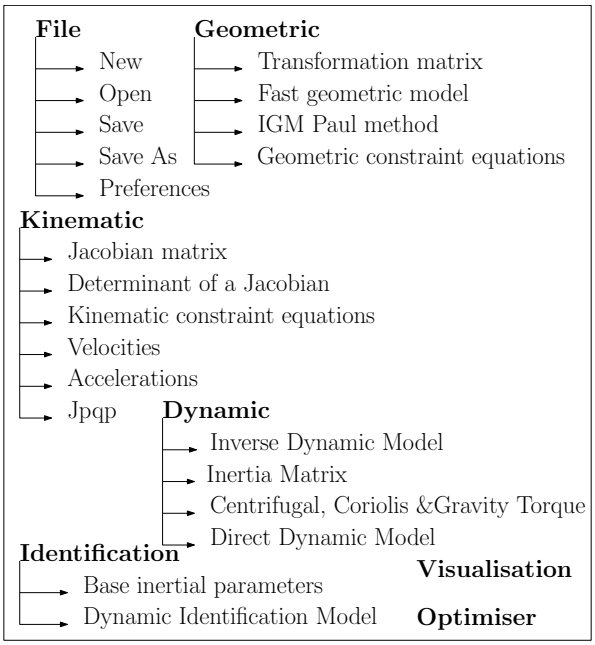

Fig. 4: Menu options available in SYMORO

IV. Algorithms

A brief description of the algorithms used in the SYMORO (more specifically pysymoro) package are presented in this section. It should be noted, only the algorithms that aren't described in [1] are presented here. This is because OpenSYMORO uses the same algorithms as those presented in SYMORO+ [1] with the modelling of robots with flexible joints, floating base and wheeled mobile robots as the only additions to those.

A. Dynamic modelling of robots

The general form of the dynamic model of robots obtained using Lagrange method [2], [4], [5] is,

$$\mathbf{\Gamma} = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) \tag{2}$$

where, $\mathbf{A}$ is the inertia matrix as a function of joint positions, $\mathbf{q}$ and $\mathbf{H}$ is the Coriolis, centrifugal and gravity torques as a function of joint velocities, $\dot{\mathbf{q}}$ and joint positions, $\mathbf{q}$. The joint acceleration vector is indicated by $\ddot{\mathbf{q}}$. This is known as the inverse dynamic model as torque, $\mathbf{\Gamma}$ is obtained as a function of $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. Solving to obtain $\ddot{\mathbf{q}}$ as a function of $(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{\Gamma})$ is the direct dynamic model and is given by,

$$\ddot{\mathbf{q}} = \mathbf{A}^{-1}\left[\mathbf{\Gamma} - \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})\right] \tag{3}$$

For rigid robots with fixed base, the inverse dynamic model is obtained efficiently using the recursive Newton-Euler algorithm. Recursive method is also developed for the direct dynamic models which do not need to calculate the inertia matrix [6]. For closed-loop robots and wheeled mobile robots the inertia matrix must be calculated and this is developed by the algorithm proposed in [7].

B. Recursive Newton-Euler method for tree-structure robots with fixed base

This algorithm which is based on [8] constitutes the cornerstone on which all the other algorithms are built upon. The main characteristic of this algorithm is the recursivity leading to reduced number of operations.

This algorithm involves a forward recursion and a backward recursion. During the forward recursion link velocities, link accelerations and dynamic wrench on each link are computed from link 1 to link n. The backward recursion process computes the reaction wrenches on the links and the joint torques from link n to link 0 (base). This algorithm can be denoted by,

$$\mathbf{\Gamma} = \mathrm{NE}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{f}_e, \mathbf{m}_e) \tag{4}$$

where, $\mathbf{f}_e$ and $\mathbf{m}_e$ are the forces and moments respectively exerted by the links on the environment.

According to [2], the forward recursive equations for $j = 1, \ldots, n$ with $i = a(j)$ are:

$${}^j\boldsymbol{\omega}_j = \underbrace{{}^j\mathbf{R}_i\ {}^i\boldsymbol{\omega}_i}_{{}^j\boldsymbol{\omega}_i} + \bar{\sigma}_j\ \dot{q}_j\ {}^j\bar{\mathbf{a}}_j \tag{5}$$

$${}^j\mathbf{S}_i = \begin{bmatrix} {}^j\mathbf{R}_i & -\,{}^j\mathbf{R}_i\ {}^i\hat{\mathbf{P}}_j \\ \mathbf{0}_{3\times3} & {}^j\mathbf{R}_i \end{bmatrix} \tag{6}$$

$$\begin{aligned} {}^j\dot{\mathbf{V}}_j = {}& {}^j\mathbf{S}_i\ {}^i\dot{\mathbf{V}}_i + \ddot{q}_j\ {}^j\mathbf{a}_j \\ & + \begin{bmatrix} {}^j\mathbf{R}_i\left[{}^i\boldsymbol{\omega}_i \times \left({}^i\boldsymbol{\omega}_i \times {}^i\mathbf{P}_j\right)\right] + 2\sigma_j\left({}^j\boldsymbol{\omega}_i \times \dot{q}_j\ {}^j\bar{\mathbf{a}}_j\right) \\ \bar{\sigma}_j\left({}^j\boldsymbol{\omega}_i \times \dot{q}_j\ {}^j\bar{\mathbf{a}}_j\right) \end{bmatrix} \end{aligned} \tag{7}$$

$${}^j\mathbf{F}_{tj} = \begin{bmatrix} {}^j\mathbf{f}_{tj} \\ {}^j\mathbf{m}_{tj} \end{bmatrix} = {}^j\mathbf{I}_{Oj}\ {}^j\dot{\mathbf{V}}_j + \begin{bmatrix} {}^j\boldsymbol{\omega}_j \times \left({}^j\boldsymbol{\omega}_j \times {}^j\mathbf{MS}_j\right) \\ {}^j\boldsymbol{\omega}_j \times \left({}^j\bar{I}_{Oj}\ {}^j\boldsymbol{\omega}_j\right) \end{bmatrix} \tag{8}$$

where, ${}^j\bar{\mathbf{a}}_j = [0\ 0\ 1]^T$ is the unit vector along $\mathbf{z}_j$ axis which is the axis of joint $\mathtt{j}$. $(\hat{*})$ is the skew-symmetric matrix of the $(*)$ (3x1) vector. Here ${}^j\mathbf{a}_j = [0\ 0\ \sigma_j\ 0\ 0\ \bar{\sigma}_j]^T$, ${}^j\mathbf{S}_i$ is the screw transformation matrix, ${}^j\mathbf{I}_{Oj} = \begin{bmatrix} M_j I_3 & {}^j\widehat{\mathbf{MS}}_j^T \\ {}^j\widehat{\mathbf{MS}}_j & {}^j\bar{I}_{Oj} \end{bmatrix}$ is the spatial (6x6) inertia matrix and ${}^j\dot{\mathbf{V}}_j = \begin{bmatrix} {}^j\dot{\mathbf{v}}_j^T & {}^j\dot{\boldsymbol{\omega}}_j^T \end{bmatrix}^T$.

The above equations (from (5) to (8)) are initialised with ${}^0\boldsymbol{\omega}_0 = 0$, ${}^0\dot{\boldsymbol{\omega}}_0 = 0$ and ${}^0\dot{\mathbf{v}}_0 = -{}^0\mathbf{g}$ where, $\mathbf{g}$ is the gravity vector.

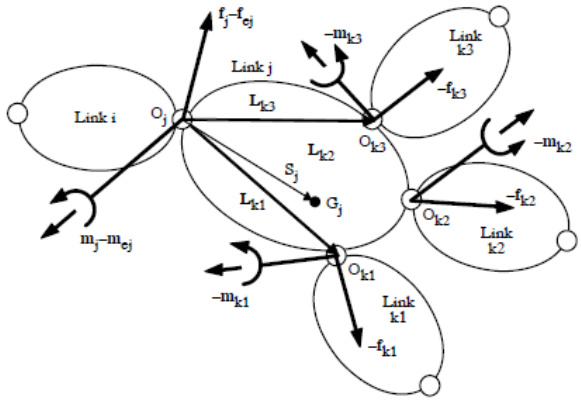

Fig. 5: Forces and moments acting on link $\mathtt{j}$.

Likewise, the backward recursive equations for $j = n, \ldots, 1$ with $i = a(j)$ are:

$${}^j\mathbf{F}_j = \begin{bmatrix} {}^j\mathbf{f}_j \\ {}^j\mathbf{m}_j \end{bmatrix} = {}^j\mathbf{F}_{tj} + {}^j\mathbf{F}_{ej} \tag{9}$$

$${}^i\mathbf{F}_{ei} = {}^i\mathbf{F}_{ei} + {}^j\mathbf{S}_i^T\ {}^j\mathbf{F}_j \tag{10}$$

$$\Gamma_j = {}^j\mathbf{F}_j^T\ {}^j\mathbf{a}_j + F_{cj}\ sign(\dot{q}_j) + F_{vj}\ \dot{q}_j + Ia_j\ \ddot{q}_j \tag{11}$$

Here ${}^j\mathbf{F}_{ej} = \begin{bmatrix} {}^j\mathbf{f}_{ej}^T & {}^j\mathbf{m}_{ej}^T \end{bmatrix}^T$. Please see Section III for the definitions on some of the symbols used in (5) – (11).

C. Dynamic modelling of robots with flexible joints

For a robot with flexible joints the general form of the dynamic model [9] is,

$$\underbrace{\begin{bmatrix} \mathbf{\Gamma}_r \\ \mathbf{\Gamma}_f \end{bmatrix}}_{\mathbf{\Gamma}} = \underbrace{\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \ddot{\mathbf{q}}_r \\ \ddot{\mathbf{q}}_f \end{bmatrix}}_{\ddot{\mathbf{q}}} + \underbrace{\begin{bmatrix} \mathbf{H}_r \\ \mathbf{H}_f \end{bmatrix}}_{\mathbf{H}} \tag{12}$$

where, $\mathbf{\Gamma}_r$ is the vector of rigid joint torques and $\mathbf{\Gamma}_f$ is the vector of flexible joint torques. If a joint, $\mathtt{j}$ is flexible then,

$$\Gamma_{fj} = -k_j \underbrace{(q_j - q_{rj})}_{\Delta q_j} \tag{13}$$

where, k_j is the stiffness of the flexible joint and q_{rj} is the reference joint position corresponding to zero elasticity force. From (12) for the inverse dynamic model we get,

$$\ddot{\mathbf{q}}_f = \mathbf{A}_{22}^{-1}\left(\mathbf{\Gamma}_f - \mathbf{H}_f - \mathbf{A}_{12}^T\ \ddot{\mathbf{q}}_r\right) \tag{14}$$

$$\mathbf{\Gamma}_r = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_r \\ \ddot{\mathbf{q}}_f \end{bmatrix} + \mathbf{H}_r \tag{15}$$

As stated in [9], [10], a three step recursive algorithm can be used for the calculation of the inverse dynamic model to calculate $\ddot{\mathbf{q}}_f$ and $\mathbf{\Gamma}_r$ as a function of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}_r$ or the direct dynamic model to calculate $\ddot{\mathbf{q}}_f$ and $\ddot{\mathbf{q}}_r$ in terms of $\mathbf{\Gamma}_r$, $\mathbf{q}$, $\dot{\mathbf{q}}$. In this paper we present only the inverse dynamic model but both algorithms are developed in SYMORO.

(i) first forward recursion for $j = 1, \ldots, n$ computes the screw transformation matrices ${}^j\mathbf{S}_i$, the link velocities ${}^j\mathbf{V}_j$, the gyroscopic accelerations ${}^j\boldsymbol{\gamma}_j$ and the wrench ${}^j\boldsymbol{\beta}_j$ which is a combination of the external forces, Coriolis forces and centrifugal forces.

$$ {}^j\mathbf{V}_j = {}^j\mathbf{S}_i \ {}^i\mathbf{V}_i + \dot{\mathbf{q}}_j \ {}^j\mathbf{a}_j \tag{16} $$

$$ {}^j\boldsymbol{\gamma}_j = \begin{bmatrix} {}^j\mathbf{R}_i \left[{}^i\boldsymbol{\omega}_i \times \left({}^i\boldsymbol{\omega}_i \times {}^i\mathbf{P}_j \right) \right] + 2\sigma_j \left({}^j\boldsymbol{\omega}_i \times \dot{q}_j \ {}^j\bar{\mathbf{a}}_j \right) \\ \bar{\sigma}_j \left({}^j\boldsymbol{\omega}_i \times \dot{q}_j \ {}^j\bar{\mathbf{a}}_j \right) \end{bmatrix} \tag{17} $$

$$ {}^j\boldsymbol{\beta}_j = - \ {}^j\mathbf{F}_{ej} - \begin{bmatrix} {}^j\boldsymbol{\omega}_j \times \left({}^j\boldsymbol{\omega}_j \times {}^j\mathbf{MS}_j \right) \\ {}^j\boldsymbol{\omega}_j \times \left({}^j\bar{I}_{Oj} \ {}^j\boldsymbol{\omega}_j \right) \end{bmatrix} \tag{18} $$

with ${}^j\mathbf{V}_j = \begin{bmatrix} {}^j\mathbf{v}_j^T & {}^j\boldsymbol{\omega}_j^T \end{bmatrix}^T$.

(ii) backward recursion for $j = n, \ldots, 1$ calculates the matrices giving the elastic accelerations $\ddot{\mathbf{q}}_j$ and ${}^j\mathbf{F}_j$ as a function of ${}^i\dot{\mathbf{V}}_i$. When joint `j` is flexible,

$$ H_j = {}^j\mathbf{a}_j{}^T \ {}^j\mathbf{I}_j^* \ {}^j\mathbf{a}_j \tag{19} $$

$$ {}^j\mathbf{K}_j = {}^j\mathbf{I}_j^* \ - {}^j\mathbf{I}_j^* \ {}^j\mathbf{a}_j \ H_j^{-1} \ {}^j\mathbf{a}_j{}^T \ {}^j\mathbf{I}_j^* \tag{20} $$

$$ {}^j\boldsymbol{\alpha}_j = {}^j\mathbf{K}_j \ {}^j\boldsymbol{\gamma}_j + {}^j\mathbf{I}_j{}^* \ {}^j\mathbf{a}_j \ H_j^{-1} \left(\Gamma_{fj} + {}^j\mathbf{a}_j{}^T \ {}^j\boldsymbol{\beta}_j^* \right) - {}^j\boldsymbol{\beta}_j^* \tag{21} $$

When joint `j` is rigid ($\ddot{q}_j$ is known),

$$ {}^j\mathbf{K}_j = {}^j\mathbf{I}_j^* \tag{22} $$

$$ {}^j\boldsymbol{\alpha}_j = {}^j\mathbf{K}_j \ {}^j\boldsymbol{\gamma}_j + {}^j\mathbf{I}_j{}^* \ {}^j\mathbf{a}_j \ \ddot{q}_j - {}^j\boldsymbol{\beta}_j^* \tag{23} $$

If $a(j) \neq 0$, compute:

$$ {}^i\mathbf{I}_i^* = {}^i\mathbf{I}_i^* + {}^j\mathbf{S}_i{}^T \ {}^j\mathbf{K}_j \ {}^j\mathbf{S}_i \tag{24} $$

$$ {}^i\boldsymbol{\beta}_i^* = {}^i\boldsymbol{\beta}_i^* - {}^j\mathbf{S}_i{}^T \ {}^j\boldsymbol{\alpha}_j \tag{25} $$

The equations (19) – (25) are initialised by ${}^j\mathbf{I}_j^* = {}^j\mathbf{I}_{Oj}$ and ${}^j\boldsymbol{\beta}_j^* = {}^j\boldsymbol{\beta}_j$.

(iii) second forward recursion for $j = 1, \ldots, n$ computes $\ddot{q}_j$ for the flexible joints and the joint torques for the rigid joints. This recursion is initialised by ${}^0\dot{\mathbf{V}}_0 = \begin{bmatrix} {}^0\mathbf{g}^T & \mathbf{0}_{3\times1}^T \end{bmatrix}^T$ to take gravity into account.

$$ {}^j\mathbf{F}_j = {}^j\mathbf{K}_j \ {}^j\mathbf{S}_i \ {}^i\dot{\mathbf{V}}_i + {}^j\boldsymbol{\alpha}_j \tag{26} $$

When joint `j` is flexible,

$$ \ddot{q}_j = H_j^{-1} \left[\Gamma_{fj} - {}^j\mathbf{a}_j^T \ {}^j\mathbf{I}_j^* \left({}^j\mathbf{S}_i {}^i\dot{\mathbf{V}}_i + {}^j\boldsymbol{\gamma}_j \right) + {}^j\mathbf{a}_j{}^T \ {}^j\boldsymbol{\beta}_j^* \right] \tag{27} $$

$$ {}^j\dot{\mathbf{V}}_j = {}^j\mathbf{S}_i \ {}^i\dot{\mathbf{V}}_i \ + {}^j\mathbf{a}_j \ \ddot{q}_j + {}^j\boldsymbol{\gamma}_j \tag{28} $$

When joint `j` is rigid,

$$ \mathbf{\Gamma}_j = {}^j\mathbf{F}_j^T \ {}^j\mathbf{a}_j + F_{cj} \ sign(\dot{q}_j) + F_{vj} \ \dot{q}_j + Ia_j \ \ddot{q}_j \tag{29} $$

D. Dynamic modelling of robots with floating base

This category includes a large variety of systems such as: humanoid robots, walking robots, eel-like robots [11], snake-like robots [11], spatial vehicles, offshore structures and flying robots. The difference between all of these systems will be in the calculation of the interaction forces. For these robots there is neither geometric nor kinematic relationships between the Cartesian motion of the base and the joint variables and the dynamic model must be used to simulate these systems. In this case, the acceleration of the base has to be determined while computing the inverse and direct dynamic models.

The base frame `0` is defined with respect to a fixed reference frame `W` and the relationship between these two frames are known at time `t = 0`. The transformation matrix ${}^W\mathbf{T}_0$ indicated by `Z`, is updated over time by integrating the base acceleration ${}^W\dot{\mathbf{V}}_b$. The base velocity is indicated by $\mathbf{V}_b$.

The general form of the dynamic model of a robot with floating base [9] is,

$$ \begin{bmatrix} \mathbf{0}_{6\times1} \\ \mathbf{\Gamma} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} {}^0\dot{\mathbf{V}}_0 \\ \ddot{\mathbf{q}} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix}}_{\mathbf{H}} \tag{30} $$

where,

- $\mathbf{A}_{11}$ is the (6x6) inertia matrix of the composite link `0`, which is composed of the inertia of all links referred to frame `0` (the base).
- $\mathbf{A}_{22}$ is the (nxn) inertia matrix of the other links when the head is fixed.
- $\mathbf{A}_{12}$ is the (6xn) coupled inertia matrix of the joints and the base. It reflects the effect of joint accelerations of the base motion, and the dual effect of base accelerations on the joint motions.
- $\mathbf{H}_1$ is the Coriolis, centrifugal, gravity and external forces on the base.
- $\mathbf{H}_2$ is the Coriolis, centrifugal, gravity and external forces on the links 1, ..., n.

Similar to Section IV-C, a three step recursion - a forward, a backward and then a forward again can be used for the computation of ${}^0\dot{\mathbf{V}}_0$ and $\mathbf{\Gamma}$.

(i) first forward recursion for $j = 1, \ldots, n$ computes ${}^j\mathbf{S}_i$, ${}^j\mathbf{V}_j$, ${}^j\boldsymbol{\gamma}_j$ and ${}^j\boldsymbol{\beta}_j$ (see (16) – (18)). Additionally, relative accelerations ${}^j\boldsymbol{\zeta}_j$ is calculated as,

$$ {}^j\boldsymbol{\zeta}_j = {}^j\boldsymbol{\gamma}_j + \ddot{q}_j \ {}^j\mathbf{a}_j \tag{31} $$

(ii) backward recursion for $j = n, \ldots, 0$ computes the base acceleration using the inertial parameters of the composite link `0`, where the composite link `j` consists of the links articulated on it.

$$ {}^j\mathbf{F}_j = {}^j\mathbf{I}_j^c \ {}^j\dot{\mathbf{V}}_j - {}^j\boldsymbol{\beta}_j^c \tag{32} $$

with, inertial matrix of the composite link `j`,

$$ {}^i\mathbf{I}_i^c = {}^i\mathbf{I}_i^c + {}^j\mathbf{S}_i^T \ {}^j\mathbf{I}_j^c \ {}^j\mathbf{S}_i \tag{33} $$

and

$$ {}^{i}\boldsymbol{\beta}_i^c = {}^{i}\boldsymbol{\beta}_i^c + {}^{j}\mathbf{S}_i^T \ {}^{j}\boldsymbol{\beta}_j^c - {}^{j}\mathbf{S}_i^T \ {}^{j}\mathbf{I}_j^c \ {}^{j}\boldsymbol{\zeta}_j \tag{34} $$

Since ${}^{0}\mathbf{F}_0 = 0$, for j = 0 from (32) we get,

$$ {}^{0}\dot{\mathbf{V}}_0 = \begin{bmatrix} {}^{0}\dot{\mathbf{v}}_0 \\ {}^{0}\dot{\boldsymbol{\omega}}_0 \end{bmatrix} = \left({}^{0}\mathbf{I}_0^c\right)^{-1} \ {}^{0}\boldsymbol{\beta}_0^c \tag{35} $$

This backward recursion step is initialised by ${}^{j}\mathbf{I}_j^c = {}^{j}\mathbf{I}_{Oj}$ and ${}^{j}\boldsymbol{\beta}_j^c = {}^{j}\boldsymbol{\beta}_j$. When $j = 0$, ${}^{0}\boldsymbol{\beta}_0$ is first computed using (18).

(iii) second forward recursion for $j = 1, \ldots, n$ calculates the wrench ${}^{j}\mathbf{F}_j$ and the joint torques for $j = 1, \ldots, n$.

$$ {}^{j}\dot{\mathbf{V}}_j = {}^{j}\mathbf{S}_i \ {}^{i}\dot{\mathbf{V}}_i + {}^{j}\boldsymbol{\zeta}_j \tag{36} $$

$$ {}^{j}\mathbf{F}_j = \begin{bmatrix} {}^{j}\mathbf{f}_j \\ {}^{j}\mathbf{m}_j \end{bmatrix} = {}^{j}\mathbf{I}_j^c \ {}^{j}\dot{\mathbf{V}}_j - {}^{j}\boldsymbol{\beta}_j^c \tag{37} $$

$$ \Gamma_j = {}^{j}\mathbf{F}_j^T \ {}^{j}\mathbf{a}_j + F_{cj} \ sign(\dot{q}_j) + F_{vj} \ \dot{q}_j + Ia_j \ \ddot{q}_j \tag{38} $$

To simulate the Cartesian motion of the base, the base acceleration needs to be computed. From (35), taking gravity into account we get,

$$ {}^{0}\dot{\mathbf{V}}_b = {}^{0}\dot{\mathbf{V}}_0 + \begin{bmatrix} {}^{0}\mathbf{g} \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{39} $$

Here it should be noted that the base acceleration got from (39), ${}^{0}\dot{\mathbf{V}}_b \neq \dfrac{d}{dt}{}^{0}\mathbf{V}_0$. Thus to integrate the acceleration, the base acceleration with respect to fixed frame `W` must be calculated.

$$ {}^{W}\dot{\mathbf{V}}_b = \begin{bmatrix} {}^{W}\mathbf{R}_0 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & {}^{W}\mathbf{R}_0 \end{bmatrix} {}^{0}\dot{\mathbf{V}}_b \tag{40} $$

By integrating (40) we get ${}^{W}\mathbf{V}_b$. Now the base velocity, ${}^{0}\mathbf{V}_b$ is,

$$ {}^{0}\mathbf{V}_0 = {}^{0}\mathbf{V}_b = \begin{bmatrix} {}^{0}\mathbf{R}_W & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & {}^{0}\mathbf{R}_W \end{bmatrix} {}^{W}\mathbf{V}_b \tag{41} $$

E. Dynamic modelling of wheeled mobile robots

This category of robots consist of a tree-structure robot on a mobile base representing link 0 and classical wheels. It can be described as shown in Section IV-D. However, in this case the system is characterised by having kinematic non-holonomic relationship between the Cartesian variables of link `0` and the joint variables. Hence to study the Cartesian evolution of these robots the computation of the dynamic model is not a must. Whereas in (Section IV-D) the dynamic model is essential to obtain the Cartesian motion of the system.

The classical wheels are fixed, steering and castor. These wheels have the rotational variable φ_j. The steering and castor wheels also have β_j the orientation variable. Hence, the vector of configuration variables is composed of φ_j, β_j and $\boldsymbol{\xi}$ - the posture of cart position and orientation. For the sake of simplicity, the motion of the mobile base is considered as planar and thus $\boldsymbol{\xi} = [x \quad y \quad \theta]^T$. Detailed discussion on wheeled mobile robots is presented in [12].

These wheeled mobile robots contains non-holonomic constraints between the configuration variables. The constraint equations are,

$$ \mathbf{W} \ \dot{\mathbf{q}} = 0 \quad \& \quad \dot{\mathbf{q}} = \mathbf{G} \ \dot{\mathbf{q}}_a \tag{42} $$

where, $\mathbf{q} = \left[\boldsymbol{\xi}^T \ \boldsymbol{\beta}^T \ \boldsymbol{\varphi}^T\right]^T$, $\mathbf{q}_a$ is the actuated variables of $\mathbf{q}$.

The difference between floating and wheeled mobile robots is that the coupling between the configuration variables is kinematically defined using the non-slipping and non-sliding constraints of the wheels with respect to the ground. The inverse dynamic model is obtained at first using the equations of tree-structure to obtain $\boldsymbol{\Gamma}$. Then (43) is applied to project $\boldsymbol{\Gamma}$ on the actuated joints.

$$ \boldsymbol{\Gamma}_a = \left(\frac{\partial \dot{\mathbf{q}}}{\partial \dot{\mathbf{q}}_a}\right)^T \boldsymbol{\Gamma} = \mathbf{G}^T \boldsymbol{\Gamma} \tag{43} $$

where, $\boldsymbol{\Gamma}$ is the joint torques calculated by the method in Section IV-B or Section IV-C and $\boldsymbol{\Gamma}_a$ is the actuator torques.

V. Conclusion

This paper presents OpenSYMORO, an open-source software package for symbolic modelling of robots. The features of the software package allow the generation of robot models that are required for simulation, identification, control and design. An overview of the algorithms that are used to model the robots with flexible joints, floating base and wheeled mobile robots are also presented. Further development of the package concerns the support for modelling robots with flexible links [13]. The capability of accessing the features of the package through a web interface is also considered for future versions.

References

[1] W. Khalil and D. Creusot, "Symoro+: a system for the symbolic modelling of robots," *Robotica*, vol. 15, pp. 153–161, 1997.

[2] W. Khalil and E. Dombre, *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[3] W. Khalil and J. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3. IEEE, 1986, pp. 1174–1179.

[4] J. J. Craig, *Introduction to robotics: mechanics and control*. Addison-Wesley, 1986.

[5] J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Springer, 2007.

[6] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *The International Journal of Robotics Research*, vol. 2, no. 1, pp. 13–30, 1983.

[7] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 3, pp. 205–211, 1982.

[8] J. Y. Luh, M. W. Walker, and R. P. Paul, "On-line computational scheme for mechanical manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, pp. 69–76, 1980.

[9] W. Khalil, "Dynamic modeling of robots using recursive newton-euler techniques," *ICINCO2010*, 2010.

[10] W. Khalil and M. Gautier, "Modeling of mechanical systems with lumped elasticity," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 3964–3969.

[11] F. Boyer, M. Porez, and W. Khalil, "Macro-continuous computed torque algorithm for a three-dimensional eel-like robot," *IEEE Transactions on Robotics*, pp. 563–775, 2006.

[12] G. Campion and W. Chung, "Wheeled robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, ch. 17, pp. 391–410.

[13] F. Boyer and W. Khalil, "An efficient calculation of flexible manipulator inverse dynamics," *The International Journal of Robotics Research*, vol. 17, no. 3, pp. 282–293, 1998.