Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Explicit Formulation for Mobility Analysis and Dynamics of General Multibody Systems

Yi Wang

631501537@qq.com

HiSilicon (Shanghai) Technologies CO., LTD

Dapeng Lao

HiSilicon (Shanghai) Technologies CO., LTD

Yongsheng Shao

HiSilicon (Shanghai) Technologies CO., LTD

Research Article

Keywords: Robot Mechanisms, Multibody systems, Mobility analysis, Dynamic analysis, Unified algorithm

Posted Date: October 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11192047/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Multibody System Dynamics manuscript No.
(will be inserted by the editor)

An Explicit Formulation for Mobility Analysis and Dynamics of General Multibody Systems

Yi Wang · Dapeng Lao · Yongsheng Shao

Received: date / Accepted: date

Abstract This paper proposes an explicit formulation for mobility analysis and dynamic simulation of general robot mechanisms, and the formulation is applicable to arbitrary multibody systems. To make the formulation applicable to arbitrary multibody systems, including serial, parallel, and hybrid serial–parallel mechanisms, we first introduce a universal description method for multibody systems. This method represents the position and orientation of a coordinate frame by four marker points defined on the frame. With one rigid body fixed, the constraint between two rigid bodies is then established using the velocity matrix of the four marker points on the coordinate frame associated with the other rigid body. Based on this description method, we derive an explicit formula for computing the mechanism mobility matrix, which describes the feasible motions of any rigid body in the robot mechanism. We further formulate the Lagrangian dynamic equations using the same representation and develop a general dynamic simulation algorithm. A dynamic simulation software is implemented based on the proposed universal mechanism description and simulation algorithm. A double-pendulum system is simulated using this tool, and the results agree well with those obtained from an open-source dynamic simulation package.

Keywords Robot Mechanisms · Multibody systems · Mobility analysis · Dynamic analysis · Unified algorithm

1 Introduction

Robots with classical serial and parallel mechanisms have been employed in manufacturing applications for decades. Today, robots are also used in many

Corresponding author. E-mail: wangyi279@huawei.com; 631501537@qq.com

Yi Wang, Dapeng Lao and Yongsheng Shao
HiSilicon (Shanghai) Technologies Co., Ltd., Room 101, 2 Hongqiaogang Road, Qingpu District, Shanghai, China

other areas, such as transportation, rehabilitation, domestic services, and scientific research. New multibody systems are continually being invented to meet the requirements of these applications. For each new mechanism, a large amount of work is usually required to develop algorithms for mobility analysis, kinematics, and dynamics. Most existing studies focus on a specific type of robot mechanism, and the resulting algorithms cannot be directly applied to other mechanism types. This limitation reduces the efficiency of developing new robot products.

To reduce repeated work, generalized methods for robot kinematics and dynamics modeling have been widely studied. Benhabib [1] and Kelmar [2] proposed generalized kinematics for serial modular robots using the Denavit-Hartenberg model. Chen [3,4] and Yang [5] proposed automatic modeling techniques for serial modular robots and branching modular robots based on the product-of-exponentials formula. Related work has also addressed general inverse kinematics and dynamics modeling for multibody systems [6–10]. However, previous studies usually consider only representative constraints, such as revolute pairs, prismatic pairs, and spherical joints. Different types of constraints are commonly described using variables with different physical meanings, which makes the modeling algorithms less flexible. For example, a guide-rail slider module is usually regarded as a prismatic pair. However, when the guide rail has a very small curvature, it becomes difficult to determine whether a revolute pair or a prismatic pair should be used to define the constraint. Therefore, it is important to develop a unified format for describing general constraints, including commonly used constraints such as revolute pairs, prismatic pairs, and spherical joints.

To obtain the kinematic and dynamic model of a general mechanism, a fundamental task is to compute the number of degrees of freedom of the mechanism. Gogu [11] presents a critical review of research on mechanism mobility. Among existing methods, the Grubler-Kutzbach criterion [12,13] is widely used. However, many mechanisms, such as the Delta robot [14], do not satisfy the Grubler-Kutzbach criterion. Kong and Gosselin [15] present a method for obtaining the mobility of a mechanism using the Jacobian matrix, but their work does not provide a unified formula for computing the Jacobian matrix of a general mechanism. Angeles and Gosselin [16] also studied the determination of the degree of freedom of kinematic chains. Huang et al. [17] present a general methodology for mobility analysis of mechanisms, in which screw theory is applied to analyze the mobilities of six representative puzzling mechanisms. A more general method for obtaining the number of degrees of freedom of a mechanism is presented by Lu, Zeng, and Huang [18,19]. However, obtaining explicit kinematic algorithms and dynamic equations for a user-defined general robot mechanism remains challenging.

The purpose of this paper is to provide a solution for mobility analysis and dynamics of a user-defined arbitrary robot mechanism or, more generally, a multibody system. The theoretical framework of the paper can be understood in three layers. First, a rigid body and its local coordinate frames are represented by four marker points, so that displacement, velocity, mass properties,

and constraints can be expressed in the same marker-point variables. Second, the feasible motion space of the whole mechanism is obtained by starting from a set of free bodies and then adding constraints one by one; each constraint removes incompatible motion directions through a null-space computation, and the remaining directions form the mobility matrix. Third, the same mobility matrix is used in the Lagrangian dynamic equation, so that the generalized acceleration, velocity, and displacement can be computed for time-domain simulation. The paper is organized as follows. Section 2 introduces a new representation of the position and orientation of a rigid body and a standard format for describing general constraints. Section 3 presents the mobility analysis of a general mechanism, including algorithms for computing the mobility matrix and its derivative. Section 4 introduces the Lagrangian dynamic equation and the simulation algorithm. Section 5 presents the development of a dynamic simulation software based on the proposed method, together with an example simulation task. Finally, Section 6 gives the conclusions and future work.

2 Modeling of Rigid Bodies and Constraints

This section introduces the common variables used throughout the paper. The main idea is to describe every rigid body by the positions and velocities of four fixed marker points. With this representation, different types of kinematic constraints can be written in a unified form: a velocity-level matrix specifies the allowed relative motion, while an error vector and its gradients specify the position-level constraint.

2.1 Description of the Displacement of a Rigid Body

A robot mechanism contains a set of bodies. To describe the position and orientation of each body, let $body_i$ represent the i-th body in a mechanism, and let $\{P_i\}$ represent the reference frame of $body_i$. Four marker points are attached to frame $\{P_i\}$, as shown in Fig. 1, so that the position and orientation of frame $\{P_i\}$ can be described by a **pose matrix** P_i.

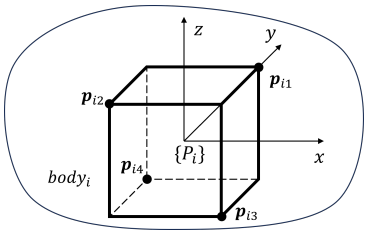


Fig. 1 Marker-point representation of a coordinate frame.

Here,

$$P_i = \begin{bmatrix} \boldsymbol{p}_{i1} & \boldsymbol{p}_{i2} & \boldsymbol{p}_{i3} & \boldsymbol{p}_{i4} \end{bmatrix} \in R^{3\times 4} \tag{1}$$

and $\boldsymbol{p}_{ik}(k = 1, 2, 3, 4)$ is the displacement vector of the k-th marker point of frame $\{P_i\}$.

For any frame $\{P_i\}$, the displacement vectors of the four marker points on $\{P_i\}$ **with respect to** (w.r.t.) $\{P_i\}$ are constant and are given by

$$\begin{aligned} \boldsymbol{p}_{i1}^{\{P_i\}} &= [1, 1, 1]^T \\ \boldsymbol{p}_{i2}^{\{P_i\}} &= [-1, -1, 1]^T \\ \boldsymbol{p}_{i3}^{\{P_i\}} &= [1, -1, -1]^T \\ \boldsymbol{p}_{i4}^{\{P_i\}} &= [-1, 1, -1]^T \end{aligned} \tag{2}$$

Let $P_j^{\{P_i\}}$ represent the relative pose matrix of frame $\{P_j\}$ w.r.t. $\{P_i\}$. Then, the mapping between the pose matrix $P_j^{\{P_i\}}$ and the commonly used homogeneous transformation matrix $T_{i,j}$ is bijective. Here,

$$T_{i,j} = \begin{bmatrix} R_{i,j} & \boldsymbol{p}_{i,j} \\ O \in R^{1\times 3} & 1 \end{bmatrix} \tag{3}$$

where $R_{i,j} \in R^{3\times 3}$ is the relative rotation matrix between frame $\{P_i\}$ and $\{P_j\}$, and $\boldsymbol{p}_{i,j}$ is the displacement vector of the origin of frame $\{P_j\}$ w.r.t. $\{P_i\}$.

The formulas for mapping between the pose matrix $P_j^{\{P_i\}}$ and the homogeneous transformation matrix $T_{i,j}$ are given as follows and can be easily verified.

Forward mapping:

$$P_j^{\{P_i\}} = R_{i,j}P_0 + \boldsymbol{p}_{i,j}[1, 1, 1, 1] \tag{4}$$

Inverse mapping:

$$\begin{aligned} \boldsymbol{p}_{i,j} &= \frac{1}{4}P_j^{\{P_i\}}[1, 1, 1, 1]^T \\ R_{i,j} &= \frac{1}{4}P_j^{\{P_i\}}P_0^T \end{aligned} \tag{5}$$

Specifically,

$$P_i^{\{P_i\}} = P_j^{\{P_j\}} = P_0 = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{bmatrix}, \tag{6}$$

where P_0 is referred to as the **unit pose matrix**, which corresponds to $T_{i,i} = T_{j,j} = I$. Therefore, the pose matrix is not an approximate description of a rigid-body pose. It contains the same information as the homogeneous transformation matrix, but stores it in a form that is convenient for constructing uniform velocity and constraint matrices.

The displacement of frame $\{P_i\}$ w.r.t. $\{P_j\}$ can also be represented by a **displacement vector**, defined as

$$\boldsymbol{x}_i^{\{P_j\}} = tovec\left(P_i^{\{P_j\}} - P_0\right) \in R^{12\times 1} \tag{7}$$

where $tovec(\cdot)$ is a function that reshapes a matrix into a vector column by column.

In this paper, $\boldsymbol{x}_i$ always represents $\boldsymbol{x}_i^{\{P_{i0}\}}$, i.e.,

$$\boldsymbol{x}_i = \boldsymbol{x}_i^{\{P_{i0}\}} = tovec\left(P_i^{\{P_{i0}\}} - P_0\right) \in R^{12\times 1} \tag{8}$$

It should be noted that $\{P_i\}$ is a time-varying frame, whereas $\{P_{i0}\}$ is a stationary frame that coincides with $\{P_i\}$ at the initial instant.

2.2 Description of the Velocity of a Rigid Body

The velocity of $body_i$ is represented by the velocities of the four marker points on its reference frame. Although this velocity vector has 12 components, a rigid body has only six independent velocity components. The role of the mobility matrix is to describe this six-dimensional rigid-body velocity subspace inside the 12-dimensional marker-point velocity space. The velocity of $\{P_i\}$ w.r.t. $\{P_{i0}\}$ is represented by the velocity vector

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} \dot{\boldsymbol{p}}_{i1}^{\{P_{i0}\}} \\ \dot{\boldsymbol{p}}_{i2}^{\{P_{i0}\}} \\ \dot{\boldsymbol{p}}_{i3}^{\{P_{i0}\}} \\ \dot{\boldsymbol{p}}_{i4}^{\{P_{i0}\}} \end{bmatrix} \tag{9}$$

To describe the velocity space of $body_i$, the **mobility matrix** is introduced. The mobility matrix of $body_i$ is defined so that the velocity vector of the four marker points of $\{P_i\}$ is always a linear combination of its columns. In this paper, $B_{P_i}^{\{P_j\}}$ denotes the mobility matrix of $\{P_i\}$ (or $body_i$) w.r.t. $\{P_j\}$. Specifically, if $body_i$ is a free body, its mobility matrix is given by

$$B_{P_i}^{\{P_{i0}\}} = B_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & -1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{10}$$

Let D_0 represent the null space of B_0. Each column of D_0 represents the displacement of the four marker points under pure deformation of a body. Let

$$C_{c_k} = nullspace([B_{c_k}, D_0]) \tag{11}$$

be the **constraint space matrix**. It can be easily verified that

$$C_{c_k}^T B_{c_k} = O \tag{12}$$

Let $v_i^{\{P_{i0}\}} \in R^{3\times 1}$ and $\omega_i^{\{P_{i0}\}} \in R^{3\times 1}$ be the linear velocity and angular velocity of $\{P_i\}$ w.r.t. $\{P_{i0}\}$, respectively. Then,

$$\dot{\boldsymbol{x}}_i = B_0 \begin{bmatrix} v_i^{\{P_{i0}\}} \\ \omega_i^{\{P_{i0}\}} \end{bmatrix} \tag{13}$$

Since B_0 has full column rank, the corresponding linear and angular velocities can be recovered from the marker-point velocity by the left inverse of B_0, i.e.,

$$\begin{bmatrix} v_i^{\{P_{i0}\}} \\ \omega_i^{\{P_{i0}\}} \end{bmatrix} = \left(B_0^T B_0\right)^{-1} B_0^T \dot{\boldsymbol{x}}_i.$$

Thus, B_0 is the bridge between the conventional twist representation and the marker-point velocity representation. The first three columns correspond to translational velocity, and the last three columns correspond to angular velocity.

An important issue is to determine the velocity relationship between two frames on the same body.If $\{P_i\}$ and $\{P_j\}$ are two frames on the same body, it can be shown that

$$\dot{\boldsymbol{x}}_j = H_{P_j^{\{P_i\}}} \dot{\boldsymbol{x}}_i \tag{14}$$

where

$$H_{P_j^{\{P_i\}}} = \begin{bmatrix} R_{j,i} & -R_{j,i}\hat{\boldsymbol{p}}_{j1}^{\{P_i\}} \\ R_{j,i} & -R_{j,i}\hat{\boldsymbol{p}}_{j2}^{\{P_i\}} \\ R_{j,i} & -R_{j,i}\hat{\boldsymbol{p}}_{j3}^{\{P_i\}} \\ R_{j,i} & -R_{j,i}\hat{\boldsymbol{p}}_{j4}^{\{P_i\}} \end{bmatrix} \left(B_0^T B_0\right)^{-1} B_0^T \tag{15}$$

is called the **mobility transformation matrix** between $\{P_i\}$ and $\{P_j\}$.

Proof:

The velocity of the k-th marker point of $\{P_j\}$ is given by

$$\begin{aligned} \dot{\boldsymbol{p}}_{jk}^{\{P_{j0}\}} &= R_{j,i}\dot{\boldsymbol{p}}_{jk}^{\{P_{i0}\}} \\ &= R_{j,i}\left[I \; -\hat{\boldsymbol{p}}_{jk}^{\{P_{i0}\}}\right] \begin{bmatrix} v_i^{\{P_{i0}\}} \\ \omega_i^{\{P_{i0}\}} \end{bmatrix} \\ &= \left[R_{j,i} \; -R_{j,i}\hat{\boldsymbol{p}}_{jk}^{\{P_{i0}\}}\right] \left(B_0^T B_0\right)^{-1} B_0^T \dot{\boldsymbol{x}}_i \end{aligned} \tag{16}$$

Therefore,

$$\dot{\boldsymbol{x}}_j = \begin{bmatrix} \dot{\boldsymbol{p}}_{j1}^{\{P_{j0}\}} \\ \dot{\boldsymbol{p}}_{j2}^{\{P_{j0}\}} \\ \dot{\boldsymbol{p}}_{j3}^{\{P_{j0}\}} \\ \dot{\boldsymbol{p}}_{j4}^{\{P_{j0}\}} \end{bmatrix} = H_{P_j^{\{P_i\}}} \dot{\boldsymbol{x}}_i \tag{17}$$

With (17), it is easily verified that

$$B_{P_j}^{\{P_{j0}\}} = H_{P_j^{\{P_i\}}} B_{P_i}^{\{P_{i0}\}} \tag{18}$$

2.3 Description of General Constraints

To make the proposed modeling method compatible with different types of kinematic constraints, a standard format for describing a general constraint is presented. In general, a constraint defines not only the permitted relative velocity between two bodies but also the permitted relative displacement between them. Let c_k represent the k-th constraint in a robot mechanism. The proposed standard description is

$$c_k = \{body_i, body_j, B_{c_k}, \boldsymbol{e}_{c_k}, \frac{\partial \boldsymbol{e}_{c_k}}{\partial x_i}, \frac{\partial \boldsymbol{e}_{c_k}}{\partial x_j}\} \tag{19}$$

Here, $body_i$ and $body_j$ are the two bodies constrained by c_k. $B_{c_k} \in R^{12 \times n_{c_k}}$ is the mobility matrix of $body_j$ when $body_i$ is fixed, and it describes the relative velocity between the two bodies. $\boldsymbol{e}_{c_k} \in R^{n_{c_k} \times 1}$ is a vector that describes the constraint errors. n_{c_k} is the number of degrees of freedom of joint c_k.

It should be noted that, when defining the constraints, we provide algorithms for computing B_{c_k}, $\boldsymbol{e}_{c_k}$, $\frac{\partial \boldsymbol{e}_{c_k}}{\partial x_i}$, and $\frac{\partial \boldsymbol{e}_{c_k}}{\partial x_j}$ instead of specifying their values directly. In this format, B_{c_k} is used in the mobility analysis, while $\boldsymbol{e}_{c_k}$ and its gradients are used to evaluate and correct position-level constraint errors during simulation.

In this paper, the standard descriptions of the **plane-point constraint** (PPC) is presented as an example.

To define a PPC between $body_i$ and $body_j$, a plane on $body_i$ and a point on $body_j$ should be specified. As shown in Fig. 2, $\boldsymbol{p}_a$ is a point in the plane on $body_i$, and $\boldsymbol{n}_a$ is the normal vector of the plane on $body_i$. $\boldsymbol{p}_b$ is the point on $body_j$. $\{P_a\}$ is a frame on $body_i$ whose origin coincides with $\boldsymbol{p}_a$ and whose z-axis is parallel to $\boldsymbol{n}_a$. $\{P_b\}$ is a frame on $body_j$ whose origin coincides with $\boldsymbol{p}_b$ and whose axes are parallel to those of $\{P_a\}$.

First, by definition, B_{c_k} is the mobility matrix of $\{P_j\}$ under the PPC and is given by

$$B_{c_k} = H_{P_j^{\{P_b\}}} B_{p_b}^{\{P_b\}} \tag{20}$$

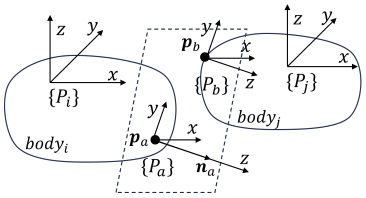


Fig. 2 Plane-point constraint between two bodies.

For point $\boldsymbol{p}_b$,

$$B_{p_b}^{\{P_b\}} = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & -1 & -1 & 0 \\ 1 & 0 & 0 & -1 & -1 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{21}$$

Second, since the PPC has one degree of freedom, $\boldsymbol{e}_{c_k}$ can be described by the projection of the vector $(\boldsymbol{p}_b - \boldsymbol{p}_a)$ onto $\boldsymbol{n}_a$, i.e.,

$$\boldsymbol{e}_{c_k} = \boldsymbol{n}_a^T \left(\boldsymbol{p}_b - \boldsymbol{p}_a\right) \tag{22}$$

Finally, the gradient of the constraint vector is given by

$$\begin{aligned} \frac{\partial \boldsymbol{e}_{c_k}}{\partial x_j} &= [0,0,1,0,0,0] \frac{\partial vw}{\partial x_b} \frac{\partial x_b}{\partial x_j} \\ &= [0,0,1,0,0,0] \left(B_0^T B_0\right)^{-1} B_0^T H_{P_b^{\{P_j\}}} \end{aligned} \tag{23}$$

and

$$\frac{\partial \boldsymbol{e}_{c_k}}{\partial x_i} = -\frac{\partial \boldsymbol{e}_{c_k}}{\partial x_j} H_{P_j^{\{P_i\}}} \tag{24}$$

This example shows how a user-defined constraint can be added to the proposed framework. Once its allowed relative velocity, constraint error, and error gradients are available, the same mobility and dynamic algorithms can be applied without deriving a new mechanism-specific formulation.

3 Mobility Analysis

The goal of mobility analysis is to obtain a basis for all feasible instantaneous motions of the mechanism. In the proposed representation, this basis is the mobility matrix A. Each column of A represents one independent motion direction of the complete mechanism, and any admissible marker-point velocity can be written as a linear combination of these columns.

3.1 Mobility Matrix

For a general robot mechanism with m bodies, the displacement of the robot mechanism is denoted by $\boldsymbol{x}$ and is defined as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_m \end{bmatrix} \in R^{12m\times 1} \tag{25}$$

where $\boldsymbol{x}_i$ is the displacement of $body_i$ w.r.t. $\{P_{i0}\}$.

The mobility of the robot mechanism is defined as the combination of all independent displacements of the robot mechanism. For a robot mechanism with n degrees of freedom, the mobility can be represented by a mobility matrix A given by

$$A = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} \in R^{12m\times n} \tag{26}$$

where $\boldsymbol{q}$ is the generalized displacement of the robot mechanism.

To obtain the mobility matrix A of a general robot mechanism, a three-step algorithm is presented.

Step 1. Obtain the mobility matrix of an m-body robot mechanism without constraints. Usually, a robot mechanism contains a body fixed to the ground, which is indexed as $body_0$ without loss of generality. Before considering any other constraints, the robot mechanism has one fixed body and $m-1$ free bodies, and thus its mobility matrix is given by

$$A_0 = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}_0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_0 & 0 & \cdots & 0 \\ 0 & B_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_0 \end{bmatrix} \in R^{12m\times 6(m-1)} \tag{27}$$

It should be noted that, if there is no ground-fixed body in the robot mechanism, a virtual ground-fixed body can be defined. This matrix is the largest possible motion space of the mechanism before the joint constraints are imposed.

Step 2. Obtain the relationship between A_k and A_{k-1} for $k = 1, 2, \cdots, n_c$. Here, n_c represents the total number of constraints;

$$A_k = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}_k} \tag{28}$$

represents the mobility matrix of the robot mechanism after the k-th constraint is added; $\boldsymbol{q}_k$ represents the generalized displacement of the robot mechanism after the k-th constraint is added.

According to (28), the relationship between A_k and A_{k-1} is given by

$$A_k = A_{k-1}\frac{\partial \boldsymbol{q_{k-1}}}{\partial \boldsymbol{q}_k} \tag{29}$$

To compute $\frac{\partial \boldsymbol{q_{k-1}}}{\partial \boldsymbol{q}_k}$, the definition of c_k should be considered. According to the definition of c_k, the relative displacement between $body_i$ and $body_j$ is a linear combination of the column vectors of B_{c_k}. Therefore,

$$B_{c_k}\boldsymbol{u} = \boldsymbol{x}_j - \boldsymbol{x}_i^{\{P_j\}} = \boldsymbol{x}_j - H_{P_{j0}^{\{P_{i0}\}}}\boldsymbol{x}_i \tag{30}$$

where $\boldsymbol{u}$ is an arbitrary vector. According to (12), we have

$$C_{c_k}^T\left(\boldsymbol{x}_j - H_{P_{j0}^{\{P_{i0}\}}}\boldsymbol{x}_i\right) = O \tag{31}$$

Then,

$$C_{c_k}^T\left(A_{(k-1)}[j]\boldsymbol{q}_{(k-1)} - H_{P_{j0}^{\{P_{i0}\}}}A_{(k-1)}[i]\boldsymbol{q}_{(k-1)}\right) = O \tag{32}$$

Here, $A_{(k-1)}[j]$ represents the submatrix of $A_{(k-1)}$ constructed from rows $12j$ to $12(j+1)$. This equation states that, after the k-th constraint is imposed, the relative motion between $body_i$ and $body_j$ must remain in the velocity space allowed by that constraint.

Equation (32) is satisfied for arbitrary $\boldsymbol{q}_k$. Therefore,

$$C_{c_k}^T\left(A_{(k-1)}[j] - H_{P_{j0}^{\{P_{i0}\}}}A_{(k-1)}[i]\right)\frac{\partial \boldsymbol{q_{k-1}}}{\partial \boldsymbol{q}_k} = 0 \tag{33}$$

Then, the formula for computing $\frac{\partial \boldsymbol{q_{k-1}}}{\partial \boldsymbol{q}_k}$ is obtained, i.e.,

$$\frac{\partial \boldsymbol{q_{k-1}}}{\partial \boldsymbol{q}_k} = nullspace\left(C_{c_k}^T\left(A_{k-1}[j] - H_{P_{j0}^{\{P_{i0}\}}}A_{k-1}[i]\right)\right) \tag{34}$$

Step 3. Obtain the mobility matrix of the robot mechanism with all n_c constraints, i.e.,

$$A = A_{n_c} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}_0}\frac{\partial \boldsymbol{q}_0}{\partial \boldsymbol{q}_1}\frac{\partial \boldsymbol{q}_1}{\partial \boldsymbol{q}_2}\cdots\frac{\partial \boldsymbol{q}_{n_c-1}}{\partial \boldsymbol{q}_{n_c}} \tag{35}$$

The final matrix A gives both the number of independent motions and their directions. In particular, the number of columns of A is the mobility of the mechanism, and the relation $\dot{\boldsymbol{x}} = A\dot{\boldsymbol{q}}$ maps generalized velocity to the velocity of all marker points.

3.2 Derivative of the Mobility Matrix

Computing the derivative of the mobility matrix is important because $\dot{A}$ is required in dynamic simulations. Since A depends on the current mechanism configuration, it changes during motion. The following derivation expresses this change using the current mobility matrix, the mobility matrix from the previous simulation step, and the current marker-point velocities. According to the definition of the mobility matrix,

$$\begin{aligned}\dot{A} &= \frac{d}{dt}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}\right) \\ &= \lim_{-\delta t \to 0} \frac{1}{\delta t}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} - \frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}}\right) \\ &= \lim_{-\delta t \to 0} \frac{1}{\delta t}\frac{\partial}{\boldsymbol{q}}\left(\frac{\boldsymbol{x} - \boldsymbol{x}^{-\delta t}}{\delta t}\right) \\ &= \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{q}}\end{aligned} \tag{36}$$

According to (36), both $\boldsymbol{x}$ and $\boldsymbol{x}^{-\delta t}$ are dependent variables of $\boldsymbol{q}$, whereas $\dot{\boldsymbol{x}}$ is a dependent variable of $\boldsymbol{x}$ and $\boldsymbol{x}^{-\delta t}$. Therefore,

$$\dot{A} = \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} + \lim_{-\delta t \to 0} \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}^{-\delta t}}\frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}} \tag{37}$$

In (37), $\frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}}$ can be computed using

$$\frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{x}}_1}{\partial \boldsymbol{x}_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial \dot{\boldsymbol{x}}_2}{\partial \boldsymbol{x}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \boldsymbol{x}_m} \end{bmatrix} \tag{38}$$

where

$$\frac{\partial \dot{\boldsymbol{x}}_i}{\partial \boldsymbol{x}_i} = \begin{bmatrix} O_{\in R^{3\times 3}} & [\dot{\boldsymbol{x}}_{i1}]_\times \\ O_{\in R^{3\times 3}} & [\dot{\boldsymbol{x}}_{i2}]_\times \\ O_{\in R^{3\times 3}} & [\dot{\boldsymbol{x}}_{i3}]_\times \\ O_{\in R^{3\times 3}} & [\dot{\boldsymbol{x}}_{i4}]_\times \end{bmatrix} (B_0^T B_0)^{-1} B_0^T \in R^{12\times 12} \tag{39}$$

In addition, from (37) we have

$$\begin{aligned}&\lim_{-\delta t \to 0} \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}^{-\delta t}}\frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}} \\ &= \lim_{-\delta t \to 0} \frac{1}{\delta t}\frac{\partial(\boldsymbol{x} - \boldsymbol{x}^{-\delta t})}{\partial \boldsymbol{x}^{-\delta t}}\frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}} \\ &= \lim_{-\delta t \to 0} \frac{1}{\delta t}\left(A - \frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}}\right)\end{aligned} \tag{40}$$

where

$$\frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}^{-\delta t}} \frac{\partial \boldsymbol{q}^{-\delta t}}{\partial \boldsymbol{q}} = A^{-\delta t} \frac{\partial \boldsymbol{q}^{-\delta t}}{\partial \boldsymbol{q}} \tag{41}$$

in which $\frac{\partial \boldsymbol{q}^{-\delta t}}{\partial \boldsymbol{q}}$ is obtained using

$$\frac{\partial \boldsymbol{q}^{-\delta t}}{\partial \boldsymbol{q}} = \left(A^{-\delta t^T} A^{-\delta t}\right)^{-1} A^{-\delta t^T} A \tag{42}$$

Substituting (40) and (41) into (37) yields

$$\begin{aligned} \dot{A} &= \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}} A + \frac{1}{\delta t}\left(A - \frac{\partial \boldsymbol{x}^{-\delta t}}{\partial \boldsymbol{q}}\right) \\ &= \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}} A + \frac{1}{\delta t}\left(A - A^{-\delta t} \frac{\partial \boldsymbol{q}^{-\delta t}}{\partial \boldsymbol{q}}\right) \\ &= \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}} A + \frac{1}{\delta t}(A - A^{-\delta t}\left(A^{-\delta t^T} A^{-\delta t}\right)^{-1} A^{-\delta t^T} A) \end{aligned} \tag{43}$$

For numerical simulations, δt can be set as the simulation period, and $A^{-\delta t}$ is the mobility matrix of the robot mechanism in the previous simulation period. Therefore, all the terms in (43) can be computed, and the derivative of the mobility matrix $\dot{A}$ is obtained. The first term in (43) accounts for the dependence of marker-point velocity on the current pose, while the second term accounts for the change of the mobility basis between two consecutive time steps.

4 Dynamic Analysis

This section formulates the dynamic equation using the same marker-point variables and mobility matrix introduced above. The key step is to express the kinetic energy of each rigid body by equivalent mass points and then substitute $\dot{\boldsymbol{x}} = A\dot{\boldsymbol{q}}$ into the Lagrange equation.

4.1 Dynamic Parameters of a Body

To describe the dynamic parameters of a body, let $\{P_{mi}\}$ be a frame whose origin is at the center of gravity of $body_i$ and whose axes coincide with the principal axes of inertia of $body_i$. As shown in Fig. 3, four mass points $\boldsymbol{p_{mi1}}$, $\boldsymbol{p_{mi2}}$, $\boldsymbol{p_{mi3}}$ and $\boldsymbol{p_{mi4}}$ are attached to the body, and the mass of each point is $\frac{1}{4}m_i$, where m_i is the total mass of $body_i$.

The local displacements of the four mass points w.r.t. $\{P_{mi}\}$ are given by

$$\begin{bmatrix}\boldsymbol{p}_{mi1} & \boldsymbol{p}_{mi2} & \boldsymbol{p}_{mi3} & \boldsymbol{p}_{mi4}\end{bmatrix} = \begin{bmatrix} d_x & -d_x & d_x & -d_x \\ d_y & -d_y & -d_y & d_y \\ d_z & d_z & -d_z & -d_z \end{bmatrix} \tag{44}$$

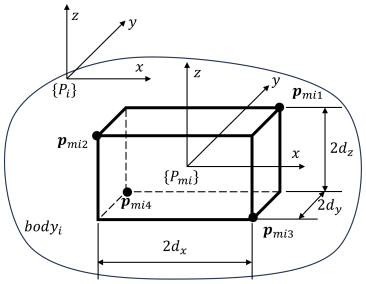


Fig. 3 Definition of the four mass points of $body_i$.

where

$$\begin{aligned} d_x &= \sqrt{(-I_{xx} + I_{yy} + I_{zz})/(2m_i)} \\ d_y &= \sqrt{(I_{xx} - I_{yy} + I_{zz})/(2m_i)} \\ d_z &= \sqrt{(I_{xx} + I_{yy} - I_{zz})/(2m_i)} \end{aligned} \tag{45}$$

It can be easily verified that the four mass points have the same equivalent mass, inertia, and kinetic energy as the body. Therefore, the dynamic parameters of a rigid body can also be represented in the marker-point format. This is useful because the kinetic energy can then be written directly as a quadratic form of marker-point velocities.

4.2 Dynamic Equation of the Robot Mechanism

The Lagrange equation is given by

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial (E_k - E_p)}{\partial \boldsymbol{q}} = 0 \tag{46}$$

where

$$E_k = \sum_{i=1}^{m} E_{ki} \tag{47}$$

is the sum of the kinetic energy of all bodies, while

$$E_p = \sum_{i=1}^{m} E_{pi} \tag{48}$$

is the sum of the potential energy of all bodies.

The kinetic energy of $body_i$ can be represented by

$$E_{ki} = \frac{1}{2}\dot{\boldsymbol{x}}_{mi}^T M_i \dot{\boldsymbol{x}}_{mi} \tag{49}$$

where

$$\dot{\boldsymbol{x}}_{mi} = \begin{bmatrix} \dot{\boldsymbol{p}}_{mi1} \\ \dot{\boldsymbol{p}}_{mi2} \\ \dot{\boldsymbol{p}}_{mi3} \\ \dot{\boldsymbol{p}}_{mi4} \end{bmatrix} \in R^{12\times 1} \tag{50}$$

is the velocity vector of the four mass points of $body_i$, and

$$M_i = \frac{1}{4} m_i I_{12} \in R^{12\times 12} \tag{51}$$

is called the mass matrix of $body_i$.

In (46),

$$\begin{aligned} \frac{\partial E_k}{\partial \dot{\boldsymbol{q}}} &= \frac{\partial E_k}{\partial \dot{\boldsymbol{x}}_m} \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \frac{\partial \dot{\boldsymbol{x}}}{\partial \dot{\boldsymbol{q}}} \\ &= \dot{\boldsymbol{x}}_m^T M \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \frac{\partial \dot{\boldsymbol{x}}}{\partial \dot{\boldsymbol{q}}} \\ &= \dot{\boldsymbol{q}}^T \frac{\partial \dot{\boldsymbol{x}}}{\partial \dot{\boldsymbol{q}}}^T \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}}^T M \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \frac{\partial \dot{\boldsymbol{x}}}{\partial \dot{\boldsymbol{q}}} \\ &= \dot{\boldsymbol{q}}^T A^T M_{ref} A \end{aligned} \tag{52}$$

where

$$M_{ref} = \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}}^T M \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \tag{53}$$

is the mass matrix of the robot mechanism, where

$$M = \begin{bmatrix} M_1 & O & \dots & O \\ O & M_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & M_m \end{bmatrix} \tag{54}$$

Thus, $A^T M_{ref} A$ can be regarded as the generalized mass matrix of the constrained mechanism.

Therefore,

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\boldsymbol{q}}}\right) = \ddot{\boldsymbol{q}}^T A^T M_{ref} A + \dot{\boldsymbol{q}}^T \dot{A}^T M_{ref} A + \dot{\boldsymbol{q}}^T A^T M_{ref} \dot{A} \tag{55}$$

In addition, from (46), it can be derived that

$$\begin{aligned} \frac{\partial E_k}{\partial \boldsymbol{q}} &= \frac{\partial E_k}{\partial \dot{\boldsymbol{x}}_m} \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{q}} \\ &= \dot{\boldsymbol{x}}_m^T M \frac{\partial \dot{\boldsymbol{x}}_m}{\partial \dot{\boldsymbol{x}}} \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{q}} \\ &= \dot{\boldsymbol{x}}^T M_{ref} \dot{A} \end{aligned} \tag{56}$$

Using the principle of virtual work, we have

$$\begin{aligned}\frac{\partial E_p}{\partial \boldsymbol{q}} &= \frac{\partial E_p}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} \\ &= -\boldsymbol{f}^T A\end{aligned} \tag{57}$$

where

$$\boldsymbol{f} = \begin{bmatrix} f_1{}^{\{P_1\}} \\ f_2{}^{\{P_2\}} \\ \vdots \\ f_m{}^{\{P_m\}} \end{bmatrix} \tag{58}$$

represents the force on the marker points of each body w.r.t. the reference frame of that body.

Substituting (55), (56), and (57) into (46), the generalized acceleration of the robot mechanism $\ddot{\boldsymbol{q}}$ can be computed. At this point, all terms except $\ddot{\boldsymbol{q}}$ are known from the current state, the mobility matrix, its derivative, and the applied forces. After $\ddot{\boldsymbol{q}}$ is obtained, a numerical integration algorithm can be applied to compute the generalized velocity and generalized displacement of the robot mechanism for the next simulation period, i.e., $\dot{\boldsymbol{q}}^{\delta t}$ and $\boldsymbol{q}^{\delta t}$. Then,

$$\dot{\boldsymbol{x}}^{\delta t} = A\dot{\boldsymbol{q}}^{\delta t} + \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}\delta t \tag{59}$$

and

$$\boldsymbol{x}^{\delta t} = A\boldsymbol{q}^{\delta t} \tag{60}$$

The pose matrix of $body_i$ in the next period is given by

$$P_i^{\delta t} = P_i + R_i tomat(\boldsymbol{x}_i^{\delta t}) \tag{61}$$

where $tomat(\cdot)$ is the inverse function of $tovec(\cdot)$ and reshapes a 12×1 vector into a 3×4 matrix column by column.

4.3 Displacement Rectification Algorithm

In each simulation period, the displacement of the marker points of each frame is obtained using numerical integration. Due to numerical integration error and the nonlinearity of the system, the new pose matrix of each body may not satisfy the constraints exactly. Therefore, a rectification step is required after each simulation period.

Two types of errors should be corrected. First, the distance between any two marker points in one frame should remain $2\sqrt{2}$. Let

$$\epsilon_i = \begin{bmatrix} \|\boldsymbol{p}_{i1} - \boldsymbol{p}_{i2}\| - 2\sqrt{2} \\ \|\boldsymbol{p}_{i1} - \boldsymbol{p}_{i3}\| - 2\sqrt{2} \\ \|\boldsymbol{p}_{i1} - \boldsymbol{p}_{i4}\| - 2\sqrt{2} \\ \|\boldsymbol{p}_{i2} - \boldsymbol{p}_{i3}\| - 2\sqrt{2} \\ \|\boldsymbol{p}_{i2} - \boldsymbol{p}_{i4}\| - 2\sqrt{2} \\ \|\boldsymbol{p}_{i3} - \boldsymbol{p}_{i4}\| - 2\sqrt{2} \end{bmatrix} \in R^{6\times 1} \tag{62}$$

be the distance error between each pair of marker points in the reference frame of $body_i$, and

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix} \in R^{6m\times 1} \tag{63}$$

be the distance errors of all bodies. Then,

$$\frac{\partial \epsilon}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \epsilon_1}{\partial \boldsymbol{x}_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial \epsilon_2}{\partial \boldsymbol{x}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \epsilon_m}{\partial \boldsymbol{x}_m} \end{bmatrix} \in R^{6m\times 12m} \tag{64}$$

Second, the constraint error of each constraint should be zero. Let ε_k represent the constraint error of the k-th joint, and

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_{n_c} \end{bmatrix} \tag{65}$$

be the constraint errors of the n_c constraints in the mechanism. $\frac{\partial \varepsilon}{\partial \boldsymbol{x}}$ is given by

$$\frac{\partial \varepsilon}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \varepsilon_1}{\partial \boldsymbol{x}} \\ \frac{\partial \varepsilon_2}{\partial \boldsymbol{x}} \\ \vdots \\ \frac{\partial \varepsilon_{n_c}}{\partial \boldsymbol{x}} \end{bmatrix} \tag{66}$$

The algorithm for rectifying the pose matrix of the reference frame of each body can then be obtained. The pose-matrix rectification algorithm is given by

$$\boldsymbol{x}_{rectified} = \boldsymbol{x} - \begin{bmatrix} \frac{\partial \epsilon}{\partial \boldsymbol{x}} \\ \frac{\partial \varepsilon}{\partial \boldsymbol{x}} \end{bmatrix}^{\#} \begin{bmatrix} \epsilon \\ \varepsilon \end{bmatrix} \tag{67}$$

Here, $(\cdot)^{\#}$ denotes the pseudoinverse of a matrix. This step can be interpreted as projecting the numerically integrated pose back toward the manifold defined by rigid-body shape constraints and joint constraints.

The rectified pose matrix of $body_i$ is given by

$$P_{irectified} = P_i + R_i \cdot tomat(\boldsymbol{x}_{irectified}) \tag{68}$$

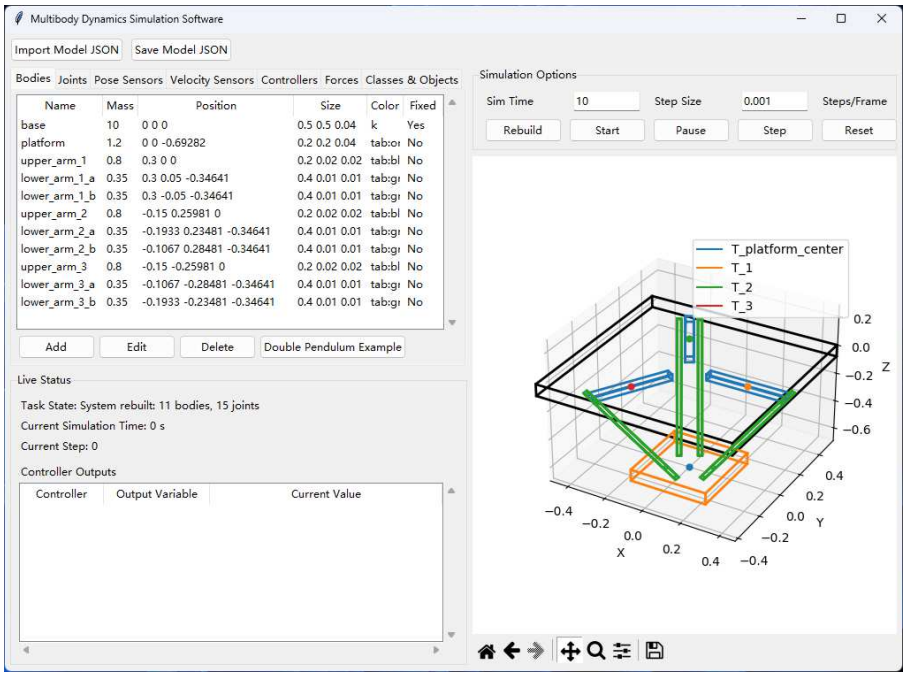


Fig. 4 The graphical user interface of the multibody dynamic simulation software.

5 Development of a multibody Dynamic Simulation Software

5.1 Introduction to the multibody Dynamic Simulation Software

Based on the formulation presented in the previous sections, a multibody dynamic simulation software has been developed in Python. The purpose of this tool is to provide an integrated environment for building multibody systems, defining dynamic parameters, executing time-domain simulations, and visualizing the resulting motion. The software combines the dynamic analysis library with a graphical user interface, so that a user can construct a mechanism without directly writing all simulation scripts. It supports the definition of rigid bodies, kinematic constraints, external forces, user-defined controllers, sensors, and simulation options.

Figure 4 shows the graphical interface of the simulation software. At the modeling level, each rigid body can be specified by its name, mass, principal moments of inertia, initial position, initial orientation, display size, color, and fixed-ground condition. The current version provides several common constraint types, including spherical joints, revolute joints, plane-point constraints, and plane-plane constraints.

External forces can be applied in either the world frame or the body frame. Instead of only accepting constant force vectors, the software allows forces to be written as Python expressions. These Python expressions can use simulation

variables, such as the current time, time step, total simulation time, and step number. They can also use sensor outputs and controller outputs, which makes it possible to define feedback forces and time-varying actuation laws.

The controller module allows the user to write custom Python functions. A controller can read pose sensors, velocity sensors, and previously computed controller outputs, and then return scalar or vector outputs that are used by force expressions. In addition, a separate script editor is provided for defining shared classes, objects, and constants before the simulation starts, which is useful for controllers with internal states, such as PID controllers.

The graphical interface is organized around the normal workflow of a simulation task. First, the user creates or imports a model and defines all rigid bodies. Second, the user adds the joints or constraints between the bodies. Third, external forces, controllers, and sensors are configured according to the purpose of the simulation. Fourth, the simulation time, integration time step, and number of steps per visual frame are specified. After the model is rebuilt, the user can run the simulation continuously, pause it, reset it, or advance it by a single step. During the simulation, the three-dimensional visualization window displays the current configuration of the mechanism and the trajectories recorded by the pose sensors. The live status panel reports the current task state, simulation time, step number, and selected controller outputs.

For model management and post-processing, the software supports JSON model import and export. Therefore, the software serves both as a verification platform for the proposed dynamic algorithm and as a practical tool for constructing and testing multibody robotic mechanisms.

5.2 An Example Simulation Task

The software provides two example models, a Delta-robot with three degrees of freedom and a double pendulum model. Here a simulation of the free motion of a double pendulum is presented.

The double pendulum consists of three rigid bodies, where $body_0$ is fixed to the ground, while $body_1$ and $body_2$ are two rods. The geometrical and dynamic parameters of $body_1$ and $body_2$ are given in Table 1.

Table 1 Dynamic parameters of the bodies

Body index	Mass	COM	Inertia	Size
$body_1$	1	[0,0,0]	$I_{R^3\times 3}$	[2,0.1,0.1]
$body_2$	1	[0,0,0]	$I_{R^3\times 3}$	[2,0.1,0.1]

There are two spherical joints in the double pendulum. The first spherical joint is located at $[0, 0, 0]$ and connects $body_0$ and $body_1$, while the other spherical joint is located at $[2, 0, 0]$ and connects $body_1$ and $body_2$.

The simulation condition is defined as follows. First, the gravitational acceleration is 9.8 m/s^2. Second, the time step of this simulation task is 0.001

s, and the total simulation time is 10 s. Finally, the same simulation task is also conducted in Exudyn, an open-source dynamic simulation library with a Python interface.

During the simulation, the displacement of the end of the double pendulum is recorded. The trajectory of the end of the double pendulum is shown in Fig. 5.

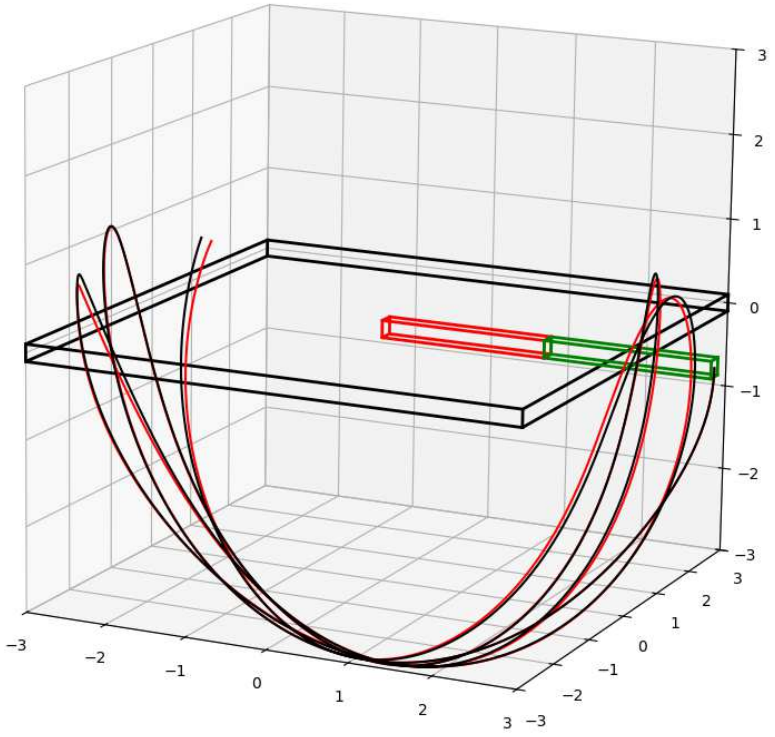


Fig. 5 Trajectory of the endpoint of the double pendulum.

The simulation results show that the proposed simulation software and the open-source simulation tool provide nearly identical results.

6 Conclusions and Future Work

This paper introduces a unified formulation for solving the simulation problem of a general robot mechanism. The proposed marker-point representation

provides a common format for describing rigid-body poses, constraints, mobility, and dynamic equations. Based on this representation, explicit formulas for mobility analysis and dynamic simulation are derived. An example involving a double pendulum demonstrates that the proposed software can produce simulation results consistent with those of an open-source dynamic simulation package. Future work will focus on improving computational efficiency, particularly by reducing the cost of repeated null-space computations.

References

1. B. Benhabib, G. Zak, M. G. Lipton, A generalized kinematic modelling method for modular robots, *J. Rob. Syst.* 6(5) (1989) 545–571.
2. L. Kelmar, P. K. Khosla, Automatic generation of forward and inverse kinematics for a reconfigurable modular manipulator system, *J. Rob. Syst.* 7(4) (1990) 599–619.
3. I. M. Chen, G. Yang, Configuration independent kinematics for modular robots, in: *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, USA, 1996, pp. 1845–1849.
4. I. M. Chen, S. H. Yeo, G. Chen, G. Yang, Kernel for modular robot applications - automatic modeling techniques, *Int. J. Rob. Res.* 18(2) (1999) 225–242.
5. G. Yang, Kinematics, dynamics, calibration, and configuration optimization of modular reconfigurable robots, Ph.D. Thesis, School of Mechanical and Production Engineering, Nanyang Technological University, Singapore, 1999.
6. A. J. Elias, J. T. Wen, IK-Geo: Unified robot inverse kinematics using subproblem decomposition, *Mech. Mach. Theory* 209 (2025) 105971. `https://doi.org/10.1016/j.mechmachtheory.2025.105971`
7. J. Leoro, T. Hsiao, C. Betancourt, A new geometric subproblem to extend solvability of inverse kinematics based on screw theory for 6R robot manipulators, *Int. J. Control Autom. Syst.* 19(1) (2021) 562–573.
8. T. Geike, J. McPhee, On the automation generation of inverse dynamic solution for parallel manipulators, in: *Proc. Workshop Fundam. Issues Future Res. Dir. Parallel Mech. Manipulators*, 2002, pp. 348–358.
9. T. D. Thanh, J. Kotlarski, B. Heimann, T. Ortmaier, On the inverse dynamics problem of general parallel robots, in: *Proc. IEEE Int. Conf. Mechatronics*, Malaga, 2009, pp. 1–6. `https://doi.org/10.1109/ICMECH.2009.4957202`
10. W. Khalil, O. Ibrahim, General solution for the dynamic modeling of parallel robots, *J. Intell. Rob. Syst.* 49 (2007) 19–37.
11. G. Gogu, Mobility of mechanisms: a critical review, *Mech. Mach. Theory* 40(9) (2005) 1068–1097.
12. M. Grübler, Allgemeine Eigenschaften der zwangläufigen ebenen kinematischen Kette: I, *Civilingenieur* 29 (1883) 167–200.
13. K. Kutzbach, Mechanische leitungsverzweigung, ihre gesetze und anwendungen, *Maschinenbau* 8(21) (1929) 710–716.
14. R. Clavel, DELTA, a fast robot with parallel geometry, in: *Proc. 18th Int. Symp. Ind. Rob.*, Springer-Verlag, New York, 1988, pp. 91–100.
15. X. Kong, C. M. Gosselin, Type synthesis of linear translational parallel manipulators, in: *Adv. Robot. Kinemat.*, Springer, Dordrecht, Netherlands, 2002, pp. 453–462.
16. J. Angeles, C. Gosselin, Détermination du degré de liberté des chaînes cinématiques, *Trans. Can. Soc. Mech. Eng.* 12(4) (1988) 219–226. `https://doi.org/10.1139/tcsme-1988-0031`
17. Z. Huang, J. Liu, D. Zeng, A general methodology for mobility analysis of mechanisms based on constraint screw theory, *Sci. China Ser. E-Tech Sci.* 52(5) (2009) 1337–1347.
18. W. Lu, D. Zeng, Z. Huang, Over-constraints and a unified mobility method for general spatial mechanisms part 1: essential principle, *Chin. J. Mech. Eng.* 28(5) (2015) 869–877.
19. W. Lu, D. Zeng, Z. Huang, Over-constraints and a unified mobility method for general spatial mechanisms part 2: application of the principle, *Chin. J. Mech. Eng.* 29(1) (2016) 869–877.