**Noname manuscript No.**
(will be inserted by the editor)

# Multibody Model of the Human-Inspired Robot CHARMIE

**Fernando Gonçalves** [1,2] ·
**Tiago Ribeiro**[3] · **A. Fernando Ribeiro**[3] ·
**Gil Lopes**[4] · **Paulo Flores**[1,2]

Received: date / Accepted: date

**Abstract** The rapid ageing of the worldwide population raises pressing concerns related to ensuring proper healthcare and quality of life for older adults. A human-like mobile domestic robot, named CHARMIE, is being produced to aid in these situations by performing household chores, thus increasing the autonomy of persons with mobility limitations. The present work provides a valuable contribution to the development of CHARMIE by building a simulation environment that computes the system's main dynamics. The obtained environment is used to evaluate the quality of the robot's control system, to perform its structural optimization and to allow a proper selection of actuators. The system is tackled as a kinematic tree that starts on the robot's base and then splits into three branches at the torso, the left arm, the right arm, and the head. The multibody model solves the forward kinematics and inverse dynamics of the main mechanisms by employing two recursive algorithms centred around the Newton-Euler formulation. A novel, modular, and efficient seven-step methodology was created to implement these two algorithms and program a simulator from start to finish. These seven steps include studying the system's configuration, converting its properties into software inputs, and computing the phenomena that can't be automatically addressed by the two recursive formulations. The presented methodology was fully validated by comparing its results to those obtained from a commercial software; the two models produced identical results.

**Keywords** Multibody dynamics · Mobile manipulator robot · Newton-Euler formulation

✉ F. Gonçalves id8699@alunos.uminho.pt
[1] CMEMS-UMinho, Department of Mechanical Engineering, University of Minho, Guimarães 4804-533, Portugal
[2] LABBELS – Associate Laboratory, Braga/Guimarães, Portugal
[3] Centro ALGORITMI, Department of Industrial Electronics, University of Minho, Guimarães 4804-533, Portugal
[4] INESC TEC, Universidade da Maia - UMAIA, 4475-690 Maia, Portugal

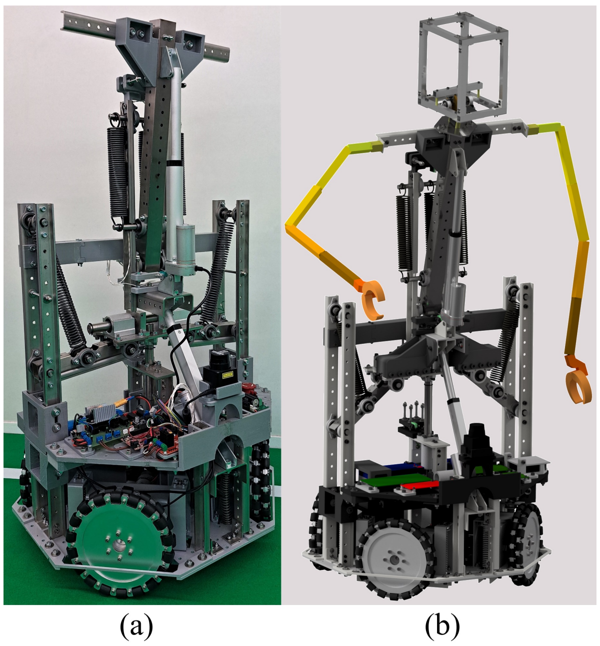

**Fig. 1** Physical prototype (a) and CAD model (b) of the CHARMIE robot.

## 1 Introduction

One of the most pressing concerns of modern societies is the rapid ageing of the worldwide population [1]. With the recent developments in robotic technologies, robots can play a growing role in ensuring the quality of life of older adults by increasing their autonomy via, for example, performing daily chores [2]. CHARMIE, a novel human-inspired mobile manipulator (see Figure 1) [3], has been developed to aid in tackling this issue.

Multibody dynamic simulations are crucial for the research of robotic systems, namely by allowing the study of the robot's motion and trajectories, the efficiency comparison of different control solutions, and the design optimization of the mechanism's bars, joints and actuators [4]. This work focuses on the development of CHARMIE's multibody computational model.

The methods for analysing multibody systems can be grouped into two categories in which: i) a commercial (or open-source) software is used to create the computational model, or ii) the equations of motion are derived and then implemented into an integrated development environment. The survey conducted in [5] shows that humanoid robot researchers mostly use method (i), justifying this decision with the usefulness of the various tools provided by the software, the quality of the user interfaces, and the ability to tackle complex systems without requiring in-depth studies. Method (ii) demands a more laborious setup stage, but it also provides powerful advantages such as allowing an optimization of the code to more efficiently tackle specific issues, giving the programmer full control over the simulation environment, and producing models with higher compatibility with external computational tools. The choice of the best method ultimately depends on the nature of the project and the intended applications of the virtual model. As an example, graphical interfaces and computer vision problems are easier to deal with using method (i), while the greater customizability granted by method (ii) allows for quicker parametric optimization studies. CHARMIE's multibody model will have three main uses, namely determining actuator loads, performing parametric optimization of static balancing mechanisms, and training neural network control solutions for trajectory control. The two methods are equally valid for the first study, but the second and third studies benefit greatly from the use of adequately implemented own code, therefore, this was the chosen methodology for the CHARMIE project.

The forward kinematics of this system was solved by using the recursive algorithm presented in [6]. In this method, bodies are first modelled in their local coordinates, then rotated using rotation matrices defined by Euler angles, and finally translated into their final positions. The inverse dynamic analysis is then carried out by using the the recursive Newton-Euler algorithm of [7] expressed in the mathematical notation of [4]. The present work further adapts the formulation of [4] to allow it to: tackle tree structure kinematic chains; integrate formulations that solve closed loop systems; and permit any choice of axis-orientations, avoiding the limits imposed by the Denavit–Hartenberg parameters. The aforementioned recursive algorithms are integrated into a seven-step methodology (Figure 2), which was used to assemble the multibody model of CHARMIE.

The main contribution of this work is twofold. First, it addresses a pressing concern in an ongoing project, namely, the need for an in-house optimized code to achieve the multibody model of CHARMIE, which is required for performing tests of various control solutions, and to study the optimization of the robot's mechanisms. Second, the work describes and validates a systematic seven-step methodology (see Figure 2) intended to simulate complex articulated robotic systems. This methodology groups and interconnects mathematical formulations, which are well-known from the literature, to create a complete multibody simulation from start to finish. The feasibility of the proposed approach was confirmed by employing it to a specific case study. By comparing the used methodology with the methods described in the literature, three main advantages were identified: i) It groups a full set of formulations that can be applied to build a complete dynamics simulator; ii) The obtained simulator has a high computational efficiency [8]; and iii) The resulting program is extremely modular and adaptable.

The remainder of this work begins with a Literature Review in Section 2, where the choice of used mathematical formulations is justified, and the main available alternatives to tackle the system's sub-problems are described. The seven steps of the new proposed methodology are grouped into three main parts, which correspond to the following sections of the paper: Section 3, constructing the multibody model and preparing it by converting its properties

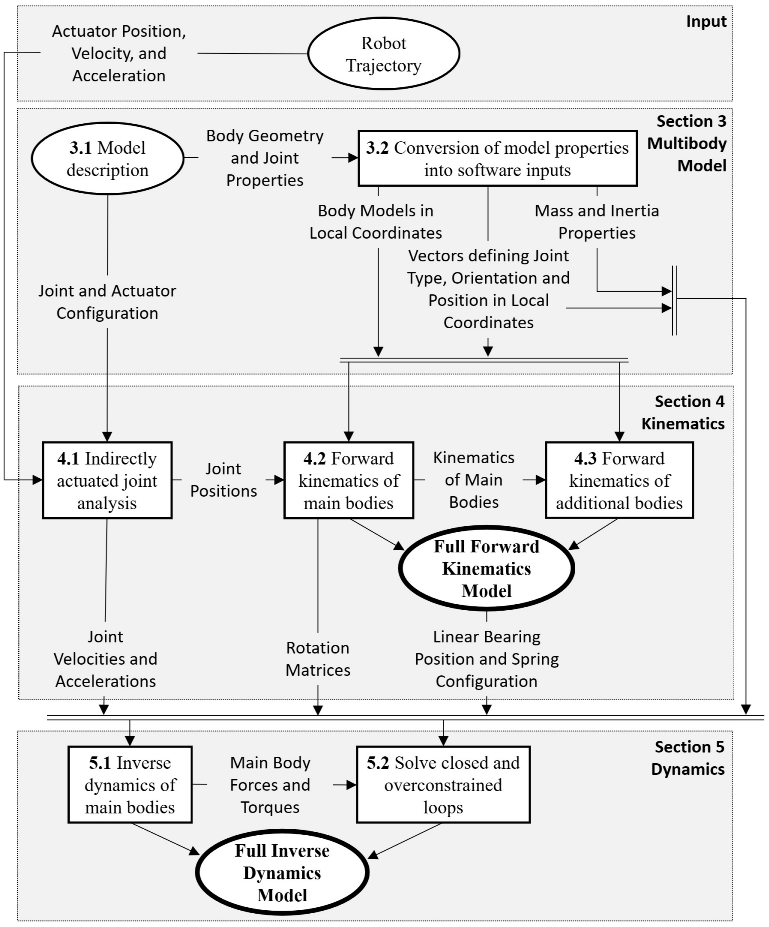
Input
Actuator Position, Velocity, and Acceleration
Robot Trajectory
Section 3 Multibody Model
3.1 Model description
Body Geometry and Joint Properties
3.2 Conversion of model properties into software inputs
Body Models in Local Coordinates
Mass and Inertia Properties
Vectors defining Joint Type, Orientation and Position in Local Coordinates
Joint and Actuator Configuration
Section 4 Kinematics
4.1 Indirectly actuated joint analysis
Joint Positions
4.2 Forward kinematics of main bodies
Kinematics of Main Bodies
4.3 Forward kinematics of additional bodies
Full Forward Kinematics Model
Joint Velocities and Accelerations
Rotation Matrices
Linear Bearing Position and Spring Configuration
Section 5 Dynamics
5.1 Inverse dynamics of main bodies
Main Body Forces and Torques
5.2 Solve closed and overconstrained loops
Full Inverse Dynamics Model

**Fig. 2** Flowchart describing the proposed seven-step method for the multibody computational analysis.

into the relevant inputs for the recursive algorithms; Section 4, analysing the forward kinematics, including the geometric study of indirectly actuated joints, implementing the recursive algorithm for the kinematics analysis, and solving the kinematics of additional bodies; and Section 5, analysing the inverse dynamics, including employing the Newton-Euler recursive algorithm and solving the dynamics of closed and over-constrained loops. Section 4 does not require any input from Section 5, allowing the program to be utilized for purely kinematic analysis (for example, for trajectory optimization). In Section 6 the developed methodology is validated by comparing the obtained outcomes with those produced from a commercial software. Thus, Section 7 concludes the work by presenting suggestions for future work and concluding remarks regarding the methodology's performance and applications.

## 2 Literature Review

The goal of this Literature review is to present alternative notations which could have been used to model CHARMIE, justify the choice of notation used, and compare it to the listed alternatives. The forward kinematics will first be presented, then methodologies for the inverse dynamics, and finally the multibody problem will be analysed as a whole.

The kinematics of a body is modelled by defining its position and orientation, as well as how these properties vary over time. The most employed method for the kinematic study of multibody robotic systems is the Denavit–Hartenberg parameters, in which four scalar variables define a [4x4] homogeneous transformation matrix that describes the rotation and translation between two consecutive joints in a serial chain of bodies [9]. Despite its popularity, this method possesses its own set of drawbacks, such as limiting the choice of orientation for the axes, which can make software outputs harder to interpret, and using homogeneous transformation matrices that reduce the computational efficiency by increasing the number of multiplications by ones and zeros [10]. This second limitation can be addressed by dividing the translation and rotation operations into two sequential calculations. Positions and displacements in Euclidean space can be simply defined using three-dimensional Cartesian coordinates, however, orientations and rotations provide a more complex problem. Rotation matrices ($[3 \times 3]$ matrices for three-dimensional space) are commonly utilized to define the rotation between two references because they are simple to understand, implement and manipulate. One of the several methods used alongside rotation matrices is the Euler angles, a set of three predefined rotations that can represent any orientation in space [11]. Some researchers avoid using Euler angles due to their two well-known limitations: gimbal lock and the Euler angle singularity [12]. An alternative for avoiding these limitations is to use quaternions, where rotations are defined by a vector with four scalars [13]. Quaternions are often favoured for their high computational efficiency [14]. Another common approach in multibody systems is screw theory, where two three-dimensional vectors characterize the Plücker coordinates of a line in space, and the magnitude of a screw and its pitch, which can be harnessed to describe the position and orientation of a rigid body [15]. Screw theory may be used alongside Lie Group algebra to obtain general and highly efficient recursive algorithms for spatial algebra [16]. A common application of screw theory within the field of multibody dynamics is the use spatial vector quantities to develop kinematics and dynamic algorithms based on spatial vector algebra [8,17]. Of the possibilities listed above (only some of several methodologies), the forward kinematics analysis of CHARMIE uses Euler angles [6]. Since a constrained system is studied, the gimbal lock and singularity problems of Euler angles are avoided by choosing the correct axes orientations [12]. This notation is also advantageous since it directly outputs the required inputs for the selected dynamic analysis method [18].

The goal of the inverse dynamic analysis of multibody systems is to determine the loads applied by each pair of connected bodies based on the known kinematics and physical properties of the system. Several main formulations are applicable for obtaining the equations of motion of dynamic systems: the Newton-Euler equations [19], the Lagrange equations [20], the Hamilton equations [21], and the equations from Kane's method [22]. In robotics, recursive Newton-Euler algorithms [7] are mainly used due to their simplicity and high computational efficiency (the number of calculations increases linearly with the number of bodies in the system) [17]. Recursive Newton-Euler algorithms are divided into two stages: first, iterations progress from the ground reference to the end-effectors to obtain the forward kinematics of each body; then, iterations progress from the end-effectors to the ground reference to calculate the forces and torques applied between each pair of connected bodies. The work by [23] derives highly efficient recursive Newton-Euler algorithms structured around screw theory and Lie algebra. In [17], the recursive Newton-Euler algorithm is adapted to work with spatial vector algebra, but the author notes that this causes a reduction in computational efficiency (in comparison to the standard notation) since it requires calculating the linear velocity of all bodies. One of the Newton-Euler notations with the highest computational efficiency is described in [24]. The efficiency of this notation was matched by [25], which employs Kane's equations to solve the dynamics of manipulators with only revolute or prismatic joints, modeled using the Denhavit-Hartenberg parameters. Parallel computation based notations are more commonly used for forward dynamics problems [26,27,28], but they can also be applied to further improve the computational effiency of inverse dynamics problems [29]. CHARMIE was modeled using the Newton-Euler formulation of [7] expressed in the mathematical notation of [4]. This base formulation is quite simple, and it is expressed as an algorithm which inputs joint properties and the rotation matrices between consecutive bodies to address serial kinematic systems modelled using Denhavit-Hartenberg parameters. This algorithmn was expanded in Section 5 of this work to better perform the inverse dynamic analysis of CHARMIE.

The two aforementioned analyses, direct kinematics and inverse dynamics, must both be performed to achieve the desired complete multibody model. These two studies are intertwined, so picking the most computationally efficient model for each problem independently may result in a non-optimal solution (due to the required computational effort of converting between notations). Therefore, several works in the literature are prepared to tackle these problems as a whole, for example [17] expresses the equations using spatial vectors algebra, while [30] uses spatial kernel operators (SKO) and spatial propagation operators (SPO) for the multibody analysis of complex systems.

The analysis of CHARMIE must solve two specific problems, namely tree structures [31] and closed loops [32]. Diverse methods have been presented to systematically solve tree like kinematic structures, such as [16,17,30,31]; also, formulations for tackling serial kinematic chains can be easily extended to address tree kinematic structure [8]. Depending on the complexity of the system, the solution of closed loops can become a more intricate problem, for example, the SKO models in [30] have issues in dealing with closed loops. Nonetheless, authors have described and integrated into their models systematic methods of dealing with closed loops; the work by [17] describes a set of strategies for solving closed systems, the work by [32] completes the model it presented in [31] to allow it to tackle closed chains, in [33] the divide-and-conquer (DCA) method is extended for the inverse dynamic analysis of open and closed loops, and [34]uses develops an efficient method for studying the inverse dynamics of manipulators based on the Newton-Euler equations and the virtual work principle.

In the present work, the proposed systematic seven-step approach requires a more in-depth analysis of the target of the study in comparison to more generalized formulations, which are prepared to deal with any geometry. The advantages of developing a more specific solution for the required problem is that a high-efficiency inverse dynamics algorithm could be used [17], and the closed loops were tackled with more specific notations that result in higher computational efficiency. The proposed approach integrates the solution of several key issues in obtaining a multibody simulator, namely tackling tree structure kinematic chains and closed loops, implementing the equations of motion and generating a graphical output. The resulting program is highly modular, which provides three main advantages: other formulations from the literature review can be tested at a later stage to further increase computational efficiency; functions can be disabled for more focused studies (such as performing a trajectory optimization based on kinematics only); and additional mathematical notations that interact with the model can be added, for example, in an ongoing study, CHARMIE's locomotion has been modeled using a direct dynamic approach which was fully integrated within this model.

## 3 Construction and Preparation of the Multibody Model

The initial step for creating the multibody model of CHARMIE is to define each of the robot's bodies and joints. First, bodies are labelled according to their position in the kinematic chain. Then, using information obtained from the robot's CAD model, data defining the geometry, mass and inertia of the bodies and the orientation, type and position of the joints is computed to serve as input for the recursive algorithms.

### 3.1 Description of the Multibody Model

CHARMIE's structure is divided into five main sections. From base to end-effector they are: i) the omnidirectional base, which includes the suspension and locomotion systems; ii) the 2-DOF (Degrees of Freedom) articulated lower and upper body of the robot, responsible for increasing the robot's workspace; iii) the left arm and iv) the right arm, each with 6 DOF for their motion, and

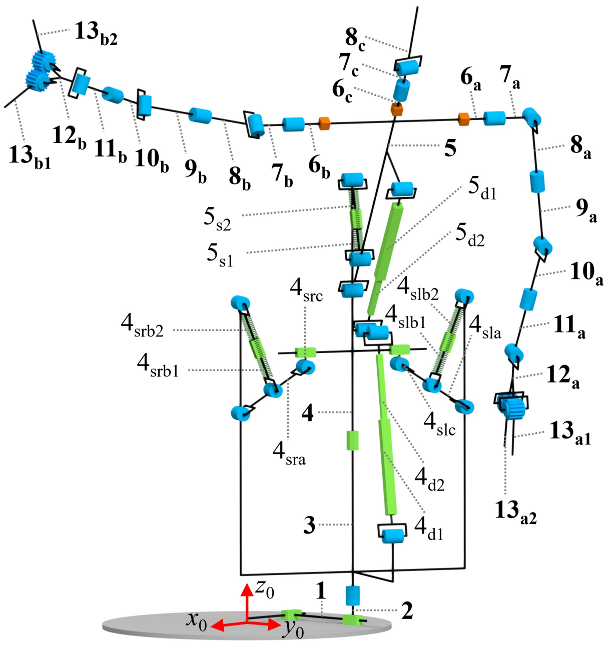

**Fig. 3** Kinematic model of the CHARMIE mobile manipulator robot.

an additional one for opening and closing a claw-like end-effector; and v) the robot's head with a 2-DOF neck. In the current stage of the virtual model, the kinematics and dynamics of the locomotion and suspension systems are simplified, represented as three DOFs that allow the robot to slide and rotate along the ground plane. This results in a total of 21 DOFs, which are actuated by 16 revolute motors, two linear actuators, and three additional fictitious actuators (two linear, and one revolute) to control the locomotion.

The complete assembly of the five sections results in a total of 40 rigid bodies, interconnected by 34 revolute joints, ten prismatic joints and three rigid joints (see Figure 3). Bodies with varying lengths, namely linear actuators or tension springs, are represented as two bodies connected by a prismatic joint.

The proposed methodology requires identifying the main kinematic chains of the studied system. These chains progress from base to end-effector; links $\mathrm{i}-1$ are considered as preceding link i, while links $\mathrm{i}+1$ are succeeding link i. In CHARMIE's case, the bodies from the floor plane to the robot's upper body (links 1-5) constitute a single serial kinematic chain. This main chain is then split into three that represent the left arm (links $6_\mathrm{a}$-$13_\mathrm{a}$), the right arm (links $6_\mathrm{b}$-$13_\mathrm{b}$), and the head (links $6_\mathrm{c}$-$8_\mathrm{c}$) respectively. The indirect actuation and static balancing systems placed between the body pairs 3-4, and 4-5, create closed and over-constrained loops that must be tackled separately in the dynamic analysis. Figure 3 distinguishes the bodies that belong to the main kinematic chains from those that are auxiliary by labelling the main ones with bold letters.

## 3.2 Conversion of the Body Properties into Software Inputs

The recursive algorithms for the kinematics and dynamics analyses require two main types of inputs. The first type is the position, velocity and acceleration of each actuator. These spatial properties are defined by the robot's trajectory planning and control (which can either be predetermined or updated during the execution of the simulation). The second type of input is the physical and geometrical properties of the robot, which include each body's dimension, shape, mass, and inertia, as well as the type, position and orientation of all joints. All these characteristics must be completely defined for bodies belonging to the main kinematic chains.

The first defined property was the shape of each body. This process is started by associating a local coordinate axis with origin $\mathrm{O_i}$ to body i. Then, a point cloud is used to describe the required shapes. At this stage, a reduced number of points were defined manually to convey the key information of the geometries. Bodies are modelled in the point cloud choosing the most convenient axis direction to simplify the robot's assembly and description (for example, for the robot's arms, the $z$ axis represents the length of all parts). In the equations throughout this work, these points can be expressed regarding different coordinate axis using the following notation:

- $\mathbf{P}'_\mathrm{i}$: the coordinates of point P of body i expressed in the local axis of i.
- $\mathbf{P}''_\mathrm{i}$: the coordinates of point P of body i expressed in an auxiliary axis that represents body i rotated around $\mathrm{O}'_\mathrm{i}$ to its current global orientation considering the configuration of the robot.
- $\mathbf{P}_\mathrm{i}$: the coordinates of point P of body i expressed in the global coordinate axis.
- $\mathbf{P}'^{\mathrm{j}}_\mathrm{i}$: the coordinates of point P of body i expressed in the local axis of body j.

Besides the origin, a set of important points must be identified and kept coherent for all bodies so that the equations are simpler to implement. Of these points, $\mathrm{A_i}$ always indicates the connection between body i and its predecessor in the kinematic chain. If a revolute joint precedes body i, point $\mathrm{A_i}$ is fixed and overlaps with $\mathrm{O_i}$. If a prismatic joint precedes i, the position of $\mathrm{A_i}$ in local coordinates is calculated using:

$$\mathbf{A}'_\mathrm{i} = \mathbf{O}'_\mathrm{i} + d_i\mathbf{z}_\mathrm{i} \tag{1}$$

where $d_i$ is the displacement between the origin of the body and the contact point between body i and its predecessor in the kinematic chain, and $\mathbf{z}_\mathrm{i}$ is the unit vector defining the orientation of the joint preceding i expressed in local coordinates (better defined in Equation 28).

The algorithms also require defining points $\mathrm{B_i}$, which describe the position of the joints between body i and its successors in the kinematic chain, and points $\mathrm{C_i}$, which represent the coordinates of the centre of mass of body i (determined using the CAD model). Points $\mathrm{O_i}$, $\mathrm{A_i}$, $\mathrm{B_i}$, and $\mathrm{C_i}$ are sufficient to

**Table 1** Point coordinates defining the geometry of CHARMIE's body 5, whose visual representation is shown in Figure 4.

| Point | Coordinates [mm] | | |
|---|---|---|---|
| | $x$ | $y$ | $z$ |
| $O'_5$ | 0.0 | 0.0 | 0.0 |
| $A'_5$ | 0.0 | 0.0 | 0.0 |
| $Ba'_5$ | −200.0 | −2.0 | 460.0 |
| $Bb'_5$ | 200.0 | −2.0 | 460.0 |
| $Bc'_5$ | 0.0 | 0.0 | 495.0 |
| $C'_5$ | 0.0 | −5.7 | 338.5 |
| $D'_5$ | 0.0 | 0.0 | −15.0 |
| $E'_5$ | 0.0 | 0.0 | 85.0 |
| $F'_5$ | 0.0 | 0.0 | 397.2 |
| $G'_5$ | 0.0 | 56.0 | 397.2 |
| $H'_5$ | 0.0 | 0.0 | 460.0 |
| $I'_5$ | 0.0 | −20.0 | 460.0 |
| $J'_5$ | 200.0 | −20.0 | 460.0 |
| $K'_5$ | −200.0 | −20.0 | 460.0 |

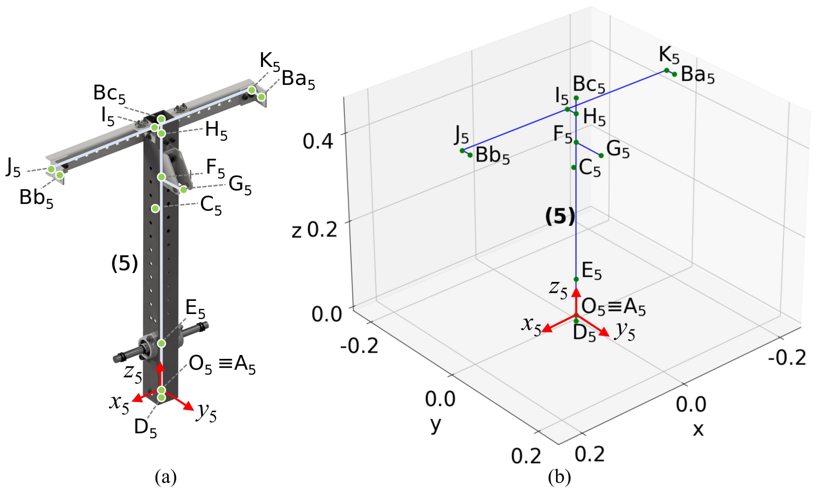

**Fig. 4** (a) CAD model of CHARMIE's body 5, and (b) CHARMIE's body 5 in the simulation environment depicted in local coordinates by the key points described in Table 1.

fully define geometries for executing the recursive algorithms. Any number of additional points can be used to better describe each body's shape. Table 1 and Figure 4 represent the points used for modelling CHARMIE's body 5.

With the shapes fully characterized, the focus then becomes the joints between them. Each joint was described using three variables. $JT_\mathrm{i}$ defines the joint preceding body i as P—Prismatic, R—Revolute, or F—Fixed. The axis this joint revolves around, or slides along, is then identified with variable $JO_\mathrm{i}$. Finally, using information from the body's geometry, vector $\mathbf{r}^{i}_{\mathrm{Ai,Bi}}$ expresses the displacement between the preceding and succeeding joints of body i in

**Table 2** Description of CHARMIE's joints by type ($JT_{\mathrm{i}}$), orientation ($JO_{\mathrm{i}}$) and position ($\mathbf{r}^{i}_{\mathrm{Ai,Bi}}$).

| Body i | $JT_{\mathrm{i}}$ | $JO_{\mathrm{i}}$ | $\mathbf{r}^{i}_{\mathrm{Ai,Bi}}$ [mm] | | |
|---|---|---|---|---|---|
| | | | $x$ | $y$ | $z$ |
| 1 | P | $x$ | $d_1$ | 0.0 | 0.0 |
| 2 | P | $y$ | 0.0 | $d_2$ | 0.0 |
| 3 | R | $z$ | 0.0 | 0.0 | 290.0 |
| 4 | P | $z$ | 0.0 | 0.0 | 438.0 $-d_4$ |
| 5 (a) | R | $x$ | −200.0 | −2.0 | 460.0 |
| (b) | R | $x$ | 200.0 | −2.0 | 460.0 |
| (c) | R | $x$ | 0.0 | 0.0 | 495.0 |
| $6_{\mathrm{a=b}}$ | F | — | 0.0 | 0.0 | 30.0 |
| $7_{\mathrm{a=b}}$ | R | $z$ | 0.0 | 0.0 | 61.5 |
| $8_{\mathrm{a=b}}$ | R | $x$ | 0.0 | 0.0 | 145.0 |
| $9_{\mathrm{a=b}}$ | R | $z$ | 0.0 | 0.0 | 145.0 |
| $10_{\mathrm{a=b}}$ | R | $x$ | 0.0 | 0.0 | 140.0 |
| $11_{\mathrm{a=b}}$ | R | $z$ | 0.0 | 0.0 | 140.0 |
| $12_{\mathrm{a=b}}$ (1) | R | $x$ | 0.0 | 0.0 | 52.3 |
| (2) | R | $x$ | 0.0 | 0.0 | 52.3 |
| $13_{\mathrm{a1=b1}}$ | R | $y$ | — | — | — |
| $13_{\mathrm{a2=b2}}$ | R | $y$ | — | — | — |
| $6_{\mathrm{c}}$ | R | — | 0.0 | 0.0 | 46.3 |
| $7_{\mathrm{c}}$ | R | $z$ | 14.8 | −5.3 | 18.3 |
| $8_{\mathrm{c}}$ | R | $x$ | — | — | — |

its local coordinates. The values of these three variables for all main joints of CHARMIE are shown in Table 2.

Three fixed joints were introduced into body 5 of CHARMIE to perform constant rotations, allowing more natural choices of axes orientations for the sub-kinematic chains a, b, and c. The changes in orientation were defined as intrinsic ZXZ Euler rotations (see Equation 17) as shown in Equations 2-4.

$$\mathbf{R}^{5}_{6\mathrm{a}} = \mathrm{ZXZEuler}(\pi/2, -\pi/2, \pi) \tag{2}$$

$$\mathbf{R}^{5}_{6\mathrm{b}} = \mathrm{ZXZEuler}(\pi/2, \pi/2, 0) \tag{3}$$

$$\mathbf{R}^{5}_{6\mathrm{c}} = \mathrm{ZXZEuler}(0, 0, 0) \tag{4}$$

The aforementioned variables are sufficient to study the kinematics of the robot. For the dynamic analysis, it is also necessary to define the masses $m_i$ (Table 3) and inertia matrices $\bar{\mathbf{I}}^{i}_{i}$ (Table 4). To simplify the system, bodies that don't belong to the main kinematic chains were considered massless. The system's total mass was maintained by adding the removed masses into adjacent bodies of the main kinematic chains. The positions of the centres of mass were defined in the local coordinates of each body i using vector $\mathbf{r}^{i}_{\mathrm{Ai,Ci}}$, which expresses the displacement between the body's joint with its preceding link and its centre of mass.

The definition of all inputs listed above concludes the preparatory stages for the robot's multibody simulation.

**Table 3** Mass ($m_{\mathrm{i}}$) and centre of mass position ($\mathbf{r}^{i}_{\mathrm{Ai,Ci}}$) of CHARMIE's bodies.

| Body i | $m_{\mathrm{i}}$[kg] | $\mathbf{r}^{i}_{\mathrm{Ai,Ci}}$ [mm] | | |
|---|---|---|---|---|
| | | $x$ | $y$ | $z$ |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 45.7 | −0.5 | 16.5 | 88.7 |
| 4 | 7.8 | 0.0 | 21.0 | $330-d_4$ |
| 5 | 2.2 | 0.0 | −5.7 | 338.5 |
| $6_{\mathrm{a=b}}$ | 0.3 | 0.0 | 0.0 | 15.0 |
| $7_{\mathrm{a=b}}$ | 0.3 | 0.0 | 0.0 | 30.8 |
| $8_{\mathrm{a=b}}$ | 0.4 | 0.0 | 0.0 | 72.5 |
| $9_{\mathrm{a=b}}$ | 0.4 | 0.0 | 0.0 | 72.5 |
| $10_{\mathrm{a=b}}$ | 0.4 | 0.0 | 0.0 | 70 |
| $11_{\mathrm{a=b}}$ | 0.4 | 0.0 | 0.0 | 70 |
| $12_{\mathrm{a=b}}$ | 0.2 | 0.0 | 11.6 | 28.4 |
| $13_{\mathrm{a1=b1}}$ | 0.1 | −24.7 | 0.0 | 17.8 |
| $13_{\mathrm{a2=b2}}$ | 0.1 | 24.7 | 0.0 | 17.8 |
| $6_{\mathrm{c}}$ | 0.3 | 0.0 | −1.3 | 5.1 |
| $7_{\mathrm{c}}$ | 0.4 | −0.8 | 7.1 | 9 |
| $8_{\mathrm{c}}$ | 0.8 | −11.8 | 15.2 | 75.7 |

**Table 4** Physical properties defining the inertia matrices ($\bar{\mathbf{I}}^{i}_{i}$) of CHARMIE's bodies.

| Body i | Inertia[kg.mm$^2$] | | | | | |
|---|---|---|---|---|---|---|
| | $I_{xx}$ | $I_{yy}$ | $I_{zz}$ | $I_{xy}$ | $I_{xz}$ | $I_{yz}$ |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 1.3E+6 | 1.5E+6 | 1.8E+6 | 1.4E+2 | 4.7E+3 | −5.3E+6 |
| 4 | 1.4E+5 | 1.9E+5 | 9.1E+4 | 2.9E+1 | − 7.3E+1 | −2.4E+4 |
| 5 | 5.4E+4 | 6.6E+4 | 1.3E+4 | 1.2E+0 | −1.2E+1 | 1.87+3 |
| $6_{\mathrm{a=b}}$ | 3.2E+1 | 3.2E+1 | 2.0E+1 | 0.0 | 0.0 | 0.0 |
| $7_{\mathrm{a=b}}$ | 1.0E+2 | 1.0E+2 | 2.0E+1 | 0.0 | 0.0 | 0.0 |
| $8_{\mathrm{a=b}}$ | 7.1E+2 | 7.1E+2 | 2.6E+1 | 0.0 | 0.0 | 0.0 |
| $9_{\mathrm{a=b}}$ | 7.1E+2 | 7.1E+2 | 2.6E+1 | 0.0 | 0.0 | 0.0 |
| $10_{\mathrm{a=b}}$ | 6.6E+2 | 6.6E+2 | 2.6E+1 | 0.0 | 0.0 | 0.0 |
| $11_{\mathrm{a=b}}$ | 6.6E+2 | 6.6E+2 | 2.6E+1 | 0.0 | 0.0 | 0.0 |
| $12_{\mathrm{a=b}}$ | 7.5E+1 | 5.6E+1 | 3.3E+1 | 0.0 | 0.0 | −1.8E+1 |
| $13_{\mathrm{a1=b1}}$ | 4.2E+1 | 6.2E+1 | 2.7E+1 | 0.0 | 2.0E+1 | 0.0 |
| $13_{\mathrm{a2=b2}}$ | 4.2E+1 | 6.2E+1 | 2.7E+1 | 0.0 | −2.0E+1 | 0.0 |
| $6_{\mathrm{c}}$ | 1.5E+2 | 1.3E+2 | 2.5E+2 | 0.0 | 0.0 | 4.7E+0 |
| $7_{\mathrm{c}}$ | 3.0E+2 | 2.6E+2 | 5.2E+2 | 2.6E+0 | −0.8E−1 | 6.8E+0 |
| $8_{\mathrm{c}}$ | 8.5E+3 | 7.2E+2 | 4.9E+3 | 3.2E+1 | 1.4E+2 | −2.2E+2 |

## 4 Forward Kinematics Analysis

The forward kinematics analysis uses the actuator positions, velocities, and accelerations defined by the trajectory, as well as the inputs that describe the shape of the bodies and joints, to determine the motion of the robot over time. Since the used recursive algorithm [6] requires the configuration of the joints, not the actuators, this analysis is initialized by studying the behaviour

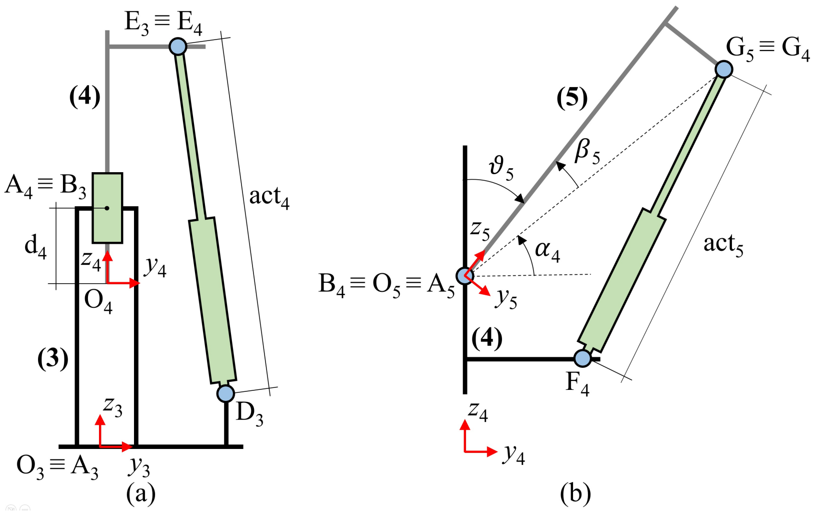

**Fig. 5** Geometric parameters required for the analysis of CHARMIE's indirectly actuated joints, namely (a) the joint preceding body 4, and (b) the joint preceding body 5.

of the indirectly actuated joints. Subsequently, the recursive algorithm can be utilized to obtain the kinematics of the main bodies. Finally, the motion of the additional bodies, such as springs and linear actuators, is determined to fully characterize the robot's kinematics.

## 4.1 Geometric Analysis of the Indirectly Actuated Joints

In this section, a geometric approach is used to study the two indirectly actuated joints of CHARMIE, which are placed preceding bodies 4 and 5. At this stage of the methodology, each body is defined in local coordinates, but the full model is yet to be assembled. Since distinct references are being considered, the distance between points from different bodies can't be used.

The joint preceding body 4 is a prismatic joint controlled by a linear actuator. The goal is to determine the joint displacement $d_4$ as a function of the actuator length $act_4$ (see Figure 5a). Since the two bodies only move linearly along the $z$ axis, coordinates in the two local $y$ axes can be compared directly. The joint displacement is expressed as:

$$d_4 = A'_4 z - O'_4 z \tag{5}$$

$A'_4 z$ can't be determined in its local coordinates without the joint displacement but, by expressing both $A'_4 z$ and $O'_4 z$ in relation to body 3, it is possible to establish $d_4$ as a function of $act_4$. $A'_4 z$ in body 3's reference corresponds directly to point $B'_3 z$. The coordinates of $O'_4 z$ can then be determined by starting in point D$_3$ and adding the vertical displacement caused by the linear actuator (obtained using the Pythagorean theorem), then subtracting the vertical displacement between point $E_4$ and point $O_4$, defined by the constant geometry of body 4. This results in Equation 6.

$$d_4 = B'_3z - \left(D'_3z + \sqrt{act_4{}^2 - (E'_3y - D'_3y)^2} - (E'_4z - O'_4z)\right) \tag{6}$$

This expression solves the required kinematic relation for the joint preceding body 4.

The joint preceding body 5 is a revolute joint controlled by a linear actuator. The goal is to determine the joint angular rotation $\vartheta_5$ as a function of the actuator length $act_5$ (see Figure 5b). The value of $\vartheta_5$ can be obtained with the auxiliary angles $\alpha_4$ and $\beta_5$ using the expression:

$$\vartheta_5 = \alpha_4 + \beta_5 - \frac{\pi}{2} \tag{7}$$

Angle $\beta_5$ is constant, defined by the geometry of body 5 as:

$$\beta_5 = \arctan\left(\frac{G'_5y - A'_5y}{G'_5z - A'_5z}\right) \tag{8}$$

Angle $\alpha_4$ varies over time with the configuration of the linear actuator. By expressing the coordinates of its endpoint $G_5$ in the reference of body 4 ($G_4$), this angle can be formulated as:

$$\alpha_5 = \arctan\left(\frac{G'_4z - B'_4z}{G'_4y - B'_4y}\right) \tag{9}$$

The local coordinates of $B_4$ are defined by the geometry of body 4. The position of $G_4$ is calculated as the intersection between two circumferences, one with centre $F_4$ and radius $act_5$, and the other with centre $B_4$ and radius equal to the distance between $B_4$ and $G_4$. To determine the circumferences' intersections, three auxiliary parameters, $e_4$, $l_4$, and $h_4$, are calculated:

$$e_4 = \sqrt{(F'_4y - B'_4y)^2 + (F'_4z - B'_4z)^2} \tag{10}$$

$$l_4 = \frac{act_5{}^2 - \left[(G'_5y - A'_5y)^2 + (G'_5z - A'_5z)^2\right] + e_4{}^2}{2e_4} \tag{11}$$

$$h_4 = \sqrt{act_5{}^2 - l_4{}^2} \tag{12}$$

It then becomes possible to determine the coordinates of $G_4$ using Equations 13-14.

$$G'_4y = \frac{l_4}{e_4}(B'_4y - F'_4y) + \frac{h_4}{e_4}(B'_4z - F'_4z) + F'_4y \tag{13}$$

$$G'_4 z = \frac{l_4}{e_4}(B'_4 z - F'_4 z) + \frac{h_4}{e_4}(B'_4 y - F'_4 y) + F'_4 z \tag{14}$$

By combining Equations 7-14, the relation between the configuration of the joint preceding body 5 and its linear actuator is established.

### 4.2 Recursive algorithm for the forward kinematics analysis of the main bodies

The forward recursive algorithm determines the position and orientation of all bodies that are a part of the main kinematic chains of CHARMIE by using the joint position, orientation, type and configuration. The algorithm models one body at a time, starting on the robot's base (body 1), and progressing along the kinematic chain until the end-effectors (bodies 13a, 13b and 8c) are reached. Each body is modelled following three sequential steps.

The first step is creating a model of body i in its local coordinates. For CHARMIE, these models were obtained using the coordinates of key points as explained in Section 3.2.

The second step is rotating body i around its origin (Point $\mathrm{O_i}$) so it assumes its current orientation concerning the global reference. This is achieved using the expression:

$$\mathbf{P}''_\mathrm{i} = \mathbf{R}^0_\mathrm{i} \mathbf{P}'_\mathrm{i} \tag{15}$$

where each point $\mathrm{P_i}$ is rotated using the $\mathbf{R}^0_\mathrm{i}$ rotation matrix that defines the orientation of body i.

The $\mathbf{R}^0_\mathrm{i}$ rotation matrix is calculated from the combination of two rotations: matrix $\mathbf{R}^0_\mathrm{i-1}$ that describes the orientation of the body preceding i (obtained in the previous iteration of the recursive algorithm); and matrix $\mathbf{R}^\mathrm{i-1}_\mathrm{i}$ that defines the rotation between the preceding body and the current body. This process is formulated as:

$$\mathbf{R}^0_\mathrm{i} = \mathbf{R}^0_\mathrm{i-1} \mathbf{R}^\mathrm{i-1}_\mathrm{i} \tag{16}$$

The rotations between consecutive bodies are expressed using intrinsic ZXZ Euler angles. The rotation matrix corresponding to a specific set of angles is obtained using the following function [11]:

$$\mathrm{ZXZEuler}(Z_1, X_2, Z_3) = \begin{bmatrix} \mathrm{c_1c_3 - s_1c_2s_3} & \mathrm{-c_1s_3 - s_1c_2c_3} & \mathrm{s_1s_3} \\ \mathrm{s_1c_3 + c_1c_2s_3} & \mathrm{c_1c_2c_3 - s_1s_3} & \mathrm{-c_1s_2} \\ \mathrm{s_2s_3} & \mathrm{s_2c_3} & \mathrm{c_2} \end{bmatrix} \tag{17}$$

where c and s represent the cosine and sine functions respectively, and the sub-indices 1, 2, and 3 refer to the three considered Euler angles.

Combining the joint parameters with Equation 17, the $\mathbf{R}^\mathrm{i\text{-}1}_\mathrm{i}$ rotation matrices can be directly determined as:

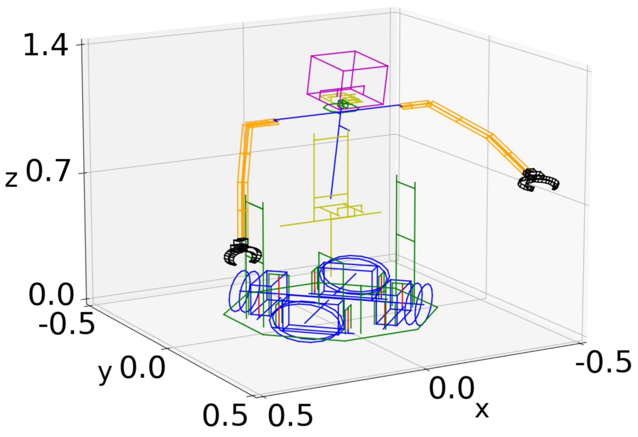

**Fig. 6** 3D model of CHARMIE's resulting from the recursive algorithm for forward kinematics. Only bodies that are a part of the main kinematic chains are represented.

$$\mathbf{R}_{\mathrm{i}}^{\text{i-1}} = \begin{cases} \text{ZXZEuler}(0, \vartheta_i, 0) & \forall JT_{\mathrm{i}} = R \wedge JO_{\mathrm{i}} = x \\ \text{ZXZEuler}\left(\frac{\pi}{2}, \vartheta_i, -\frac{\pi}{2}\right) & \forall JT_{\mathrm{i}} = R \wedge JO_{\mathrm{i}} = y \\ \text{ZXZEuler}(0, 0, \vartheta_i) & \forall JT_{\mathrm{i}} = R \wedge JO_{\mathrm{i}} = z \\ \text{ZXZEuler}(0, 0, 0) & \forall JT_{\mathrm{i}} = P \end{cases} \tag{18}$$

This expression defines all rotations between the consecutive bodies of CHARMIE, which in turn is used to determine the orientation of all bodies, concluding step two of the kinematics recursive algorithm.

The third and final step is moving the body to its current position in the global coordinate reference. This is achieved by moving point $\mathrm{A_i}$ of the current body to point $\mathrm{B_{i-1}}$ contained in its predecessor in the kinematic chain (whose position in the global coordinates is determined in the previous iteration). This process guarantees cohesion between the joints of the assembly. By moving all $\mathrm{P_i}$ points along this same path, body i is fully translated into its actual position, as described by the formulation:

$$\mathbf{P}_{\mathrm{i}} = \mathbf{P}_{\mathrm{i}}'' + (\mathbf{B}_{\mathrm{i}-1} - \mathbf{A}_{\mathrm{i}-1}'') \tag{19}$$

The three steps are repeated for all bodies of the main kinematic chains, determining their positions and orientations in the global reference. By concluding the recursive algorithm for the forward kinematics analysis, and implementing it into the CHARMIE robot, the model shown in Figure 6 was obtained.

## 4.3 Kinematics of Additional Bodies

The recursive algorithm solves only the placement of links that are a part of the main kinematic branches. In the CHARMIE robot, there are additional components placed between bodies 3, 4, and 5 that constitute closed loops, namely linear actuators, extensions springs, and the auxiliary bars of a static balancing mechanism.

In the particular case where the two ends of an auxiliary body are points incorporated within main bodies, such is the case for both linear actuators and the extensions springs between links 4 and 5, the position and orientation can be fully defined from the coordinates of points established in the recursive algorithm. The length $l_\mathrm{s}$ of these auxiliary bodies is calculated with the expression:

$$l_\mathrm{s} = \sqrt{(S_\mathrm{i}x - S_\mathrm{j}x)^2 + (S_\mathrm{i}y - S_\mathrm{j}y)^2 + (S_\mathrm{i}z - S_\mathrm{j}z)^2} \tag{20}$$

where $\mathrm{S_i}$ and $\mathrm{S_j}$ are the two ends of the considered body.

The orientation is then determined using the law of cosines. A new point, $\mathrm{R_i}$, is added to one of the adjacent main bodies. $\mathrm{R_i}$ must be aligned along one of the local axes with the connection point $\mathrm{S_i}$. Points $\mathrm{R_i}$, $\mathrm{S_i}$, and $\mathrm{S_j}$ now form a triangle whose sides have lengths $l_s$, $a_s$ and $b_s$. These lengths can all be calculated using Equation 20. The rotation $\alpha_\mathrm{s}$ between the chosen local axis of the main body, and the corresponding local axis of the auxiliary body, is then determined using the expression:

$$\alpha_\mathrm{s} = \frac{{l_\mathrm{s}}^2 + {b_\mathrm{s}}^2 - {a_\mathrm{s}}^2}{2 l_\mathrm{s} b_\mathrm{s}} \tag{21}$$

where $a_s$ is the side of the triangle opposite to the calculated internal angle.

The mechanism responsible for the static balancing of the squatting motion of CHARMIE, placed between links 3 and 4, contains all bodies that can't be directly analysed with the method described above. This mechanism is composed of two symmetrical halves, so, for simplification, only the left side (bodies $4_\mathrm{sla}$, $4_\mathrm{slb}$, and $4_\mathrm{slc}$ of Figure 3) will be studied. The coordinates of three points, $\mathrm{B_{4sla}}$, $\mathrm{B_{4slb}}$, and $\mathrm{B_{4slc}}$ (see Figure 7), must be determined to fully characterise the required bodies.

At this stage, all main bodies have been modelled, so points from both links 3 and 4 can be expressed in the local coordinates of link 3. The altitude of $\mathrm{B_{4slc}}$ can be directly determined since:

$$B'^3_{\mathrm{4slc}} z = H'^3_4 z \tag{22}$$

Equations 23-24 can then be used to determine the coordinates of $\mathrm{B_{4sla}}$:

$$B'^3_{\mathrm{4sla}} z = B'^3_{\mathrm{4slc}} z - c_{\mathrm{4slc}} \tag{23}$$

$$B'^3_{\mathrm{4sla}} x = F'^3_3 x - \sqrt{{b_{\mathrm{4sla}}}^2 - (B'^3_{\mathrm{4sla}} z - F'^3_3 z)^2} \tag{24}$$

Point $\mathrm{B_{4slc}}$ can subsequently be fully defined using:

$$B'^3_{\mathrm{4slc}} x = B'^3_{\mathrm{4slc}} x \tag{25}$$

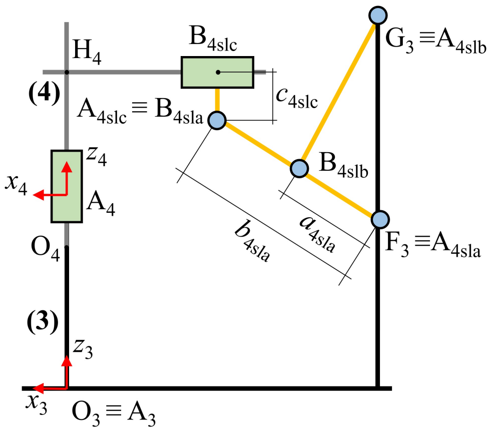

**Fig. 7** Geometric parameters required for the kinematics analysis of CHARMIE's auxiliary bodies responsible for the static balancing of the motion between links 3 and 4.

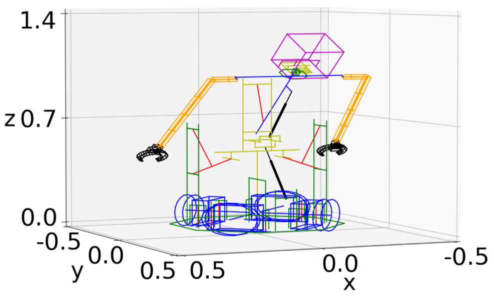

**Fig. 8** Complete model of CHARMIE assembled in the simulation environment as a result of the forward kinematic analysis.

The coordinates of the last point, B$_{4\text{slb}}$, are calculated using the triangle proportionality theorem as follows:

$$B'^3_{4\text{slb}}x = F'^3_3 x + \frac{a_{4\text{sla}}}{b_{4\text{sla}}} \left( B'^3_{4\text{sla}}x - F'^3_3 x \right) \tag{26}$$

$$B'^3_{4\text{slb}}z = F'^3_3 z + \frac{a_{4\text{sla}}}{b_{4\text{sla}}} \left( B'^3_{4\text{sla}}z - F'^3_3 z \right) \tag{27}$$

The points obtained from Equations 22-27 are expressed regarding the local reference of body 3. The global coordinates of the auxiliary bodies are determined by applying to these points the same rotation and translation that body 3 undergoes during the iterations of the recursive algorithm.

With the full definition of all auxiliary bodies, the forward kinematics analysis of CHARMIE is concluded, resulting in the model shown in Figure 8.

## 5 Inverse Dynamics Analysis

The inverse dynamics analysis processes the joint position, velocities and accelerations from the robot's trajectories, the rotation matrices from the recursive kinematics algorithm, and the physical properties of the robot's bodies and joints, to compute the forces and torques applied in each pair of connected bodies.

The approach used for the study of the dynamics revolves around the recursive Newton-Euler algorithm of [7]. The used formulation is based on the algorithm presented in [4], which solves the dynamics of bodies assembled in serial chains modelled using the Denavit–Hartenberg parameters. The present work modifies this algorithm to allow it: to automatically tackle tree structure kinematic chains, to integrate other formulations to solve closed loops, and to use any axes orientation (instead of limiting them due to the Denavit–Hartenberg notation). At this stage, some physical effects, such as the internal inertia of each motor, were neglected to increase computational efficiency.

By using this adapted formulation, the inverse dynamics analysis was divided into two main parts. The first is completely running the recursive algorithm for all branched components of the open kinematic chain. The latter is computing the additional formulations which tackle the closed systems between CHARMIE's bodies 3, 4 and 5.

### 5.1 Recursive Algorithm for the Inverse Dynamics Analysis of the Main Bodies

The recursive algorithm that studies the inverse dynamics is divided into two stages. The first stage, referred to as the forward iterations, is preparatory. Progressing from the robot's base to the end-effectors, each iteration uses information regarding the previous body in the kinematic chain, and data defining the preceding joint, to determine the angular velocity and acceleration of the current body, as well as the linear acceleration of its key points (centre of mass and joints). With the first stage concluded, the second stage, referred to as the backwards iterations, starts the calculations on the end-effectors and progresses towards the base of the robot. The algorithm uses information from the forward iterations, along with the body mass and inertia properties, to calculate the sum of forces and reactions applied on each body by its predecessors in the kinematic chain. If a body possesses only a single link preceding it, this sum automatically defines the forces and torques applied by it. On all equations in this section of the work, the superscript next to a variable denotes the coordinate reference it is expressed on.

Before beginning the algorithm, the unit vector $\mathbf{z}_\mathrm{i}$ characterising the joint orientation preceding body i must be defined using Equation 28.

$$\mathbf{z}_\mathrm{i} = \begin{cases} [1,0,0]^\mathrm{T} & \forall JO_\mathrm{i} = x \\ [0,1,0]^\mathrm{T} & \forall JO_\mathrm{i} = y \\ [0,0,1]^\mathrm{T} & \forall JO_\mathrm{i} = z \end{cases} \tag{28}$$

The forward iterations start by calculating the angular velocity $\boldsymbol{\omega}_\mathrm{i}^\mathrm{i}$ of the current body with Equation 29.

$$\boldsymbol{\omega}_\mathrm{i}^\mathrm{i} = \begin{cases} \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \boldsymbol{\omega}_\mathrm{i-1}^\mathrm{i-1} & \forall JT_\mathrm{i} = \mathrm{P} \\ \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \left( \boldsymbol{\omega}_\mathrm{i-1}^\mathrm{i-1} + \dot{\vartheta}_\mathrm{i} \mathbf{z}_\mathrm{i} \right) & \forall JT_\mathrm{i} = \mathrm{R} \end{cases} \tag{29}$$

where $\mathbf{R}_\mathrm{i}^{\mathrm{i-1T}}$ is the rotation matrix used to convert data expressed regarding the preceding body to the reference of the current body, $\boldsymbol{\omega}_\mathrm{i-1}^\mathrm{i-1}$ is the angular velocity of the preceding body in its reference, and $\dot{\vartheta}_\mathrm{i}$ is the rotation velocity of the revolute joint preceding body i.

The angular acceleration $\dot{\boldsymbol{\omega}}_\mathrm{i}^\mathrm{i}$ of body i can then be determined using:

$$\dot{\boldsymbol{\omega}}_\mathrm{i}^\mathrm{i} = \begin{cases} \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \dot{\boldsymbol{\omega}}_\mathrm{i-1}^\mathrm{i-1} & \forall JT_\mathrm{i} = \mathrm{P} \\ \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \left( \dot{\boldsymbol{\omega}}_\mathrm{i-1}^\mathrm{i-1} + \ddot{\vartheta}_\mathrm{i} \mathbf{z}_\mathrm{i} + \dot{\vartheta}_\mathrm{i} \dot{\boldsymbol{\omega}}_\mathrm{i-1}^\mathrm{i-1} \times \mathbf{z}_\mathrm{i} \right) & \forall JT_\mathrm{i} = \mathrm{R} \end{cases} \tag{30}$$

where $\dot{\boldsymbol{\omega}}_\mathrm{i-1}^\mathrm{i-1}$ is the angular acceleration of the preceding body expressed in its reference axis, and $\ddot{\vartheta}_\mathrm{i}$ the angular acceleration of the revolute joint preceding body i.

With the angular velocity and acceleration of the body determined, it is then possible to calculate the linear acceleration of its key points. These accelerations must include the Coriolis, Euler and centrifugal effects for the posterior force and torque calculations. The $\ddot{\mathbf{p}}_\mathrm{Bi}^\mathrm{i}$ linear acceleration of the $\mathrm{B_i}$ points connecting body i to its succeeding bodies are computed as:

$$\ddot{\mathbf{p}}_\mathrm{Bi}^\mathrm{i} = \begin{cases} \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \left( \ddot{\mathbf{p}}_\mathrm{Bi-1}^\mathrm{i-1} + \ddot{d}_\mathrm{i} \mathbf{z}_\mathrm{i} \right) + \dot{\boldsymbol{\omega}}_\mathrm{i}^\mathrm{i} \times \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} & \\ \quad + \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \left( \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} \right) + 2\dot{d}_\mathrm{i} \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \mathbf{z}_\mathrm{i} & \forall JT_\mathrm{i} = \mathrm{P} \\ \mathbf{R}_\mathrm{i}^{\mathrm{i-1T}} \ddot{\mathbf{p}}_\mathrm{Bi-1}^\mathrm{i-1} + \dot{\boldsymbol{\omega}}_\mathrm{i}^\mathrm{i} \times \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} + \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \left( \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} \right) & \forall JT_\mathrm{i} = \mathrm{R} \end{cases} \tag{31}$$

where $\ddot{\mathbf{p}}_\mathrm{Bi-1}^\mathrm{i-1}$ is the linear acceleration of point $\mathrm{A_i}$ expressed in the reference of the preceding body, $\dot{d}_\mathrm{i}$ and $\ddot{d}_\mathrm{i}$ are the velocity and acceleration of the linear actuator preceding body i, and $\mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i}$ is the vector pointing from the preceding joint $\mathrm{A_i}$ to the succeeding joint $\mathrm{B_i}$.

The forward iterations of the recursive algorithm are finished by determining the linear acceleration of the centre of mass $\ddot{\mathbf{p}}_\mathrm{Ci}^\mathrm{i}$ using the expression:

$$\ddot{\mathbf{p}}_\mathrm{Ci}^\mathrm{i} = \ddot{\mathbf{p}}_\mathrm{Bi}^\mathrm{i} + \dot{\boldsymbol{\omega}}_\mathrm{i}^\mathrm{i} \times \left( \mathbf{r}_\mathrm{Ai,Ci}^\mathrm{i} - \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} \right) + \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \left( \boldsymbol{\omega}_\mathrm{i}^\mathrm{i} \times \left( \mathbf{r}_\mathrm{Ai,Ci}^\mathrm{i} - \mathbf{r}_\mathrm{Ai,Bi}^\mathrm{i} \right) \right) \tag{32}$$

where $\mathbf{r}_\mathrm{Ai,Ci}^\mathrm{i}$ is the vector pointing from the preceding joint $\mathrm{A_i}$ to the center of mass $\mathrm{C_i}$.

The backwards iterations commence by first determining the sum of the $\mathbf{f}^{\mathrm{i}}_{\mathrm{j_a}}$ forces applied by the $n_a$ preceding bodies on the current body i using the formulation:

$$\sum_{\mathrm{j_a}=1}^{n_a} \mathbf{f}^{\mathrm{i}}_{\mathrm{j_a}} = m_{\mathrm{i}}\ddot{\mathbf{p}}^{\mathrm{i}}_{\mathrm{Ci}} + \sum_{\mathrm{j_b}=1}^{n_b} \left(\mathbf{R}^{\mathrm{i}}_{\mathrm{j_b}}\mathbf{f}^{\mathrm{j_b}}_{\mathrm{j_b}}\right) \tag{33}$$

where $\mathrm{j_a}$ is the pointer selecting each preceding body for the sum, $n_b$ is the number of succeeding bodies, $\mathrm{j_b}$ the pointer for each succeeding body, $m_{\mathrm{i}}$ the mass of the current body being studied, $\mathbf{f}^{\mathrm{j_b}}_{\mathrm{j_b}}$ the force applied by the current body on the $\mathrm{j_b}$ succeeding body expressed in the coordinates of $\mathrm{j_b}$, and $\mathbf{R}^{\mathrm{i}}_{\mathrm{j_b}}$ the rotation matrix used for converting the local reference of each body $\mathrm{j_b}$ into the reference of body i.

The sum of the $\boldsymbol{\mu}^{\mathrm{i}}_{\mathrm{j_a}}$ torques applied by the $n_a$ preceding bodies is then determined by the expression:

$$\begin{aligned}\sum_{\mathrm{j_a}=1}^{n_a} \boldsymbol{\mu}^{\mathrm{i}}_{\mathrm{j_a}} = \bar{\mathbf{I}}^{\mathrm{i}}_{\mathrm{i}}\dot{\boldsymbol{\omega}}^{\mathrm{i}}_{\mathrm{i}} + \boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{i}} \times \left(\bar{\mathbf{I}}^{\mathrm{i}}_{\mathrm{i}}\boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{i}}\right) + \sum_{\mathrm{j_a}=1}^{n_a}\left(-\mathbf{f}^{\mathrm{i}}_{\mathrm{j_a}} \times \mathbf{r}^{\mathrm{i}}_{\mathrm{Aja,Ci}}\right) \\ + \sum_{\mathrm{jb}=1}^{n_b}\left(\mathbf{R}^{\mathrm{i}}_{\mathrm{j_b}}\mathbf{f}^{\mathrm{j_b}}_{\mathrm{j_b}} \times \left(\mathbf{r}^{\mathrm{i}}_{\mathrm{Ai,Ci}} - \mathbf{r}^{\mathrm{i}}_{\mathrm{Ai,Bjb}}\right) + \mathbf{R}^{\mathrm{i}}_{\mathrm{j_b}}\boldsymbol{\mu}^{\mathrm{j_b}}_{\mathrm{j_b}}\right)\end{aligned} \tag{34}$$

where $\bar{\mathbf{I}}^{\mathrm{i}}_{\mathrm{i}}$ is the inertia matrix of body i, $\mathbf{r}^{\mathrm{i}}_{\mathrm{Ai,Bjb}}$ the vector pointing from joint $\mathrm{A_i}$ to the joint with link $\mathrm{j_b}$, and $\boldsymbol{\mu}^{\mathrm{j_b}}_{\mathrm{j_b}}$ the torque applied by the current body i on the succeding body $\mathrm{j_b}$ expressed in the coordinates of $\mathrm{j_b}$.

When applying this algorithm to the CHARMIE robot, the forward iterations can be completed for all of the links. However, since link 5 is preceded by a closed and overconstrained loop, the backwards iterations must stop upon reaching this body. To fully characterize the dynamics of links 4 and 5, additional formulations are required that define how the obtained sum of forces and torques is distributed along each of the preceding bodies.

## 5.2 Analysis of the Closed and Overconstrained Loops

The multibody dynamics analysis of overconstrained systems is a complex and continuously discussed problem in the literature. Several high-fidelity methods have been presented for dealing with these systems, such as in [35,36]. In the study of CHARMIE, the simplicity of the mechanisms, and the level of accuracy required from the results, allow some simplifications that permit a direct algebraic approach, solving these systems with minimal computational resources.

This section tackles the force and torque distribution in two of CHARMIE's links, body 4 and body 5 (see Figure 9). Since this study is a part of the backwards iterations of the recursive algorithm, body 5 must be tackled first.

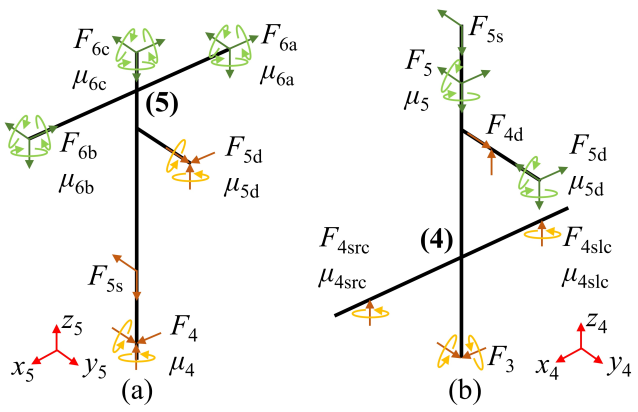

**Fig. 9** Free body diagram of CHARMIE's (a) body 5, and (b) body 4. The known forces and torque at the beginning of the study of each body are coloured in green hues, while unknown variables which will be calculated are coloured in orange hues.

Figure 9a shows the three links applying unknown forces and torques on body 5, namely the linear actuator $5_{\mathrm{d}}$, the extension spring $5_{\mathrm{s}}$, and the preceding link (body 4). The forces must be studied first since they influence the balance of torques.

The extension spring only applies a tangential force, which depends solely on its extension and orientation. The intensity $F_{5\mathrm{s}}$ of this force is determined with Equation 35.

$$F_{5\mathrm{s}} = k_s(L_0 - L_{\mathrm{s}}) + F_0\mathrm{s} \tag{35}$$

where $k_s$ is the spring constant, $L_0$ the spring's free length, $L_{\mathrm{s}}$ the spring's current length and $F_0$s the force originating from the pretension applied during the spring's manufacturing. The current spring length $L_{\mathrm{s}}$ and the orientation of the force can be determined using Equations 20-21.

From this point onward, to simplify the expressions, the force applied by spring $5_{\mathrm{s}}$ is treated similarly to those applied by succeeding bodies. To obtain the remaining forces, the following three conditions were taken into account:

- Since the joints of both the linear actuator and body 4 are revolute around the $x$ axis, their applied forces must guarantee the $x$ torque equilibrium;
- The force applied by the linear actuator in the $yz$ plane is tangential to the orientation of the actuator;
- Due to the similar construction of both joints, it was estimated that the linear actuator and body 4 apply half of the force reaction on the $x$ axis.

By equating these three conditions, and combining them with the sum of forces determined with Equation 33, Equation 36 was obtained.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & \mathbf{r}^5_{\mathrm{A5d,C5}}\hat{\mathbf{z}} & -\mathbf{r}^5_{\mathrm{A5d,C5}}\hat{\mathbf{y}} & 0 & \mathbf{r}^5_{\mathrm{A4,C5}}\hat{\mathbf{z}} & -\mathbf{r}^5_{\mathrm{A4,C5}}\hat{\mathbf{y}} \\ 0 & \cos(\alpha_{\mathrm{5d}}) & -\sin(\alpha_{\mathrm{5d}}) & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{f}^5_{\mathrm{5d}}\hat{\mathbf{x}} \\ \mathbf{f}^5_{\mathrm{5d}}\hat{\mathbf{y}} \\ \mathbf{f}^5_{\mathrm{5d}}\hat{\mathbf{z}} \\ \mathbf{f}^5_{4}\hat{\mathbf{x}} \\ \mathbf{f}^5_{4}\hat{\mathbf{y}} \\ \mathbf{f}^5_{4}\hat{\mathbf{z}} \end{bmatrix} \tag{36}$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ are unit vectors related to the three axes, $\mathbf{r}^5_{\mathrm{A5d,C5}}$ is the vector representing the displacement from the joint of the linear actuator $\mathrm{A}_{\mathrm{5d}}$ to the centre of mass $\mathrm{C}_5$, $\mathbf{r}^5_{\mathrm{A4,C5}}$ is the vector representing the displacement from the joint with body 4, $\mathrm{A}_4$, to the centre of mass $\mathrm{C}_5$, and $\alpha_{\mathrm{5d}}$ is the angle defining the rotation of the linear actuator in relation to body 5 around the $x_5$ axis. The left side of the equation system is defined by the variables:

$$\begin{aligned} a_1 &= \sum_{\mathrm{j_a}=1}^{2} \mathbf{f}^5_{\mathrm{j_a}}\hat{\mathbf{x}}, \qquad a_2 = \sum_{\mathrm{j_a}=1}^{2} \mathbf{f}^5_{\mathrm{j_a}}\hat{\mathbf{y}}, \qquad a_3 = \sum_{\mathrm{j_a}=1}^{2} \mathbf{f}^5_{\mathrm{j_a}}\hat{\mathbf{z}}, \\ a_4 &= \bar{\mathbf{I}}^5_5\dot{\boldsymbol{\omega}}^5_5\hat{\mathbf{x}} + \boldsymbol{\omega}^5_5 \times \left(\bar{\mathbf{I}}^5_5\boldsymbol{\omega}^5_5\right)\hat{\mathbf{x}} - \left(\mathbf{f}^5_{\mathrm{5s}} \times \mathbf{r}^5_{\mathrm{A5s,C5}}\right)\hat{\mathbf{x}} \\ &+ \sum_{\mathrm{j_b}=1}^{4} \left[\mathbf{R}^5_{\mathrm{j_b}}\mathbf{f}^{\mathrm{j_b}}_{\mathrm{j_b}} \times \left(\mathbf{r}^5_{\mathrm{A5,C5}} - \mathbf{r}^5_{\mathrm{A5,Bjb}}\right) + \mathbf{R}^5_{\mathrm{j_b}}\boldsymbol{\mu}^{\mathrm{j_b}}_{\mathrm{j_b}}\right]\hat{\mathbf{x}} \end{aligned} \tag{37}$$

where $a_1$, $a_2$, and $a_3$ represent the sum of the two unknown forces in the $x$, $y$, and $z$ axis calculated by using Equation 33 (including the spring force in the right side of the equation), and $a_4$ is the sum of torques of the system around the $x$ axis not considering the forces from the linear actuator and body 4.

Solving this equation system fully defines all forces applied in body 5. To calculate the unknown torques, one simplification was considered:

– Since the joints of the linear actuator and body 4 are similar, it was estimated that each applies half of the reaction torques around the $y$ and $z$ axes.

This simplification is expressed in two equations which, alongside the sum of torques around the $y$ and $z$ axis determined with Equation 34, result in the following formulation:

$$\begin{bmatrix} \sum_{\mathrm{ja}=1}^{2} \boldsymbol{\mu}^5_{\mathrm{ja}}\hat{\mathbf{y}} \\ \sum_{\mathrm{ja}=1}^{2} \boldsymbol{\mu}^5_{\mathrm{ja}}\hat{\mathbf{z}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}^5_{\mathrm{5d}}\hat{\mathbf{y}} \\ \boldsymbol{\mu}^5_{\mathrm{5d}}\hat{\mathbf{z}} \\ \boldsymbol{\mu}^5_{4}\hat{\mathbf{y}} \\ \boldsymbol{\mu}^5_{4}\hat{\mathbf{z}} \end{bmatrix} \tag{38}$$

By solving these systems of equations, the distribution of forces and torques for all bodies connected to link 5 is fully characterized.

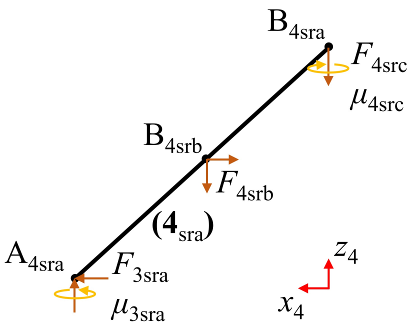

**Fig. 10** Free body diagram of CHARMIE's body $4_{\mathrm{sra}}$ to study the forces applied by the static balancing system between links 3 and 4.

Body 4 of CHARMIE has four sets of unknown forces and torques applied to it (see Figure 9b), which are exerted by the linear actuator $4_{\mathrm{d}}$, the symmetric components of the static balancing mechanisms $4_{\mathrm{src}}$ and $4_{\mathrm{slc}}$, and the preceding link (body 3).

The dynamics of body 4 were analysed by first independently addressing the static balancing mechanism (see Figure 10). This study is possible since this mechanism's forces depend only on the geometrical configuration of its spring and bars. Only the right side of the mechanism is illustrated, but the presented expressions are valid for both halves (the change in the direction of vectors automatically adjusts the equation for the respective side).

Force $F_{4\mathrm{srb}}$ is applied by an extension spring. The tangential component of this force is calculated using Equations 20 and 35. In turn, the rotation $\alpha_4$ of the spring around the $y$ axis in relation to body 4's reference is obtained using Equation 21. It is now possible to express the force applied by the spring in body $4_{\mathrm{sra}}$ as:

$$\mathbf{f}^4_{4\mathrm{sb}} = \left[F_{\mathrm{T4sb}}\mathrm{sin}(\alpha_4),\, 0,\, F_{\mathrm{T4s}}\mathrm{cos}(\alpha_4)\right] \tag{39}$$

where $F_{\mathrm{T4sb}}$ is the tangential force applied by the extension spring.

The unknown forces in the static balancing system can now all be determined from the balance of forces in the $x$ and $z$ axes, together with the balance of torques around the $y$ axis. It should be noted that accelerations are not relevant for this balance, since auxiliary bodies were simplified as being massless. These considerations result in the following formulation:

$$\begin{bmatrix} \mathbf{f}^4_{4\mathrm{srb}}\hat{\mathbf{x}} \\ \mathbf{f}^4_{4\mathrm{srb}}\hat{\mathbf{z}} \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \\ -\mathbf{r}^4_{\mathrm{A4sra,B4sra}}\hat{\mathbf{x}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{f}^4_{4\mathrm{src}}\hat{\mathbf{z}} \\ \mathbf{f}^4_{3\mathrm{sra}}\hat{\mathbf{x}} \\ \mathbf{f}^4_{3\mathrm{sra}}\hat{\mathbf{z}} \end{bmatrix} \tag{40}$$

where $\mathbf{r}^4_{\mathrm{A4sra,B4sra}}$ is the vector representing the displacement between the connecting points of body $4_{\mathrm{sra}}$ with links 3 and 4. The variable $a_5$ on the left side of Equation 40 is defined as:

$$a_5 = \left(\mathbf{f}^3_{4\mathrm{srb}}\hat{\mathbf{z}}\right)\left(\mathbf{r}^3_{\mathrm{A4sra,B4srb}}\hat{\mathbf{x}}\right) - \left(\mathbf{f}^3_{4\mathrm{srb}}\hat{\mathbf{x}}\right)\left(\mathbf{r}^3_{\mathrm{A4sra,B4srb}}\hat{\mathbf{z}}\right) \tag{41}$$

where $\mathbf{r}^3_{\mathrm{A4sra,B4srb}}$ is the vector representing the displacement between the connecting points of body $4_{\mathrm{sra}}$ with link 3 and the extension spring $4_{\mathrm{srb}}$.

Regarding the study of torques, body $4_{\mathrm{sra}}$ only transmits the torque applied to it around the $z$ axis directly from link 4 into link 3.

With the analysis of the static balancing mechanism concluded, the remaining forces and torques applied on body 4 can be approached. The following simplifications were taken into account:

– Actuator $4_{\mathrm{d}}$ applies a tangential force in the $yz$ plane aligned with the orientation of the actuator;
– The joint between bodies 4 and 3 possesses a much tighter tolerance than the static balancing mechanism, so it was estimated that this joint applies all undetermined reaction forces in the $x$ and $y$ axis.

This set of simplifications allows determining the currently unknown forces applied by both the linear actuator $4_{\mathrm{d}}$, and body 3, using the formulation:

$$\begin{bmatrix} \sum_{\mathrm{ja}=1}^{2} \mathbf{f}^4_{\mathrm{ja}}\hat{\mathbf{x}} \\ \sum_{\mathrm{ja}=1}^{2} \mathbf{f}^4_{\mathrm{ja}}\hat{\mathbf{y}} \\ \sum_{\mathrm{ja}=1}^{2} \mathbf{f}^4_{\mathrm{ja}}\hat{\mathbf{z}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ \cos(\alpha_{4\mathrm{d}}) & -\sin(\alpha_{4\mathrm{d}}) & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{f}^4_{4\mathrm{d}}\hat{\mathbf{y}} \\ \mathbf{f}^4_{4\mathrm{d}}\hat{\mathbf{z}} \\ \mathbf{f}^4_{3}\hat{\mathbf{x}} \\ \mathbf{f}^4_{3}\hat{\mathbf{y}} \end{bmatrix} \tag{42}$$

where the sum of unknown forces is obtained using Equation 33 (considering the static balancing forces on the right side of the equation), and $\alpha_{4\mathrm{d}}$ is the angle between the linear actuator and body 4 around the $x_4$ axis.

It is then possible to study the unknown torques applied in body 4. The following simplifications were taken into account:

– The joint between body 4 and 3 possesses a tighter tolerance than the static balancing mechanism and linear actuator, so it was estimated that this joint applies all undetermined reaction torques in the $x$ and $y$ axes;
– Since the joint with body 3 cannot apply any torque around the $z$ axis, and the static balancing mechanism joints are more rigid than the linear actuator ones, it was estimated that each of these joints applies half of the torque reaction in the $z$ axis.

These conditions result in the following system of equations:

$$\begin{bmatrix} \sum_{\mathrm{ja}=1}^{2} \boldsymbol{\mu}^4_{\mathrm{ja}}\hat{\mathbf{x}} \\ \sum_{\mathrm{ja}=1}^{2} \boldsymbol{\mu}^4_{\mathrm{ja}}\hat{\mathbf{y}} \\ \sum_{\mathrm{ja}=1}^{2} \boldsymbol{\mu}^4_{\mathrm{ja}}\hat{\mathbf{z}} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}^4_{3}\hat{\mathbf{x}} \\ \boldsymbol{\mu}^4_{3}\hat{\mathbf{y}} \\ \boldsymbol{\mu}^4_{3\mathrm{slr}}\hat{\mathbf{z}} \\ \boldsymbol{\mu}^4_{3\mathrm{slc}}\hat{\mathbf{z}} \end{bmatrix} \tag{43}$$

where the sum of torques are determined using Equation 34. With this analysis, the forces and torques applied to link 4 are determined.

With the distribution of all forces and torques applied on bodies 4 and 5 established, the recursive algorithm can conclude its iterations for the remaining bodies, finishing the full dynamic analysis of the CHARMIE robot.

**Table 5** Comparison of the average computational time required for 1000 timesteps of simulation (including the code initialization) for different models under the same conditions. Studies a) were conducted with no graphical interface, while studies b) illustrated the robot's motion over time.

| Model | 2-DOF Arm [s] | 4-DOF Arm [s] | 6-DOF Arm [s] | CHARMIE [s] |
|---|---|---|---|---|
| Kinematic a) | $0.211 \pm 0.002$ | $0.269 \pm 0.006$ | $0.325 \pm 0.006$ | $2.409 \pm 0.110$ |
| Dynamic a) | $0.482 \pm 0.002$ | $0.715 \pm 0.003$ | $0.947 \pm 0.002$ | $7.856 \pm 0.047$ |
| Kinematic b) | $35.810 \pm 0.194$ | $36.955 \pm 0.441$ | $37.672 \pm 0.086$ | $328.254 \pm 2.084$ |
| Dynamic b) | $36.140 \pm 0.239$ | $37.838 \pm 0.288$ | $38.080 \pm 0.191$ | $334.150 \pm 1.683$ |

## 6 Results and Discussion

The algorithms and mathematical equations described in this work were implemented into Python. The numpy library was used to assist in mathematical and algebraic operations, and the matplotlib library allowed the generation of graphics showing data and 3D plots representing the robot's motion. The resulting code is hardware independent and can be implemented into the robot's embedded computer.

Four different models were prepared to evaluate the algorithm's computational efficiency: a 2-DOF Arm, a 4-DOF Arm, a 6-DOF Arm and the CHARMIE Robot. These models were simulated in PyCharm on a computer with an AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz. Each model was simulated by analysing either only the kinematic, or both kinematic and dynamic properties. The animation of the motion was activated and deactivated to evaluate its computational weight, producing the results shown in Table 5.

The computational time of both the kinematic and dynamic formulations increases linearly with the number of bodies of the tested robotic arms, as expected from a recursive algorithm. This linearity cannot be extrapolated directly for the CHARMIE robot, since this analysis must also tackle closed kinematic loops. The animation of the model's motion, required for validation and testing by the researcher, but not for employing the simulator in its desired applications, consumes the greatest amount of computational resources when enabled, utilizing 99% of the computational time. These results also show that computational times can be reduced from 60% to 70% (the bigger the model, the greater the reduction) by disabling the inverse dynamics analysis.

The results from CHARMIE's kinematics and dynamics analyses were validated by comparing them with those obtained from a commercial software (Visual Nastran 4D). Figure 11 shows a side-by-side comparison of both studied models. Two main differences exist between the two simulations: the first is that in the commercial software, the robot was simplified by removing the closed and overconstrained loops; the second is that the commercial software calculates the inertia matrices directly from the primitive shapes and their associated mass, therefore, these estimated inertia matrices differ from those implemented in the recursive algorithm calculated from CAD models with greater detail.

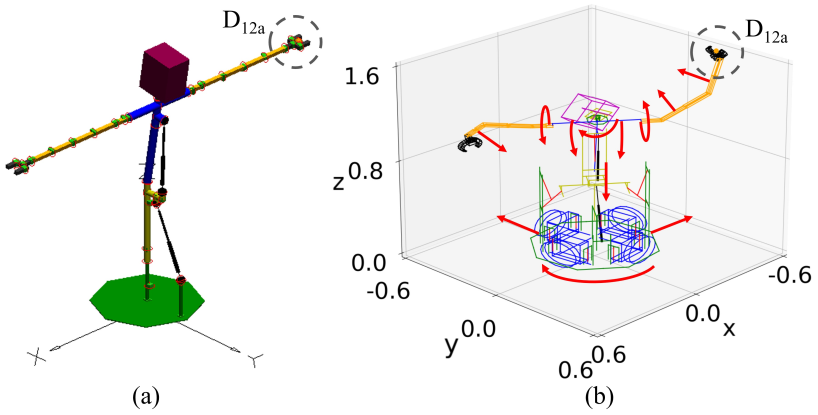

**Fig. 11** Multibody model of the CHARMIE robot in two different computational environments: (a) in a commercial software (Visual Nastran 4D) for the time frame t = 0s, (b) in the developed methodology for the time frame t = 1.3 s with red arrows representing the main directions of motion.

The same 5-second motion was implemented in both systems to allow a direct comparison of results. The goal of the chosen trajectory was not to be realistic (it includes impossible configurations in a physical prototype, such as penetrating bodies), but to be simple to compute and analyse. The movement starts from a reference position (Figure 11a) where all joints are in their 0 positions (except for the two linear actuators in the robot's body, with an initial total extension defined as 400 mm). A reference acceleration was then associated with all joints: every revolute joint (including the one defining the robot's yaw) has an angular acceleration of $\pi/9$ rad/s$^2$, the two linear actuators in the robot's body have an acceleration of 3 mm/s$^2$, and the two linear actuators controlling the robot's locomotion have an acceleration of 30 mm/s$^2$. The acceleration defined over time for each actuator in the simulations corresponds to the aforementioned reference accelerations multiplied by a sign function, which assumes a negative value for the first 2.5 seconds of motion, and a positive one for the last 2.5 seconds. The robot is fully stopped after its movement is concluded.

When implementing a recursive algorithm, any error will propagate along the iterative calculations. The part of the algorithm which defines the kinematics begins its analysis on the robot's base, and finishes in the end-effectors, therefore, the best point for evaluation will be one contained in the end-effectors. For this effect, point $D_{12a}$ was added to body 12a with the local coordinates of $[0, 0, 0.1023]$ m which corresponds to the centre of the end-effector claw (this point is highlighted with an orange sphere in Figure 11). Accelerations, velocities, positions and orientations are all interconnected, and mistakes in the velocities or accelerations would reflect in the results of the dynamic calculations, therefore, to validate the kinematic model, it is sufficient to evaluate the position of this point over time. The results from Figure 12

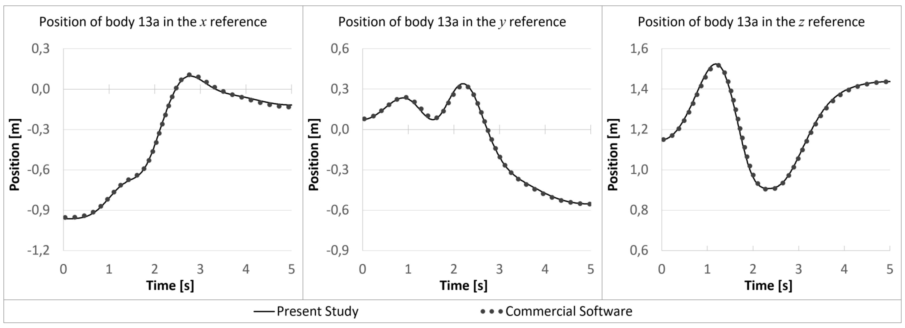

**Fig. 12** Positions of the same point in body 12a (the robot's left claw) obtained from the forward kinematics analysis using both a commercial software, and the methodology of the present study.

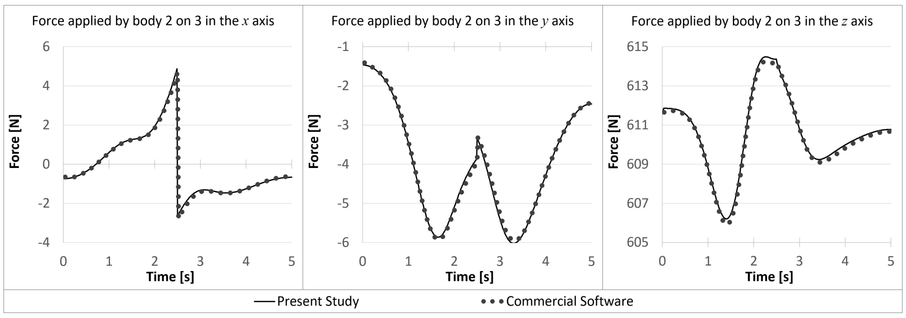

**Fig. 13** Forces applied by body 2 on body 3 (the robot's base) obtained from the inverse dynamic analysis using a commercial software, and the methodology of the present study.

show a complete overlap between the outcomes of both methods. A visual inspection of both simulations also showed identical behaviours.

The dynamics were studied by applying the same principle of error propagation as for the kinematics analysis. The comparison focuses on the forces applied by the floor plane on the robot's base which, considering the simplifications made to the locomotion system, corresponds to the loads applied by body 2 on body 3. The alterations made to the robot's configuration in the commercial software (to remove over-constrained loops) cause minimal changes in these results since they only affect how the forces propagate within the robot, not how it interacts with the floor plane. The two models are similar, yet not identical, so their dynamic properties regarding the generated torques cannot be compared directly. The juxtaposition of results in Figure 13 shows that both models produced identical results.

With two different sources providing identical results for the system's analysis, the implemented recursive algorithms were considered validated, as well as the seven-step methodology used for the CHARMIE project.

## 7 Concluding Remarks

This study presents a novel seven-step methodology for the multibody analysis of complex articulated robotic systems. This formulation is centered around two recursive algorithms, one for the forward kinematic analysis, and the other to model the inverse dynamics, which are well known from the literature. The methodology is focused on developing models tailored for the system being analysed, resulting in higher computational efficiency despite a loss in the generalizability. One of the main advantages of the used method is its modularity, allowing different mathematical notations to be used, as well as permitting an easy integration of other formulations to model more specific physical phenomena (such as applying external loads or simulating the locomotion using forward dynamics).

The seven-step methodology was further detailed by using it to successfully create the multibody model of the CHARMIE mobile manipulator. The robot's relative complexity illustrates the flexibility of the used method. This same robot was also modelled in a commercial software to confirm and validate the produced outcomes.

The obtained model is ideal for the current required analyses of CHARMIE. Nonetheless, a set of future works has been identified to characterise, improve and further validate the presented methodology. The computational efficiency of the used method can be further specified by calculating the number of mathematical operations required from each module of the program. This process should be accompanied by an optimization of the implementation of the current code in Python to maximize its computational efficiency. Separate formulations should be tested and compared within the modular structure of the seven-step methodology; besides increasing the computational efficiency, this may also improve the generelizability of the methodology without compromising its strengths. To further validate the results and the applicability of the developed simulator, it will be implemented on the embedded controller of the CHARMIE robot; the behaviour from the physical model will be compared with the predictions generated by the mathematical model.

With a finished and validated own-code solution for analysing CHARMIE, the results are now playing a pivotal role in a set of ongoing studies for the robot's development. The ability to alter geometries and properties using a single variable, which then updates the entire kinematics and dynamics accordingly, is being employed to generate quick automatic parametric optimizations. The current applications of this method include determining the required actuator torques for the arms, optimizing the static balancing mechanism associated with the linear actuators, and training a neural-network solution to control the robot's trajectories.

**Acknowledgements** This work has been supported by the Laboratory of Automation and Robotics (LAR) of University of Minho, and the ALGORITMI and CMEMS research centres. The first and second authors received funding through a doctoral scholarship from the Portuguese Foundation for Science and Technology [grant numbers SFRH/BD/145993/2019 and SFRH/BD/06944/2020], with funds from the Portuguese Ministry of Science, Technology and Higher Education and the European Social Fund through the Programa Operacional do Capital Humano (POCH). This work has been supported by FCT—Fundação para a Ciência e a Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

## Conflict of interest

The authors declare that they have no conflict of interest.

## References

1. Rudnicka, E., Napierała, P., Podfigurna, A., Męczekalski, B., Smolarczyk, R., Grymowicz, M.: The World Health Organization (WHO) approach to healthy ageing. Maturitas **139**(May), 6–11 (2020). DOI 10.1016/j.maturitas.2020.05.018. URL `https://linkinghub.elsevier.com/retrieve/pii/S0378512220302826`
2. Shishehgar, M., Kerr, D., Blake, J.: A systematic review of research into how robotic technology can help older people. Smart Health **7-8**(March), 1–18 (2018). DOI 10.1016/j.smhl.2018.03.002. URL `https://doi.org/10.1016/j.smhl.2018.03.002`
3. Ribeiro, T., Gonçalves, F., Garcia, I.S., Lopes, G., Ribeiro, A.F.: CHARMIE: A Collaborative Healthcare and Home Service and Assistant Robot for Elderly Care (2021). DOI 10.3390/app11167248
4. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics Modelling, Planning and Control, 1st edn. Springer-Verlag, London (2009). DOI 10.1007/978-1-84628-642-1
5. Ivaldi, S., Peters, J., Padois, V., Nori, F.: Tools for simulating humanoid robot dynamics: A survey based on user feedback. In: 2014 IEEE-RAS International Conference on Humanoid Robots, vol. 2015-Febru, pp. 842–849. IEEE (2014). DOI 10.1109/HUMANOIDS.2014.7041462. URL `http://ieeexplore.ieee.org/document/7041462/`
6. Gonçalves, F., Ribeiro, T., Ribeiro, A.F., Lopes, G., Flores, P.: A Recursive Algorithm for the Forward Kinematic Analysis of Robotic Systems Using Euler Angles. Robotics **11**(1), 15 (2022). DOI 10.3390/robotics11010015. URL `https://www.mdpi.com/2218-6581/11/1/15`
7. Luh, J.Y.S., Walker, M.W., Paul, R.P.C.: On-Line Computational Scheme for Mechanical Manipulators. Journal of Dynamic Systems, Measurement, and Control **102**(2), 69–76 (1980). DOI 10.1115/1.3149599. URL `https://asmedigitalcollection.asme.org/dynamicsystems/article/102/2/69/399118/OnLine-Computational-Scheme-for-Mechanical`
8. Featherstone, R.: The Calculation of Robot Dynamics Using Articulated-Body Inertias. The International Journal of Robotics Research **2**(1), 13–30 (1983). DOI 10.1177/027836498300200102. URL `http://journals.sagepub.com/doi/10.1177/027836498300200102`
9. Kim, J.H., Yang, J., Abdel-Malek, K.: A novel formulation for determining joint constraint loads during optimal dynamic motion of redundant manipulators in DH representation. Multibody System Dynamics **19**(4), 427–451 (2008). DOI 10.1007/s11044-007-9100-4. URL `http://link.springer.com/10.1007/s11044-007-9100-4`
10. Waldron, K., Schmiedeler, J.: Kinematics, pp. 9–33. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). DOI 10.1007/978-3-540-30301-5_2. URL `https://doi.org/10.1007/978-3-540-30301-5_2`
11. Aye, M.M.M.: Analysis of Euler angles in a simple two-axis gimbals set. World Academy of Science, Engineering and Technology **81**(9), 389–394 (2011). DOI 10.5281/zenodo.1330465

12. Hemingway, E.G., O'Reilly, O.M.: Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. Multibody System Dynamics **44**(1), 31–56 (2018). DOI 10.1007/s11044-018-9620-0. URL `http://link.springer.com/10.1007/s11044-018-9620-0`
13. Xu, X., Luo, J., Wu, Z.: The numerical influence of additional parameters of inertia representations for quaternion-based rigid body dynamics. Multibody System Dynamics **49**(3), 237–270 (2020). DOI 10.1007/s11044-019-09697-x. URL `http://link.springer.com/10.1007/s11044-019-09697-x`
14. Goldman, R.: Understanding quaternions. Graphical Models **73**(2), 21–49 (2011). DOI 10.1016/j.gmod.2010.10.004. URL `https://linkinghub.elsevier.com/retrieve/pii/S1524070310000172`
15. Huang, Z., Li, Q., Ding, H.: Basics of Screw Theory, pp. 1–16. Springer Netherlands, Dordrecht (2013). DOI 10.1007/978-94-007-4201-7_1. URL `https://doi.org/10.1007/978-94-007-4201-7_1`
16. Müller, A.: Screw and Lie group theory in multibody kinematics. Multibody System Dynamics **43**(1), 37–70 (2018). DOI 10.1007/s11044-017-9582-7. URL `http://link.springer.com/10.1007/s11044-017-9582-7`
17. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer US, Boston, MA (2008). DOI 10.1007/978-1-4899-7560-7. URL `http://link.springer.com/10.1007/978-1-4899-7560-7`
18. Gonçalves, F., Ribeiro, T., Ribeiro, A.F., Lopes, G., Flores, P.: Dynamic modeling of a human-inspired robot based on a newton-euler approach. In: A. Kecskeméthy, V. Parenti-Castelli (eds.) ROMANSY 24 - Robot Design, Dynamics and Control, pp. 79–90. Springer International Publishing, Cham (2022)
19. Ghaedrahmati, R., Raoofian, A., Kamali E., A., Taghvaeipour, A.: An enhanced inverse dynamic and joint force analysis of multibody systems using constraint matrices. Multibody System Dynamics **46**(4), 329–353 (2019). DOI 10.1007/s11044-019-09674-4. URL `http://dx.doi.org/10.1007/s11044-019-09674-4`
20. Emam, S.A.: Generalized Lagrange's equations for systems with general constraints and distributed parameters. Multibody System Dynamics **49**(1), 95–117 (2020). DOI 10.1007/s11044-019-09706-z. URL `http://link.springer.com/10.1007/s11044-019-09706-z`
21. Turno, S., Malczyk, P.: FPGA acceleration of planar multibody dynamics simulations in the Hamiltonian–based divide–and–conquer framework. Multibody System Dynamics **57**(1), 25–53 (2023). DOI 10.1007/s11044-022-09860-x. URL `https://link.springer.com/10.1007/s11044-022-09860-x`
22. Kane, T.R., Levinson, D.A.: Dynamics: Theory and Applications. McGraw-Hill, New York, NY (1985)
23. Müller, A.: Screw and Lie group theory in multibody dynamics. Multibody System Dynamics **42**(2), 219–248 (2018). DOI 10.1007/s11044-017-9583-6. URL `http://link.springer.com/10.1007/s11044-017-9583-6`
24. Khalil, W., Kleinfinger, J.F., Gautier, M.: Reducing the Computational Burden of the Dynamic Models of Robots. pp. 525–531 (1986). DOI 10.1109/robot.1986.1087680
25. Angeles, J., Ma, O., Rojas, A.: An algorithm for the inverse dynamics of n-axis general manipulators using Kane's equations. Computers & Mathematics with Applications **17**(12), 1545–1561 (1989). DOI 10.1016/0898-1221(89)90054-0. URL `https://linkinghub.elsevier.com/retrieve/pii/0898122189900540`
26. Featherstone, R.: A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 1: Basic Algorithm. The International Journal of Robotics Research **18**(9), 867–875 (1999). DOI 10.1177/02783649922066619. URL `http://journals.sagepub.com/doi/10.1177/02783649922066619`
27. Yamane, K., Nakamura, Y.: Comparative Study on Serial and Parallel Forward Dynamics Algorithms for Kinematic Chains*. The International Journal of Robotics Research **28**(5), 622–629 (2009). DOI 10.1177/0278364909102350. URL `http://journals.sagepub.com/doi/10.1177/0278364909102350`
28. Laflin, J.J., Anderson, K.S., Khan, I.M., Poursina, M.: New and Extended Applications of the Divide-and-Conquer Algorithm for Multibody Dynamics. Journal of Computational and Nonlinear Dynamics **9**(4) (2014). DOI 10.1115/1.

4027869. URL https://asmedigitalcollection.asme.org/computationalnonlinear/article/doi/10.1115/1.4027869/370274/New-and-Extended-Applications-of-the

29. Lee, C., Chang, P.: Efficient parallel algorithm for robot inverse dynamics computation. In: Proceedings. 1986 IEEE International Conference on Robotics and Automation, vol. 3, pp. 851–857. Institute of Electrical and Electronics Engineers (1986). DOI 10.1109/ROBOT.1986.1087560. URL http://ieeexplore.ieee.org/document/1087560/
30. Jain, A.: Robot and Multibody Dynamics, vol. 53. Springer US, Boston, MA (2011). DOI 10.1007/978-1-4419-7267-5. URL https://link.springer.com/10.1007/978-1-4419-7267-5
31. Bae, D.S., Haug, E.J.: A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems. Mechanics of Structures and Machines **15**(3), 359–382 (1987). DOI 10.1080/08905458708905124. URL http://www.tandfonline.com/doi/abs/10.1080/08905458708905124
32. Bae, D.S., Haug, E.J.: A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems. Mechanics of Structures and Machines **15**(4), 481–506 (1987). DOI 10.1080/08905458708905130. URL http://www.tandfonline.com/doi/abs/10.1080/08905458708905130
33. Kingsley, C., Poursina, M.: Extension of the divide-and-conquer algorithm for the efficient inverse dynamics analysis of multibody systems. Multibody System Dynamics **42**(2), 145–167 (2018). DOI 10.1007/s11044-017-9591-6. URL http://link.springer.com/10.1007/s11044-017-9591-6
34. Zhang, C.D., Song, S.M.: An efficient method for inverse dynamics of manipulators based on the virtual work principle. Journal of Robotic Systems **10**(5), 605–627 (1993). DOI 10.1002/rob.4620100505. URL https://onlinelibrary.wiley.com/doi/10.1002/rob.4620100505
35. Liu, W.L., Xu, Y.D., Yao, J.T., Zhao, Y.S.: Methods for Force Analysis of Overconstrained Parallel Mechanisms: A Review. Chinese Journal of Mechanical Engineering **30**(6), 1460–1472 (2017). DOI 10.1007/s10033-017-0199-9. URL http://link.springer.com/10.1007/s10033-017-0199-9
36. Liu, W., Xu, Y., Yao, J., Zhao, Y.: The weighted Moore–Penrose generalized inverse and the force analysis of overconstrained parallel mechanisms. Multibody System Dynamics **39**(4), 363–383 (2017). DOI 10.1007/s11044-016-9500-4. URL http://link.springer.com/10.1007/s11044-016-9500-4