An Analytical and Modular Software Workbench for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots

Conference Paper · August 2019

DOI: 10.1115/DETC2019-97115

2 authors:

Shivesh Kumar
Chalmers University of Technology
70 PUBLICATIONS **475** CITATIONS

SEE PROFILE

Andreas Mueller
Johannes Kepler University Linz
382 PUBLICATIONS **3,113** CITATIONS

SEE PROFILE

Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2019
August 18-21, 2019, Anaheim, California, USA

IDETC2019-97115

DRAFT: AN ANALYTICAL AND MODULAR SOFTWARE WORKBENCH FOR SOLVING KINEMATICS AND DYNAMICS OF SERIES-PARALLEL HYBRID ROBOTS

Shivesh Kumar*
Robotics Innovation Center
German Research Center for Artificial Intelligence
28359 Bremen, Germany
Email: shivesh.kumar@dfki.de

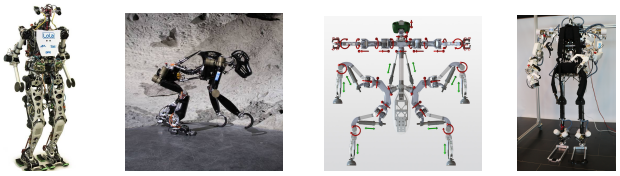
Andreas Mueller
Institute of Robotics
Johannes Kepler University
4040 Linz, Austria
a.mueller@jku.at

ABSTRACT

Parallel mechanisms are increasingly being used as modular subsystem units in various robots and man-machine interfaces for their superior stiffness, payload-to-weight ratio and dynamic properties. This leads to series-parallel hybrid robotic systems which are difficult to model and control due to the presence of various closed loops. Most model based kinematic and dynamic modeling tools resolve loop closure constraints numerically and hence suffer from inefficiency and accuracy issues. Also, they do not exploit the modularity in robot design. In this paper, we present a modular and analytical approach towards kinematic and dynamic modeling of series-parallel hybrid robots. This approach has been implemented in a software framework called Hybrid Robot Dynamics (HyRoDyn) and its application is demonstrated with the help of a series-parallel hybrid humanoid robot recently developed at DFKI-RIC.

1 INTRODUCTION

To combine the various advantages of serial and parallel topologies, hybrid serial-parallel robots have been developed by the industry and academia. For instance, the stiffness of an industrial manipulator can be significantly improved by including a simple parallelogram mechanism. In particular, industrial robots as ABB's IRB4400, IRB6660, KUKA's KR 30,50-PA., KR 700-PA robots, and Comau's Smart H, NJ, NX series, SR400 utilize this design concept [1, 2]. In academics and R&D, the idea to use closed loop mechanisms and parallel kinematics machines (PKMs) has been utilized more liberally, giving rise to a number of biologically inspired lightweight robotic systems with good dynamic characteristics. Fig. 1 shows some of the series-parallel hybrid robots developed in the last decade. Fig. 1a shows the bipedal robot LOLA [3] which is probably the first humanoid robot designed using a modular joint concept utilizing parallel mechanisms. The design of NASA Valkyrie humanoid [4] followed a similar design concept by utilizing parallel mechanisms for its wrist, torso and ankle joints. Fig. 1b shows the hominid robot CHARLIE [5], featuring a stewart platform of type 6-RUS as a six dof active joint in spine and neck. It also utilizes another parallel mechanism in the ankle joint. The multi-legged robot MANTIS [6] depicted in Fig. 1c contains PKMs of type 2-SPU+1U [7] in its ankle joints and closed loop mechanisms of type 1-RRPR that drive certain revolute joints in its legs and torso. Fig. 1d demonstrates a highly modular light weight RECUPERA full-body exoskeleton [8, 9] with 32 active dofs which is built by combining several higher dof joint mod-



(a) Lola (b) Charlie (c) MANTIS (d) Recupera

FIGURE 1: Examples of series-parallel Hybrid robots

*Address all correspondence to this author.

ules: a Stewart platform of type 6-UPS in torso, a double parallelogram [10] in shoulder for flexion–extension movement and ACTIVE ANKLE mechanism (type 3-R[2-SS]) [11] as a 3 dof joint in hip and ankle. The design motivation of such hybrid robots is evident: use of PKM-based submechanisms helps designers to achieve light weight modular design while enhancing the stiffness and dynamic characteristics of the robot. Moreover, a modular design saves manufacturing costs and integration time. These kind of robots combine advantages of both serial and parallel architectures but also inherit their kinematic complexity. Due to this complexity, these robots are typically position controlled [6,5,12] or when torque controlled the full dynamic model is often not exploited [13].

Multi-body kinematics and dynamics has been an area of extensive research during the past decades. While the term kinematics encompasses problems of position, velocity and acceleration analysis, the term dynamics refers to problems associated with the study of forces and torques and their effect on motion of multi-body systems. Notable works include Newton–Euler's [14,15], the Lagrangian [15], and Kane's method [16]. Traditionally, the equations of motion are described in 3D Euclidean space – which quickly yields a large amount of equations for systems of connected bodies [17]. To address this issue, alternative compact and user-friendly formulations have been developed based on screw theory [14, 18] and Lie group theory [19, 20] which can easily be transformed into program code for modern computers. The presence of closed loops in robots significantly increases the complexity of the kinematics and dynamics problems associated with multi-body systems (MBS). Most multi-body simulation tools (e.g. Rigid Body Dynamics Library (`RBDL`) [21] and `OpenSim` [22]) adopt numerical resolution of loop closure constraints for the sake of generality but they are prone to loop closure errors and may suffer from poor performance. Moreover, they usually can not answer various problems related to geometry of these robotic systems. For example, systems with closed loops can have variable mobility, configuration ambiguities (or assembly modes) and can impose redundant constraints on the equations of motion (EOM). Hence, it is interesting for researchers in the domain of robot kinematics, to study the analytical solutions to kinematics problems associated with a specific class or type of parallel mechanisms and their importance over numerical solutions is irrefutable. But this domain specific knowledge is often underrepresented in the design of model based kinematics and dynamics software frameworks.

The main contribution of this paper is to describe the approach behind an analytical and modular software workbench to compute the kinematics and dynamics of arbitrary series-parallel hybrid robots that are serial composition of serial or parallel submechanism modules. It allows modular composition of the analytical loop closure functions and its associated derivatives and transfers them into computation of the analytical forward and inverse dynamics algorithms. The benefit of this approach is that

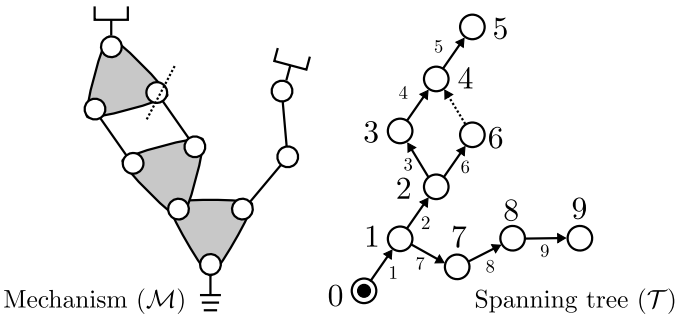


FIGURE 2: A closed loop robot and its associated spanning tree numbered using regular numbering scheme. Here, links and joints are denoted as vertices and edges respectively.

efficient and recursive $O(n)$ dynamics algorithms for tree type system can be directly used. The results are free of numerical errors resulting from loop closure and free of singularities arising from redundant constraints. This approach forms the basis of a software framework called Hybrid Robot Dynamics (HyRoDyn) which has been recently developed at DFKI-RIC. Its application in the kinematic and dynamic modeling of a series-parallel hybrid humanoid leg is demonstrated. A more detailed description of the software framework and its various applications will be reported in a forthcoming paper.

Organization The paper is organized as follows: Section 2 provides theoretical preliminaries for modeling robots with closed loops. It derives the formulas for the forward and inverse dynamics for these systems. Section 3 presents the notion of modularity and the analytical formulations for computing the kinematics and dynamics of series-parallel hybrid robotic systems. Section 4 presents the general idea of HyRoDyn and its application in kinematic and dynamic modeling of RH5 humanoid. Section 5 draws the conclusions and presents future work.

2 MODELING SYSTEMS WITH CLOSED LOOPS

This section briefly introduces the theory of multi-body dynamics subjected to holonomic and sceleronomic constraints [10]. It mostly adopts the notation and terminology introduced by Featherstone in [14]. Therefore, consider a rigid body system with N_B bodies, N_J joints, and $N_L = N_J - N_B$ kinematic loops. Assume that a spanning tree is defined and that the joints are enumerated using regular numbering scheme (see Fig. 2 for example). Let n denote the degree of freedom of the selected spanning tree, computed as $n = \sum_{i=1}^{N_B} n_i$, and let n_c denote the number of loop-closure constraints, computed as $n_c = \sum_{k=N_B+1}^{N_J} n_k^c$. Further, let $\boldsymbol{q}$ indicate the vector of all joints of the spanning tree (of size n) and let $\boldsymbol{y}$ indicate the vector of all independent variables (of size $n - n_c$).

2.1 Loop Constraints

Loop constraints are non-linear constraints on the motion variables of a multi-body system. Loop constraints can be expressed in an implicit and in an explicit way, they are summarized in Table 1 at position, velocity, and acceleration levels. Here, $\boldsymbol{K} = \frac{\partial \phi}{\partial \boldsymbol{q}}$, $\boldsymbol{k} = -\dot{\boldsymbol{K}}\dot{\boldsymbol{q}}$, $\boldsymbol{G} = \frac{\partial \gamma}{\partial \boldsymbol{y}}$, and $\boldsymbol{g} = \dot{\boldsymbol{G}}\dot{\boldsymbol{y}}$. If both functions ϕ and γ describe the same constraint, $\phi \circ \gamma = \boldsymbol{0}$, $\boldsymbol{KG} = \boldsymbol{0}$, and $\boldsymbol{Kg} = \boldsymbol{k}$ can be deduced. Algorithms to compute variables in Table 1 from the spanning tree are provided in [14] and skipped here for brevity.

TABLE 1: Loop constraints [14]

Type	position	velocity	acceleration
implicit:	$\phi(\boldsymbol{q}) = \boldsymbol{0}$	$\boldsymbol{K}\dot{\boldsymbol{q}} = \boldsymbol{0}$	$\boldsymbol{K}\ddot{\boldsymbol{q}} = \boldsymbol{k}$
explicit:	$\boldsymbol{q} = \gamma(\boldsymbol{y})$	$\dot{\boldsymbol{q}} = \boldsymbol{G}\dot{\boldsymbol{y}}$	$\ddot{\boldsymbol{q}} = \boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g}$

2.2 Equations of Motion (EOM)

The equations of motion for a tree topology multi-body system represented by a spanning tree can be written as

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are $(n \times 1)$ vectors of joint position, velocity and acceleration variables of the spanning tree, $\boldsymbol{M}(\boldsymbol{q})$ is the $(n \times n)$ generalized mass-inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a $(n \times 1)$ vector for Coriolis-centrifugal and gravity efforts, and $\boldsymbol{\tau}$ is the $(n \times 1)$ vector of force/torque variables. In case of robots with closed loops, the equivalent spanning tree of the robot system is subjected to loop constraint forces

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_a + \boldsymbol{\tau}_c \tag{2}$$

where $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_a$ are the constraint and active forces, respectively produced by the cut joints. If the selected cut joint is passive, $\boldsymbol{\tau}_a = 0$ can be substituted in Equation 2. The constraint force $\boldsymbol{\tau}_c$ is usually unknown but its value can either be calculated or eliminated from the equation following the Jourdain's principle [23] of virtual power, i.e., $\boldsymbol{\tau}_c \dot{\boldsymbol{q}} = 0$. Based on the (implicit or explicit) nature of the loop constraints, the equations of motion are developed for the entire system.

2.2.1 EOM with implicit loop constraints

The cut joints impose a set of kinematic constraints on the spanning tree which are briefly introduced in Table 1. Assuming that the position level implicit constraints have been successfully differentiated two times, the acceleration level loop constraints can be collected in a single matrix equation of the form

$$\boldsymbol{K}\ddot{\boldsymbol{q}} = \boldsymbol{k} \tag{3}$$

where $\boldsymbol{K}$ is $(n_c \times n)$ matrix. If the system is subjected to an implicit loop constraint then it can be shown that $\boldsymbol{\tau}_c$ takes the form

$$\boldsymbol{\tau}_c = \boldsymbol{K}^T \boldsymbol{\lambda} \tag{4}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers. Combining Equations 2, 3 and 4, the equation of motion of the overall system taking into account implicit loop constraints can be written as

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{K}^T \\ \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \boldsymbol{C} + \boldsymbol{\tau}_a \\ \boldsymbol{k} \end{bmatrix} . \tag{5}$$

This is a system of $(n + n_c)$ equations in $(n + n_c)$ unknowns. The coefficient matrix of dimension $(n + n_c) \times (n + n_c)$ in Equation 5 is symmetric but not positive definite. If the rank, $r = \text{rank}(\boldsymbol{K})$, is less than n_c, then coefficient matrix becomes singular and the system is said to be over-constrained. Over-constrained systems are actually very common. For example, planar kinematic loops impose redundant constraints on the system – that either need to be removed manually [14] or demand numerical decomposition techniques which deteriorate the computational performance and numerical accuracy of the solution [24]. Further, in contrast to a tree type robot, the mobility of closed loop system is dependent on r which can vary with the configuration.

2.2.2 EOM with explicit loop constraints

Using explicit velocity level loop constraint (refer to Table 1) and Jourdain's principle of virtual power, one can establish that $\boldsymbol{\tau}_c$ has the following property

$$\boldsymbol{G}^T \boldsymbol{\tau}_c = 0 . \tag{6}$$

Similarly, one can write the explicit motion constraints at an acceleration level

$$\ddot{\boldsymbol{q}} = \boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g} . \tag{7}$$

Combining Equations 2, 6 and 7, the EOM taking into account explicit loop constraints can be developed.

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{G} \\ \boldsymbol{0} & \boldsymbol{G}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\tau}_c \\ \ddot{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \boldsymbol{C} + \boldsymbol{\tau}_a \\ -\boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} \tag{8}$$

2.3 Forward and Inverse Dynamics

The EOM presented above could be either solved for independent joint accelerations $\ddot{\boldsymbol{y}}$ under given actuator force conditions or for the actuator forces $\boldsymbol{u}$ required to generate given acceleration. The former is called the forward dynamics problem and the latter is called the inverse dynamics problem [14, 18].

2.3.1 Forward dynamics solution By using Equation 6 and multiplying by $\boldsymbol{G}^T$ on both sides of Equation 2, the loop constraint forces $\boldsymbol{\tau}_c$ can be eliminated.

$$\boldsymbol{G}^T\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{G}^T(\boldsymbol{\tau} - \boldsymbol{C}) \tag{9}$$

Substituting Equation 7 in Equation 9 and simplifying one can arrive at the solution to the forward dynamics problem:

$$\ddot{\boldsymbol{y}} = (\boldsymbol{G}^T\boldsymbol{M}\boldsymbol{G})^{-1}(\boldsymbol{G}^T\boldsymbol{\tau} - \boldsymbol{G}^T(\boldsymbol{C} + \boldsymbol{M}\boldsymbol{g}))\,. \tag{10}$$

2.3.2 Inverse dynamics solution Equation 9 could be rewritten as:

$$\boldsymbol{G}^T\boldsymbol{\tau} = \boldsymbol{G}^T(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}) = \boldsymbol{G}^T\boldsymbol{\tau}_{ID} \tag{11}$$

where τ_{ID} is inverse dynamics output of a spanning tree given by $\boldsymbol{\tau}_{ID} = \boldsymbol{M}(\gamma(\boldsymbol{y}))(\boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g}) + \boldsymbol{C}(\gamma(\boldsymbol{y}), \boldsymbol{G}\dot{\boldsymbol{y}})$. The solution to Equation 11 is not unique because $\boldsymbol{G}^T$ is an $(n-r)\times n$ matrix which imposes $(n-r)$ constraints on an n dimensional vector of unknowns, leaving r freedoms of choice. In other words, there are ∞^r different values of $\boldsymbol{\tau}$ which will produce the same acceleration. To arrive at a unique solution, the actuated degrees of freedom must be separated from the passive degrees of freedom. This can be done with the help of a matrix $\boldsymbol{G}_u$ which basically contain the rows of $\boldsymbol{G}$ corresponding to the actuated degrees of freedom. If the rank of matrix $\boldsymbol{G}_u$ is equal to $(n-r)$, then the system is properly actuated[1] and a unique solution to the inverse dynamics problem can be found which is given by:

$$\boldsymbol{\tau}_u = \boldsymbol{G}_u^{-T}\boldsymbol{G}^T\boldsymbol{\tau}_{ID} \tag{12}$$

where $\boldsymbol{\tau}_u$ is a vector of actuator forces required to produce the given acceleration $\ddot{\boldsymbol{y}}$.

3 ANALYTICAL AND MODULAR APPROACH FOR SOLVING KINEMATICS AND DYNAMICS

In this section, we present the notion of modularity and the novel analytical formulations for computing the kinematics and dynamics of series-parallel hybrid robotic systems.

[1] If $p > \text{rank}(\boldsymbol{G}_u)$, the system is redundantly actuated and $\boldsymbol{\tau}_u = \boldsymbol{G}_u^{\dagger T}\boldsymbol{G}^T\boldsymbol{\tau}_{ID}$

TABLE 2: Linkages and PKM modules in series-parallel hybrid robots where EE coordinates can be sensed

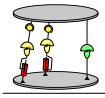
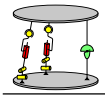
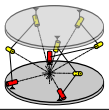
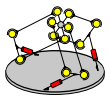
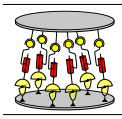
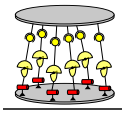
Mechanism		Practical Applications	
Schematic	Type & Mobility (m)	Application	Hybrid Robot
	1-RRRR 1 DOF rotational	Ankle pitch Series-elastic leg Joint 2 and/or Joint 3	Humanoids TORO [25],TALOS [26] ATRIAS [27] ABB IRB4400, KUKA KR 40-PA etc.
	1-RRPR 1 DOF rotational	Hip, Torso Hip flexion-extension, Knee Inner and Outer leg joints	MANTIS [6] RH5 [28], HADE leg [29] SHERPATT rover [12]
	2-SPU+1U 2 DOF universal	Wrist, Torso Ankle Ankle Hip Roll-Yaw, Ankle	RH5 [28] MANTIS [6] LOLA humanoid [3] SAFFIR & THOR [13]
	2-PUS+1U 2 DOF universal	Wrist, Ankle and Torso	VALKYRIE humanoid [4]
	2-SPRR+1U 2 DOF universal	Ankle	RH5 [30]

TABLE 3: PKM modules in series-parallel hybrid robots where EE coordinates can not be sensed

Mechanism		Practical Applications	
Schematic	Type & Mobility (m)	Application	Hybrid Robot
	3-RRR 3 DOF spherical	Hip, Ankle	SPHERICAL EXO SUIT [31]
	3-R[2-SS] 3 DOF almost spherical	Hip, Ankle	RECUPERA-REHA full-body exo [32]
	6-UPS 6 DOF free	Torso	RECUPERA-REHA full-body exo [8]
	6-RUS 6 DOF free	Torso, Neck	CHARLIE homonid [5]

3.1 Notion of modularity

3.1.1 Motivation Hybrid robots are robots that are composed of a series of serial or parallel submechanisms. Table 2 and Table 3 presents a brief survey on closed loop kinematics and parallel submechanism modules which have been utilized in series-parallel hybrid robot designs in the last decade. Red, green and yellow colors denote active, EE and other passive joints respectively. In all cases, these submechanisms are used as a mechanical generators of m-dimensional motion subspaces of $SE(3)$ i.e. they are used as an abstraction to either

an active lower pair joint (for e.g. revolute joint, spherical joint etc.) or a universal joint which is quite often used in the design of robotic systems. Such a design methodology is biologically inspired as most joints found in such systems are actuated with muscle groups in a parallel architecture. This allows the exploitation of the non-linear transmission from the actuation space to the task space and provide better actuator placement possibilities which can optimise the mass-inertia properties of the limbs. Table 2 shows the list of linkages and parallel submechanisms where the platform coordinates (highlighted with green) are a subset of generalized coordinates of the associated spanning tree. Since, it is well known that it is difficult to solve the forward kinematics of the parallel mechanisms in real time, there is a tendency to choose parallel mechanisms where additional sensors can be integrated to measure the platform coordinates. Table 3 shows the list of parallel submechanisms where the platform coordinates are not a subset of generalized coordinates of the associated spanning tree. Here, it is also not possible to put extra sensors to measure the platform coordinates directly but in some cases they may be integrated in other passive joints to simplify the calculation of platform coordinates from actuator states. The use of such parallel submechanisms is less common in the design of series-parallel hybrid robots in comparison to the ones presented in Table 2.

Two observations can be made from this survey: submechanism modules are used as a mechanical generator of a motion subspace (revolute, universal, spherical, free joint etc.) and the same type of submechanism with different physical parameters is utilized as a module to serve different purposes (ankle, wrist, torso joints etc.) in the same robot. The analysis of closed loop mechanisms is difficult and hence they require special treatments in contrast to tree type systems. Numerical treatment of the loop closure constraints can lead to inaccuracy and poor performance as described in the previous section. This inspires an analytical treatment of loop closure constraints for a submechanism module in a robot assembly so that the modularity is reflected in both hardware and software domains.

3.1.2 Definition of submechanism module

A submechanism module ($\mathcal{M}_i$) is defined as a set of links and joints which can produce any motion from m-dimensional motion subspace of $SE(3)$ while demonstrating properties of a *minimal loop cluster*. A loop cluster is any set of loops with the property that no loop within the cluster is coupled to any other loop outside it; and a minimal loop cluster is the one that can not be divided into any further loop clusters. A serial combination of links and joints can be seen as a submechanism with no closed loops. This definition helps us in two different ways:

1. it reveals the block diagonal structure in the constraint Jacobian matrix ($\boldsymbol{G}$) and its derivative ($\dot{\boldsymbol{G}}$) which can lead to efficiency in kinematics and dynamics computations,

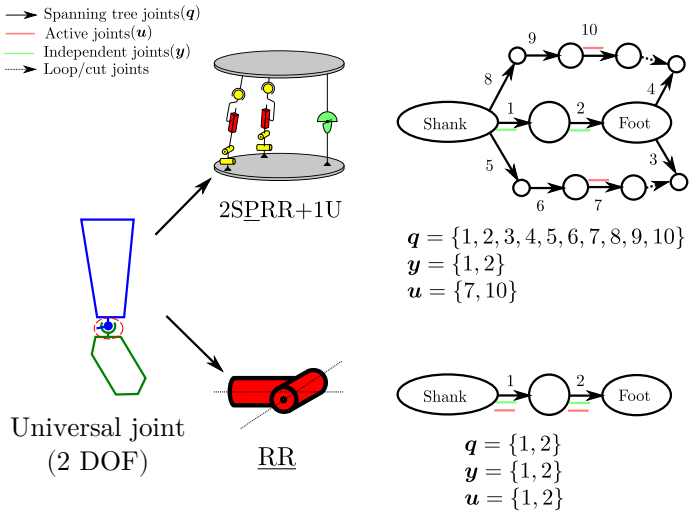


FIGURE 3: Example of submechanism definition

2. it helps in generation of simpler abstract mechanisms which do not further contain any closed loops and have physical meaning.

To define a submechanism module, it is crucial to define three subsets of joints from the connectivity graph ($\mathcal{G}_i$) of a mechanism:

1. spanning tree joints ($\boldsymbol{q}_i \in \mathbb{R}^{n_i}$): all the joints belonging to the spanning tree ($\mathcal{T}_i$) chosen by regular numbering scheme,
2. independent joints ($\boldsymbol{y}_i \in \mathbb{R}^{m_i}$): the set of platform coordinates $\boldsymbol{y}_i$ such that they define $\boldsymbol{q}_i$ uniquely,
3. active joints ($\boldsymbol{u}_i \in \mathbb{R}^{p_i}$): all the joints containing the actuators

Let us define a selection matrix, $\boldsymbol{Q}_i \in \mathbb{R}^{p_i \times n_i}$ such that $\boldsymbol{u}_i = \boldsymbol{Q}_i \boldsymbol{q}_i$. Also, in the scope of the current work, it is assumed the submechanism modules are *properly actuated* (i.e. $p_i = m_i$) and *properly constrained* (i.e. $m_i = n_i - n_{ci}$).

3.1.3 Example

The designer may construct the ankle joint of a humanoid robot either using a set of two serially connected actuators (e.g. 2R orthogonal arrangement) for the sake of simplicity or utilize a parallel mechanism (e.g. 2SPRR+1U [30]) for minimizing the lower leg inertia and exploiting the non-linear transmission. Fig. 3 shows the two design possibilities and the corresponding definitions of the submechanism modules.

3.2 Topological modeling

A submechanism module is modeled using a two-terminal graph (TTG) which is defined as a graph with two distinguished vertices, called source and sink. The source and sink basically represents fixed transformations to submechanism interface points (for e.g. physical screws or nut-bolt pair) in the overall

assembly of the hybrid robot. Further, a spanning tree is deduced from this TTG, where all the nodes except for the ones connected using fixed joint are numbered using regular numbering scheme. Fig. 3 shows the associated submechanism graphs of the humanoid ankle skipping the source and sink links.

The series composition $\mathcal{G} = \mathcal{G}_x \cup \mathcal{G}_y$ of two TTGs $\mathcal{G}_x$ and $\mathcal{G}_y$ is a TTG created from the disjoint union of graphs $\mathcal{G}_x$ and $\mathcal{G}_y$ by merging the sink of $\mathcal{G}_x$ with the source of $\mathcal{G}_y$. The source of $\mathcal{G}_x$ becomes the source of $\mathcal{G}$ and the sink of $\mathcal{G}_y$ becomes the sink of $\mathcal{G}$ [33]. A series-parallel hybrid robot can be seen as series composition of various submechanism modules represented by their respective graphs. Let $\mathcal{T}_i$ denote the spanning tree of a submechanism module and s serial or parallel submechanism modules are joined in series to compose a series-parallel hybrid robot. The spanning tree of this hybrid robot ($\mathcal{T}$) will be given by:

$$\mathcal{T} = \bigcup_{i=1}^{s} \mathcal{T}_i \tag{13}$$

The three joint variable sets namely spanning tree joints, independent joints and active joints set for the hybrid robot can be composed as: $\boldsymbol{q}^T = (\boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_s^T)$, $\boldsymbol{y}^T = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_s^T)$, $\boldsymbol{u}^T = (\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_s^T)$. Fig. 4 shows the composition of a series-parallel hybrid humanoid leg with the help of four submechanism modules.

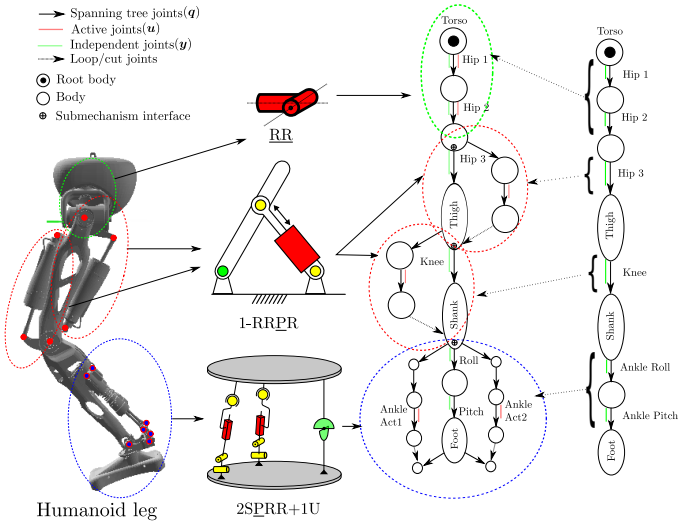


FIGURE 4: Example of series-parallel hybrid composition

3.3 Kinematic & Dynamic modeling

To model the kinematics of submechanism module, let us define loop closure functions (LCF) [14] such that they provide a unique mapping between the independent position variables $\boldsymbol{y}_i$ and position variables in the spanning tree $\boldsymbol{q}_i$. The set of independent joint variables $\boldsymbol{y}_i$ may or may not be a subset of the position variables of the spanning tree $\boldsymbol{q}$. For this assumption to hold true, let us define a set $C \subseteq \mathbb{R}^{n_i - r}$ of admissible values of $\boldsymbol{y}_i$, and assume $\boldsymbol{y}_i \in C$. For all $\boldsymbol{y}_i \in C$, there exists a function, γ_i such that

$$\begin{aligned} \boldsymbol{q}_i &= \gamma_i(\boldsymbol{y}_i) \\ \dot{\boldsymbol{q}}_i &= \boldsymbol{G}_i \dot{\boldsymbol{y}}_i \\ \ddot{\boldsymbol{q}}_i = \boldsymbol{G}_i \ddot{\boldsymbol{y}}_i + \dot{\boldsymbol{G}}_i \dot{\boldsymbol{y}}_i &= \boldsymbol{G}_i \ddot{\boldsymbol{y}}_i + \boldsymbol{g}_i \end{aligned} \tag{14}$$

For serial submechanism modules, the independent variables in the spanning tree are the same as generalized coordinates of the spanning tree i.e. $\boldsymbol{q}_i = \boldsymbol{y}_i$, $\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{y}}_i$ and $\ddot{\boldsymbol{q}}_i = \ddot{\boldsymbol{y}}_i$. Problems related to geometry or kinematics for parallel robots is usually easy to formulate but difficult to solve because they result in a set of non-linear algebraic equations which need careful analysis and treatment. The three most useful solution techniques to deal with such problems are polynomial continuation, Gröbner bases, and elimination method [34]. However, the geometer or kinematician may choose any formulation and solution method which works the best for a specific type of parallel robot and arrive at the loop closure function. The foundations for deriving LCFs for 1-RRPR and 2-SPRR+1U mechanisms used in the humanoid leg example introduced in Fig. 4 can be found in [14] and [30] respectively and are skipped here for brevity.

The loop closure function for the series-parallel hybrid robot is composed as:

$$\begin{aligned} \boldsymbol{\gamma} &= \begin{bmatrix} \boldsymbol{\gamma}_1^T & \boldsymbol{\gamma}_2^T & \cdots & \boldsymbol{\gamma}_s^T \end{bmatrix}^T_{(n\times 1)} \\ \boldsymbol{G} &= \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}_2 & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \ldots & \ldots & \boldsymbol{G}_s \end{bmatrix}_{(n\times m)} \\ \boldsymbol{g} &= \begin{bmatrix} \boldsymbol{g}_1^T & \boldsymbol{g}_2^T & \cdots & \boldsymbol{g}_s^T \end{bmatrix}^T_{(n\times 1)} . \end{aligned} \tag{15}$$

From Equation 15, three observations about $\boldsymbol{G}$ can be made: it typically contains many zeros due to branch induced sparsity, it may contain various identity matrix blocks corresponding to serial submechanism modules and it has a block diagonal nature due to modular representation of the spanning tree. These properties of the matrix can be used to save some computational costs occuring in sparse matrix multiplications. Fig. 5 shows the block diagonal nature of the loop closure Jacobian for the series-parallel hybrid leg example introduced in Fig. 4. The resulting actuator Jacobian computed as $\boldsymbol{G}_u = \boldsymbol{Q}\boldsymbol{G}$ is also a block diagonal matrix. Alg. 1 presents an algorithm to compute the position,

velocity and acceleration state of the spanning tree from the loop closure function exploiting these properties.

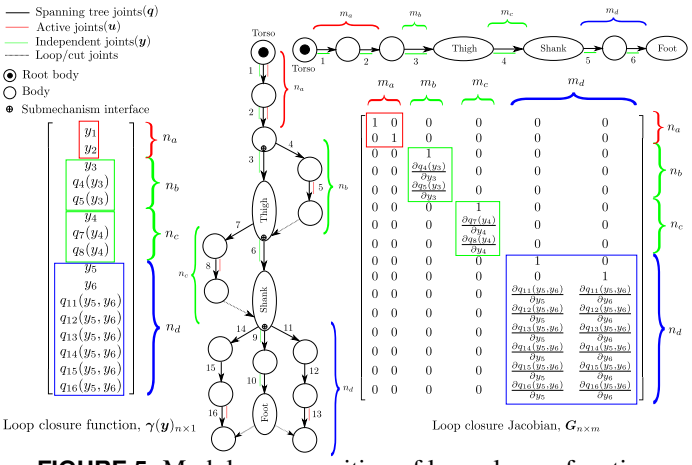


FIGURE 5: Modular composition of loop closure function

Algorithm 1 Spanning tree state (SysState)

(in) Independent joint state $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$
(out) Spanning tree joint state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$

1: **function** SysState($\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$)
2: **for** $\mathcal{T}_i \in (\mathcal{T}_1, \ldots, \mathcal{T}_s)$ **do**
3: **if** $m_i \neq n_i$ **then** ▷ Check if module is parallel
4: $\boldsymbol{q}_i = \boldsymbol{\gamma}_i(\boldsymbol{y}_i)$
5: $\dot{\boldsymbol{q}}_i = \boldsymbol{G}_i \dot{\boldsymbol{y}}_i$
6: $\ddot{\boldsymbol{q}}_i = \boldsymbol{G}_i \ddot{\boldsymbol{y}}_i + \boldsymbol{g}_i$
7: **else**
8: $\boldsymbol{q}_i = \boldsymbol{y}_i, \dot{\boldsymbol{q}}_i = \dot{\boldsymbol{y}}_i, \ddot{\boldsymbol{q}}_i = \ddot{\boldsymbol{y}}_i$
9: $\boldsymbol{q} \leftarrow \boldsymbol{y}_i, \dot{\boldsymbol{q}} \leftarrow \dot{\boldsymbol{y}}_i, \ddot{\boldsymbol{q}} \leftarrow \ddot{\boldsymbol{y}}_i \; \forall i \in [1, \ldots, s]$
10: **return** $[\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}]$

With $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ known by solving the loop constraints determined by Equation 15, only the equations (1) of the unconstrained system need to be evaluated. This is most efficiently carried out by a recursive $O(n)$ algorithm. Various of such have been proposed in the literature. The actual compuational effort depends on the representation of spatial twists and wrenches. The spatial representation [18,14] is deemed as the most efficient representation. For the i^{th} submechanism module, denote with $\boldsymbol{V}_{i,k} = \left(\boldsymbol{\omega}^T, \boldsymbol{v}^T\right)^T \in se(3)$ the twist vector of body k, and with $\boldsymbol{J}_{i,k}$ the instantaneous screw coordinate vector of joint k, both in spatial representation. The recursive Newton-Euler algorithm (RNEA) [14,19,20] consists of a forward recursion (F.R.), where the spatial state of the system is computed from the generalized coordinates, velocities, and accelerations $(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i)$. The backward recursion (B.R.) in RNEA computes the generalized forces $\boldsymbol{\tau}_i$ from the spatial state of the system. A modular form of RNEA is presented in Alg. 2 where the forward and backward recursions are carried out first within the module (intra-modular) and then across the modules (inter-modular). Here, $\boldsymbol{V}_{i,k}$, $\boldsymbol{M}_{i,k}$, $\boldsymbol{W}_{i,k}^{\text{app}}$ are the spatial twist, the inertia matrix, and the applied wrench of body k in submodule i, respectively. The 6×6 matrix

$$\mathbf{ad}_{\boldsymbol{V}} = \begin{pmatrix} \widetilde{\boldsymbol{\omega}} & \boldsymbol{0} \\ \widetilde{\boldsymbol{v}} & \widetilde{\boldsymbol{\omega}} \end{pmatrix} \tag{16}$$

is called the screw product or spatial cross product operator. Here, $\widetilde{\boldsymbol{\omega}}$ and $\widetilde{\boldsymbol{v}}$ represent the skew symmetric matrices associated with the vectors $\boldsymbol{\omega}$ and $\boldsymbol{v}$ respectively.

Algorithm 2 Modular Recursive Newton Euler Algorithm (RNEA)

(in) Spanning tree joint state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, vector of external wrenches $\boldsymbol{W}^{\text{ext}}$ and system model $(\mathcal{T})$
(out) Tree joint forces $(\boldsymbol{\tau})$

1: **function** RNEA($\mathcal{T}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$)
2: $\boldsymbol{V}_{0,0} = \boldsymbol{0}$ ▷ Velocity of root link
3: $\dot{\boldsymbol{V}}_{0,0} = (\boldsymbol{0}, -\boldsymbol{g})^T$ ▷ $\boldsymbol{g}$ – gravity vector
4: **for** $i \in (1, \ldots, s)$ **do** ▷ Inter-modular F.R.
5: **for** $k \in (1, \ldots, n_i)$ **do** ▷ Intra-modular F.R.
6: $\boldsymbol{V}_{i,k} = \boldsymbol{V}_{i,k-1} + \boldsymbol{J}_{i,k}\dot{q}_{i,k}$
7: $\dot{\boldsymbol{V}}_{i,k} = \dot{\boldsymbol{V}}_{i,k-1} + \boldsymbol{J}_{i,k}\ddot{q}_{i,k} + \mathbf{ad}_{\boldsymbol{V}_{i,k-1}}\boldsymbol{V}_{i,k}$
8: **for** $i \in (s, \ldots, 1)$ **do** ▷ Inter-modular B.R.
9: **for** $k \in (n_i - 1, \ldots, 1)$ **do** ▷ Intra-modular B.R.
10: $\boldsymbol{W}_{i,k} = \boldsymbol{W}_{i,k+1} + \boldsymbol{M}_{i,k}\dot{\boldsymbol{V}}_{i,k} - \mathbf{ad}^T_{\boldsymbol{V}_{i,k}}\boldsymbol{M}_{i,k}\boldsymbol{V}_{i,k} + \boldsymbol{W}^{\text{ext}}_{i,k}$
11: $\tau_{i,k} = \boldsymbol{J}^T_{i,k}\boldsymbol{W}_{i,k}$
12: **return** $\boldsymbol{\tau}$

Lastly, an algorithm to solve the inverse dynamics of the series-parallel hybrid composition is provided in Alg. 3 which takes as input the motion described in independent coordinates $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$. First, the full state of the spanning tree is computed (Line 2) with the help of Alg. 1. The next step is to compute the joint forces $(\boldsymbol{\tau})$ for the spanning tree which is done using a modular form of RNEA presented in Alg. 2. And lastly, the tree joint forces are converted to the actuator forces of the series-parallel hybrid robot using an inter-modular recursion (see Line 6 and Line 8) by exploiting the block diagonal nature of loop closure Jacobian matrix.

Computational effort In general, it is more difficult to analyse the computational performance of dynamics algorithms for sys-

tems with closed loops because the closed form solutions to the loop closure constraints might not always exist and the number of floating point operations involved are highly dependent on the geometry of the parallel mechanism. Nevertheless, the presented modular approach leads to some cost savings occuring in matrix-vector multiplication and inversion which will be discussed here. The tree joint forces ($\boldsymbol{\tau}$) are solved using $O(n)$ RNEA. The projection of these forces to the independent joint space of the robot i.e. $\boldsymbol{\tau}_y = \boldsymbol{G}^T \boldsymbol{\tau}$ can be computed with mn multiplications and $m(n-1)$ additions or $2m(n-1)$ floating point operations (FLOPs). Due to block diagonal structure in $\boldsymbol{G}$, this matrix-vector multiplication can be done in $2\sum_{i=1}^{s} m_i(n_i - 1)$ FLOPs. It is trivial to show that $2\sum_{i=1}^{s} m_i(n_i - 1) \leq 2m(n-1)$ which demonstrates the cost savings involved in this step [2]. The projection of independent joint forces to actuator forces requires the inversion of actuator jacobian matrix $\boldsymbol{G}_u$ which can be done with $O(m^3)$ complexity and its multiplication with $\boldsymbol{\tau}_y$ vector which requires $2m(m-1)$ FLOPs. Due to the modular formulation, actuator Jacobian matrix $\boldsymbol{G}_u$ also has a block diagonal structure and instead of a full inversion, its inverse can be computed simply by computing the inverse of its submatrix blocks along the diagonal. Hence, compared to $O((\sum_{i=1}^{s} m_i)^3)$ complexity involved in this process, the block diagonal nature leads to a reduced inversion complexity of $O(\sum_{i=1}^{s} m_i^3)$ as $\sum_{i=1}^{s} m_i^3 \leq (\sum_{i=1}^{s} m_i)^3 \forall m_i \in \mathbb{N}$. Lastly, the matrix-vector multiplication $\boldsymbol{\tau}_u = \boldsymbol{G}_u^{-1} \boldsymbol{\tau}_y$ leads to a reduced cost of $2\sum_{i=1}^{s} m_i(m_i - 1)$ FLOPs as compared to $2m(m-1)$ FLOPs.

Algorithm 3 Inverse Dynamics (IDYN)

(in) Independent joint state $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ and system model $(\mathcal{T})$
(out) Actuator forces $(\boldsymbol{\tau}_u)$

function IDYN($\mathcal{T}, \boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$)
  $[\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}] \leftarrow$ SYSSTATE($\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$) ▷ Alg. 1
  $\boldsymbol{\tau} \leftarrow$ RNEA($\mathcal{T}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$) ▷ Inv. dyn. of tree, Alg. 2
  for $\mathcal{T}_i \in (\mathcal{T}_1, \ldots, \mathcal{T}_s)$ **do**
    if $m_i \neq n_i$ **then** ▷ Check if module is parallel
      $\boldsymbol{\tau}_{ui} \leftarrow \boldsymbol{G}_{ui}^{-1} \boldsymbol{G}_i^T \boldsymbol{\tau}_i$
    else
      $\boldsymbol{\tau}_{ui} \leftarrow \boldsymbol{\tau}_i$
    $\boldsymbol{\tau}_u \leftarrow (\boldsymbol{\tau}_{ui} : 1 \leq i \leq s)$
  return $\boldsymbol{\tau}_u$

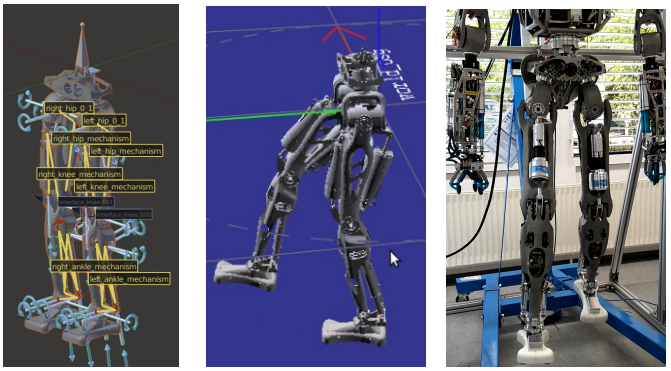
4 HYRODYN SOFTWARE AND ITS APPLICATIONS

This section presents briefly the idea of HyRoDyn software framework and presents its application in modeling series-parallel hybrid robots.

[2] $\sum_{i=1}^{s} m_i n_i \leq \sum_{i=1}^{s} m_i \sum_{i=1}^{s} n_i \ \forall \ m_i, \ n_i \in \mathbb{N}$

4.1 HyRoDyn software framework

The main idea behind HyRoDyn is to store the analytically derived LCF in a *configurable* mechanism library which is identified by its type. LCF of several novel parallel mechanisms such as 2SPRR+1U [30], 3[R-[2-SS]] [32] etc. have been derived after a rigorous kinematic analysis. Based on submechanisms defined in a hybrid robot, HyRoDyn can modularly compose the LCF of the overall system in an automated way. HyRoDyn is implemented in C++ and utilizes $O(n)$ multi-body dynamics algorithms for tree type systems from RBDL [21](based on [14]). The input to HyRoDyn is SMURF[3] file which can be generated using a Blender based visual editor called Phobos[4] (see Fig. 6a). HyRoDyn can be used both for simulation (see Fig. 6b) and real time control (see Fig. 6c) of the complex hybrid robots. For example, inverse dynamics of RH5 humanoid leg example as presented before (with 18 DOF in spanning tree, 6 active DOF, 4 independent closed loops) can be solved in the order of a few µs. Presently, closed form solutions to mechanisms such as 1-RRPR, 2-SPU+1U, 2-SPRR+1U, 6-RUS, 6-UPS, parallelogram chains are available in its submechanism libraries and HyRoDyn can be used to analytically solve the kinematics and dynamics of arbitrary series-parallel hybrid robots composed of these submechanism modules. Actuation of the robot can be arbitrarily selected. HyRoDyn has been used in the analysis and control of robots such as Recupera exoskeleton [9] and RH5 humanoid.



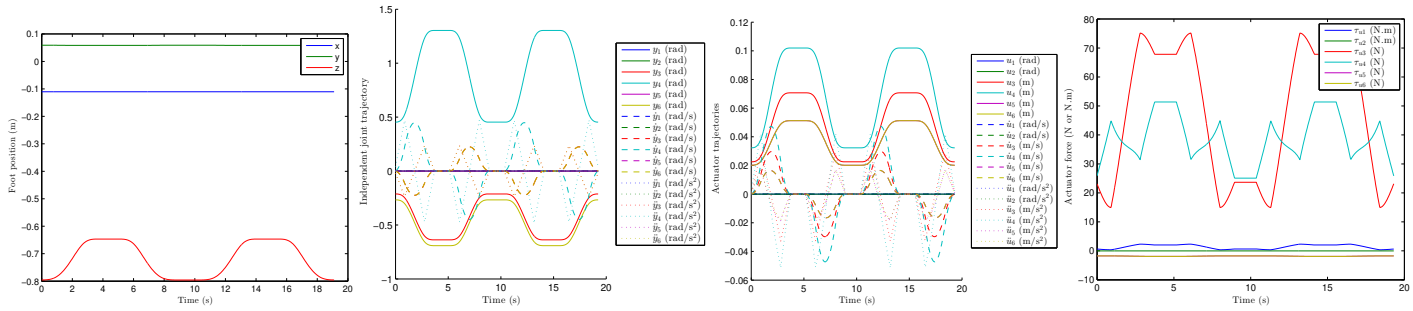
(a) Modeling (b) Simulation (c) Control

FIGURE 6: HyRoDyn: Modeling, Simulation and Control

4.2 Application on RH5 humanoid

We present the task space control of RH5 humanoid leg previously introduced in Fig. 4 using HyRoDyn. A perfect model is assumed and hence, no controller action is required. Two set points i.e. foot up position & foot down position, separated by 15 cm in z-direction, are chosen in the task space of the robot. Using inverse kinematics, the independent joint positions needed

[3] https://github.com/rock-simulation/smurf_parser
[4] https://github.com/dfki-ric/phobos



(a) Foot position (b) Independent joint trajectories (c) Actuator trajectories (d) Actuator forces

FIGURE 7: Task space and independent joint space trajectories of RH5 leg for up-down movement

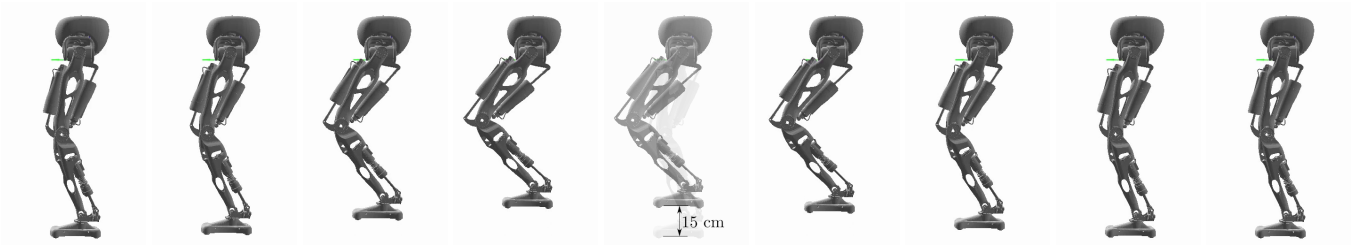


FIGURE 8: Animation of up-down movement with RH5 leg

to reach these points are computed for the abstracted serial robot. This can be done either numerically or analytically. Then, these waypoints in independent joint space are fed to an interpolator which provides smooth trajectories $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ for up and down movement of the leg. Fig. 7a and Fig. 7b shows the task space and independent joint space trajectories for the RH5 leg. These trajectories are then used to compute the actuator trajectories $(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}, \boldsymbol{\tau}_u)$ using inverse kinematics and inverse dynamics algorithms as presented earlier. Fig. 7c and Fig. 7d shows the position, velocity, acceleration and torque required in different actuators of the robot to produce this movement. Further, full spanning tree state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is computed during this process which can be used for robot visualization (see Fig. 8).

5 CONCLUSION

This paper presents an analytical and modular software workbench, called HyRoDyn, for solving kinematics and dynamics of series-parallel hybrid robotic systems which are becoming increasingly popular in the recent years. They inherit advantages and kinematic complexity of both architectures and hence require careful treatment in their analysis and control. Instead of resorting to numerical resolution of loop closure constraints, this software workbench provides a holistic approach towards dealing with the complexity of these systems using modular formulations which helps in exploiting the hierarchy in robot design. An application of this software in the task space control of RH5 humanoid leg is presented. It is planned to make this software open-source in future so that more people from the kinematics community can help in contributing to the submechanism libraries in HyRoDyn so that catalogue of supported parallel mechanisms can be enriched. This software tools serves as an efficient and error-free alternative to the already existing generic numerical multi-body kinematics and dynamics software.

ACKNOWLEDGMENT

This work was performed performed within the projects D-RoCK and Q-RoCK, funded by the German Aerospace Center (DLR) with federal funds from the Federal Ministry of Education and Research (BMBF) (Grant Numbers: 01IW15001 and FKZ 01IW18003 respectively). The second author acknowledges the support from the LCM K2 Center for Symbiotic Mechatronics within the framework of the Austrian COMET-K2 program.

REFERENCES

[1] To, M., and Webb, P., 2012. "An improved kinematic model for calibration of serial robots having closed-chain mechanisms". *Robotica,* ***30***(6), Oct., pp. 963–971.

[2] Gautier, M., Khalil, W., and Restrepo, P. P., 1995. "Identification of the dynamic parameters of a closed loop robot". In Proceedings of 1995 IEEE International Conference on Robotics and Automation, Vol. 3, pp. 3045–3050 vol.3.

[3] Lohmeier, S., and et. al., 2006. "Modular joint design for performance enhanced humanoid robot lola". In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pp. 88–93.

[4] Radford, N. A., and et. al., 2015. "Valkyrie: Nasa's first bipedal humanoid robot". *Journal of Field Robotics,* ***32***(3), pp. 397–419.

[5] Kuehn, D., Bernhard, F., Burchardt, A., Schilling, M., Stark, T., Zenzes, M., and Kirchner, F., 2014. "Distributed computation in a quadrupedal robotic system". *International Journal of Advanced Robotic Systems,* ***11***(7), p. 110.

[6] Bartsch, S., and et. al., 2016. "Development and control of the multi-legged robot mantis". In Proceedings of ISR 2016: 47st International Symposium on Robotics, pp. 1–8.

[7] Serracn, J., Puglisi, L., Saltaren, R., Ejarque, G., Sabater-Navarro, J., and Aracil, R., 2012. "Kinematic analysis of a novel 2-d.o.f. orientation device". *Robotics and Autonomous Systems,* ***60***(6), pp. 852 – 861.

[8] Kirchner, E. A., and et. al., 2016. "Recupera-reha: Exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation". In Transdisziplinaere Konferenz SmartASSIST, pp. 504–517.

[9] Kumar, S., and et. al., 2019. "Modular design and decentralized control of the recupera exoskeleton for stroke rehabilitation". *Applied Sciences,* ***9***(4).

[10] Kumar, S., Simnofske, M., Bongardt, B., Mueller, A., and Kirchner, F., 2017. "Integrating mimic joints into dynamics algorithms exemplified by the hybrid recupera exoskeleton". In Advances In Robotics (AIR-2017), June 28 - July 2, New Delhi, India, ACM-ICPS.

[11] Simnofske, M., Kumar, S., Bongardt, B., and Kirchner, F., 2016. "Active ankle - an almost-spherical parallel mechanism". In 47th International Symposium on Robotics (ISR).

[12] Cordes, F., Babu, A., and Kirchner, F., 2017. "Static force distribution and orientation control for a rover with an actively articulated suspension system". In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5219–5224.

[13] Hopkins, M. A., Ressler, S. A., Lahr, D. F., Leonessa, A., and Hong, D. W., 2015. "Embedded joint-space control of a series elastic humanoid". In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3358–3365.

[14] Featherstone, R., 2008. *Rigid Body Dynamics Algorithm*.

[15] Khalil, W., and Dombre, E., 2002. *Modeling, Identification and Control of Robots*, 3rd ed. Taylor & Francis, Inc., Bristol, PA, USA.

[16] Buffinton, K. W., 2005. *Robotics and Automation Handbook*. CRC Press, ch. Kane's Method in Robotics.

[17] Luh JS, Walker MW, P. R., 1980. "On-line computational scheme for mechanical manipulators". *ASME. J. Dyn. Sys., Meas., Control.*

[18] Jain, A., 2011. *Robot and Multibody Dynamics: Analysis and Algorithms*. Springer Verlag.

[19] Müller, A., 2017. "Screw and lie group theory in multibody kinematics". *Multibody System Dynamics*, Jul.

[20] Müller, A., 2018. "Screw and lie group theory in multibody dynamics". *Multibody System Dynamics,* ***42***(2), Feb, pp. 219–248.

[21] Felis, M. L., 2017. "RBDL: an efficient rigid-body dynamics library using recursive algorithms". *Autonomous Robots,* ***41***(2), pp. 495–511.

[22] Delp, S. L., and et. al., 2007. "OpenSim: Open-source software to create and analyze dynamic simulations of movement". *IEEE Transactions on Biomedical Engineering,* ***54***(11), Nov, pp. 1940–1950.

[23] Piedboeuf, J. C., 1993. "Kane's equations or Jourdain's principle?". In Proceedings of 36th Midwest Symposium on Circuits and Systems, pp. 1471–1474 vol.2.

[24] Mueller, A., 2014. "Implementation of a geometric constraint regularization for multibody system models". *Arch. Mech. Eng.*

[25] Englsberger, J., and et. al., 2014. "Overview of the torque-controlled humanoid robot TORO". In 2014 IEEE International Conference on Humanoid Robots, pp. 916–923.

[26] Stasse, O., and et. al., 2017. "TALOS: A new humanoid research platform targeted for industrial applications". In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pp. 689–695.

[27] Hubicki, C., and et. al., 2016. "ATRIAS: Design and validation of a tether-free 3d-capable spring-mass bipedal robot". *The International Journal of Robotics Research,* ***35***(12), pp. 1497–1521.

[28] Peters, H., Kampmann, P., and Simnofske, M., 2017. "Konstruktion eines zweibeinigen humanoiden roboters". In 2. VDI Fachkonferenz Humanoide Roboter.

[29] Garcia, E., Arevalo, J. C., Muoz, G., and Gonzalez-de Santos, P., 2011. "On the biomimetic design of agile-robot legs". *Sensors,* ***11***(12), pp. 11305–11334.

[30] Kumar, S., Nayak, A., Peters, H., Schulz, C., Mueller, A., and Kirchner, F., 2018. "Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot". In Advances in Robot Kinematics, M. Carricato, ed., Springer Verlag GmbH.

[31] Sadeqi, S., Bourgeois, S. P., Park, E. J., and Arzanpour, S., 2017. "Design and performance analysis of a 3-rrr spherical parallel manipulator for hip exoskeleton applications". *Journal of Rehabilitation and Assistive Technologies Engineering,* ***4***, p. 2055668317697596.

[32] Kumar, S., Bongardt, B., Simnofske, M., and Kirchner, F., 2018. "Design and Kinematic Analysis of the Novel Almost Spherical Parallel Mechanism Active Ankle". *Journal of Intelligent & Robotic Systems*, Mar.

[33] Eppstein, D., 1992. "Parallel recognition of series-parallel graphs". *Information and Computation,* ***98***(1), pp. 41 – 55.

[34] Nielsen, J., and Roth, B., 1999. "On the kinematic analysis of robotic mechanisms". *The International Journal of Robotics Research,* ***18***(12), pp. 1147–1160.