Modular and Hybrid Numerical-Analytical Approach - A Case Study on Improving Computational Efficiency for Series-Parallel Hybrid Robots

Rohit Kumar[1], Shivesh Kumar[1], Andreas Müller[2], and Frank Kirchner[1,3]

***Abstract*— Modeling closed loop mechanisms is a necessity for the control and simulation of various systems and poses a great challenge to rigid body dynamics algorithms. Solving the forward and inverse dynamics for such systems require resolution of loop closure constraints which are often solved via numerical procedures. This brings an additional burden to these algorithms as they have to stabilize and control the loop closure errors. In order to avoid this issue, analytical solutions are preferred for commonly studied parallel mechanisms. This paper has two contributions. Firstly, it reports a case study on a modular and hybrid numerical-analytical approach to model and control series-parallel hybrid robots which are subjected to large number of holonomic constraints. The approach exploits the modularity in the robot design to combine analytical loop closure for the known submechanisms and numerical loop closure for submechanisms where analytical solutions are not available. This offers an edge over purely numerical approach in terms of computational efficiency. Secondly, an adaption of the constraint embedding approach in Articulated Body Algorithm (ABA) is presented which yields a recursive algorithm in minimal coordinates for computing the forward dynamics of series-parallel hybrid systems. The proposed modification exploits the Lie group formulations and allows easy implementation of recursive forward dynamics of constrained systems in state of the art multi-body solvers.**

I. Introduction

Series-parallel hybrid robots can be defined as a combination of series or tree-type chains and parallel mechanisms. The serial and parallel robots are often combined to exploit advantages of both the topologies. For instance, serial robots provide larger workspace and are easier to control, and parallel robots provide high stiffness, high precision, and high payload capacities. An extensive survey on series-parallel hybrid robots is available in literature [1]. While there are many advantages of having such hybrid robots, the kinematic complexities are inherited from the serial and parallel robots. The number of holonomic constraints that the system is subjected to increases with number and complexity of parallel kinematic manipulator (PKMs). Hence, it becomes very important to resolve the loop closures in such hybrid robots in a computationally efficient and error free manner.

There are many commercial software (e.g. ADAMS, RecurDyn, Simmechanics, V-Rep, etc.) and some open source rigid body dynamics libraries (e.g. RBDL [2], OpenSim [3]) which provides support for modeling closed loop robots. Solving equations of motion for rigid body system with closed loops require resolution of loop closure constraints which are often solved via numerical procedures. This brings an additional burden to these algorithms as they have to stabilize and control the loop closure errors. Also, the numerical solvers may lead to inaccuracies, and the errors increase as the number of loop closures increase in the system. To circumvent this issue, a modular and analytical approach was implemented in Hybrid Robot Dynamics (HyRoDyn) software framework [4], [5]. HyRoDyn allows solving the loop closures in the PKMs analytically. A hybrid robot is modeled as a serial composition of submechanisms module (serial or parallel). Analytical solutions to the most common PKMs used in hybrid robots are already available in HyRoDyn. However, a disadvantage of HyRoDyn is the lack of its ability in dealing with any general mechanism. For any unknown mechanism type, symbolic expressions are required for the analyses which requires expert knowledge and their derivation may be a time-consuming process.

The most naive solution of the forward dynamics involves the computation and inversion of mass-inertia matrix for computing accelerations. If the robot has several degrees of freedom, a numerical inversion of such a large mass-inertia matrix is not efficient. Other approaches have been discussed in the literature that performs forward dynamics recursively leading to $O(n)$ algorithm, where n is the number of bodies in the system. Featherstone's Articulated Body Algorithm (ABA), introduced in [6], is a recursive forward dynamics algorithm for tree type systems and is deemed to be the most efficient. But, for complex closed loop systems, generally mass-inertia matrix inversion is performed. To circumvent this, Jain introduced the idea of embedding the loop closure constraints within the ABA algorithm [7] leading to a recursive forward dynamics for constrained systems in minimal coordinates. In the recent years, the algorithms from Featherstone [8] has become increasingly popular among the robotics community. Their implementation can be found in many open source libraries such as RBDL [2], Pinocchio [9], Drake [10], Bullet [11], DART [12] etc. The main difference between the formulations from Featherstone and Jain is the numbering scheme. Featherstone's algorithms use a numbering scheme where the numbers increase from base

This work was supported by the VeryHuman (FKZ 01IW20004) and M-Rock (FKZ 01IW21002) projects funded by the German Aerospace Center (DLR) with federal funds from the Federal Ministry of Education and Research (BMBF). We additionally acknowledge the support of RIMA project (FKZ H2020-DT-2018-2020/H2020-DT-2018-1 #824990) funded by European Unions Horizon 2020 research and innovation programme. The third author acknowledges the support of the LCM-K2 Center within the framework of the Austrian COMET-K2 program.

[1]The authors are with Robotics Innovation Center, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 28359 Bremen, Germany. {r.kumar, shivesh.kumar}@dfki.de

[2]Andreas Müller is with the Institute of Robotics, Johannes Kepler University, 4040 Linz, Austria. a.mueller@jku.at

[3]Frank Kirchner is additionally affiliated with AG Robotik, University of Bremen, 28359 Bremen, Germany. frank.kirchner@dfki.de

to tip of the kinematic tree while Jain's algorithms use an opposite numbering scheme where numbering increases from tip to base of the tree. Due to the popularity of Featherstone's notation and numbering scheme, the recursive forward dynamics algorithm from Jain has received only limited attention in the robotics community.

Contributions: The numerical approach is advantageous for dealing with arbitrary mechanisms but suffer from loop closure errors and computational inefficiency. On the other hand, the analytical approach has advantages in terms of zero loop closure errors and better computational performance but are very specific to a class of parallel mechanisms. In this paper, we combine the complementary nature of numerical and analytical loop closure methods to develop a modular hybrid numerical-analytical approach which offers higher model fidelity and computational performance than a purely numerical treatment of complex series-parallel hybrid robots. This is demonstrated with a case study on the series-parallel hybrid humanoid upper body platform RH5 Manus [13] that compares the proposed approach with state of the art numerical solver RBDL [2] and analytical solver HyRoDyn [5]. As a second contribution, a reformulation of the constraint embedding formulation for ABA proposed by Jain [7] is presented using Lie group concepts and Featherstone's notations which is more commonly used in the robotics community. The authors believe that this reformulation can benefit the robotics community by making this algorithm more accessible to developers who are more used to base to tip numbering scheme. Further, it can be used with the proposed modular and hybrid numerical-analytical approach to achieve higher computational efficiency.

Organization: Section II provides the mathematical preliminaries for modeling rigid body systems with closed loops. Section III provides the hybrid numerical-analytical approach and Section IV presents the modified ABA algorithm with constraint embedding approach in a geometric framework. Section V verifies the proposed method, presents an experimental validation and provides a case study to demonstrate the achieved computational benefit. Section VI draws conclusion and presents future work.

II. Modeling of Closed Loop Systems

This section presents some preliminaries for modeling robotic systems with closed loops using graph based topological description [7], [8]. A regular numbering scheme is described in [8] according to which the bodies are numbered from root of the graph towards the tip, i.e. 0 to N_B. Assuming that the spanning tree is defined, let n denote the degrees of freedom of the spanning tree, m denote the mobility and n^c denote the number of independent loop closure constraints. Let $\mathbf{q} \in \mathbb{R}^n$ be the vector of spanning tree joints and $\mathbf{y} \in \mathbb{R}^m$ is a vector of independent joint variables that define $\mathbf{q}$ uniquely.

A. Loop Closure Constraints

Loop constraints are the non-linear constraints defined on the motion variables of the rigid body system. They can be expressed in implicit and explicit form as provided in [8].

TABLE I: Loop closure constraints [8]

Type	Position	Velocity	Acceleration
Implicit:	$\phi(\mathbf{q}) = \mathbf{0}$	$\mathbf{K}\dot{\mathbf{q}} = \mathbf{0}$	$\mathbf{K}\ddot{\mathbf{q}} = \mathbf{k}$
Explicit:	$\mathbf{q} = \gamma(\mathbf{y})$	$\dot{\mathbf{q}} = \mathbf{G}\dot{\mathbf{y}}$	$\ddot{\mathbf{q}} = \mathbf{G}\ddot{\mathbf{y}} + \mathbf{g}$

In Table I, $\mathbf{K} = \frac{\partial \phi}{\partial \mathbf{q}}$, $\mathbf{k} = -\dot{\mathbf{K}}\dot{\mathbf{q}}$, $\mathbf{G} = \frac{\partial \gamma}{\partial \mathbf{y}}$, and $\mathbf{g} = \dot{\mathbf{G}}\dot{\mathbf{y}}$. If both implicit and explicit constraints define the same constraint in the system, then it can be deduced that $\phi \circ \gamma = \mathbf{0}$, $\mathbf{KG} = \mathbf{0}$, and $\mathbf{Kg} = \mathbf{k}$. The algorithms to compute these variables are described in [2], [8] and are skipped here for brevity.

B. Constrained Equations of Motion (EOM)

In implicit form, the EOM for a constrained system is

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & \mathbf{K}^T \\ \mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\lambda \end{bmatrix} = \begin{bmatrix} \tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \\ \mathbf{k} \end{bmatrix} \tag{1}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ are position, velocity, and acceleration variables of the spanning tree, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the mass-inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$ is the Coriolis-centrifugal and gravity efforts, $\lambda \in \mathbb{R}^{n^c}$ is the Lagrangian multipliers, and $\tau \in \mathbb{R}^n$ are the joint forces.

The EOM in minimal coordinates (explicit form) is

$$\mathbf{M}_y \ddot{\mathbf{y}} + \mathbf{C}_y = \tau_y \tag{2}$$

- $\tau_y = \mathbf{G}^T \tau$ is the generalised forces (forces and torques).
- $\mathbf{M}_y = \mathbf{G}^T \mathbf{M} \mathbf{G}$ is the generalised mass-inertia matrix
- $\mathbf{C}_y = \mathbf{G}^T(\mathbf{C} + \mathbf{M}\mathbf{g})$ is the generalised Coriolis-centrifugal efforts and gravity forces.

C. Deriving Explicit from Implicit Constraints

The explicit constraints can be derived from the implicit constraints using the method of generalized coordinate partitioning (GCP) which eliminates the possibility of constraints violation [14]. In GCP, spanning tree coordinates are split into a set of chosen generalized coordinates and remaining dependent coordinates. Note that this choice is not unique in general and may not be the same as actuator coordinates.

1) Velocity Constraints: The constrained Jacobian matrix ($\mathbf{K}$) in implicit form can be rewritten by splitting it into independent and dependent coordinates part. Dependent coordinates is a subset of generalized positions that are expressed as a function of independent coordinates $\mathbf{y} = \mathbf{q}_i \in \mathbb{R}^m$ using function γ. Splitting the equation $\mathbf{K}\dot{\mathbf{q}} = \mathbf{0}$,

$$\begin{aligned} \begin{bmatrix} \mathbf{K}_i & \mathbf{K}_d \end{bmatrix} \begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\mathbf{q}}_d \end{bmatrix} &= \mathbf{0} \\ \mathbf{K}_i \dot{\mathbf{y}} + \mathbf{K}_d \dot{\mathbf{q}}_d &= \mathbf{0} \\ \dot{\mathbf{q}}_d &= -\mathbf{K}_d^{-1} \mathbf{K}_i \dot{\mathbf{y}} \end{aligned} \tag{3}$$

where $\mathbf{K}_i \in \mathbb{R}^{n^c \times m}$ and $\mathbf{K}_d \in \mathbb{R}^{n^c \times (n-m)}$ are the independent and dependent coordinates part of $\mathbf{K}$ respectively, and $\dot{\mathbf{q}}_d \in \mathbb{R}^{n-m}$ is the dependent velocity vector. In matrix form,

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \dot{\mathbf{q}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_d^{-1} \mathbf{K}_i \end{bmatrix} \dot{\mathbf{y}} \tag{4}$$

Comparing above equation with $\dot{\mathbf{q}} = \mathbf{G}\dot{\mathbf{y}}$, we have the constrained Jacobian matrix in explicit form as,

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_d^{-1} \mathbf{K}_i \end{bmatrix}. \tag{5}$$

2) Acceleration Constraints: The equation $\mathbf{K}\ddot{\mathbf{q}} = \mathbf{k}$ can be rewritten in terms of dependent and independent parts.

$$\begin{bmatrix}\mathbf{K}_i & \mathbf{K}_d\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{y}} \\ \ddot{\mathbf{q}}_d\end{bmatrix} = -\dot{\mathbf{K}}\dot{\mathbf{q}}$$
$$\mathbf{K}_i\ddot{\mathbf{y}} + \mathbf{K}_d\ddot{\mathbf{q}}_d = -\dot{\mathbf{K}}\dot{\mathbf{q}} \tag{6}$$

Multiplying by $\mathbf{K}_d^{-1}$,

$$\mathbf{K}_d^{-1}\mathbf{K}_i\ddot{\mathbf{y}} + \ddot{\mathbf{q}}_d = -\mathbf{K}_d^{-1}\dot{\mathbf{K}}\dot{\mathbf{q}}$$
$$\ddot{\mathbf{q}}_d = \mathbf{K}_d^{-1}\mathbf{k} - \mathbf{K}_d^{-1}\mathbf{K}_i\ddot{\mathbf{y}} \tag{7}$$

Substituting Eq. 5 in $\ddot{\mathbf{q}} = \mathbf{G}\ddot{\mathbf{y}} + \mathbf{g}$ and comparing with the above equation, we have

$$\mathbf{g} = \begin{bmatrix}\mathbf{0} \\ \mathbf{K_d}^{-1}\mathbf{k}\end{bmatrix}. \tag{8}$$

III. Modular and Hybrid Numerical-Analytical Approach for Series-Parallel Hybrid Robots

This section describes the hybrid numerical-analytical approach for series-parallel hybrid robots which exploit the modularity as well as domain-specific expert knowledge about the robot design.

A. Motivation

In most of the numerical approaches, the aim is to solve the implicit constraints as they are fairly easy to setup at acceleration level. As the loop closures in the system increases, loop closure errors are inevitable and the approach suffers from computational inefficiency. To control the violation of constraints, stabilization techniques like Baumgarte stabilization [8] are needed. On the other hand, explicit constraints are complex to setup as it requires the knowledge of analytical solutions to loop closure equations but overcomes the problem of computational inefficiency and numerical errors. Series-parallel hybrid robots [1] often employ a combination of serial and parallel submechanism modules connected in a tree-type fashion. This motivates us to combine the complementary nature of numerical and analytical loop closure methods to develop a hybrid numerical-analytical approach which exploits the modularity in the system design and the analytical solutions to loop closure constraints in commonly studied parallel mechanisms. The approach offers higher model fidelity and computational performance than a purely numerical treatment of complex series-parallel hybrid robots.

B. Approach

The main idea behind the approach is to exploit the modularity in the robot design to combine analytical loop closure for the known submechanisms and numerical loop closure for submechanisms where analytical solutions are not available while solving the loop closure constraints of the overall robot. This results in minimal coordinate algorithms which can be solved efficiently and recursively. In Fig. 1, the blue column follows the derivation of implicit constraints via numerical approach and orange column follows the derivation of explicit constraints via symbolic expressions

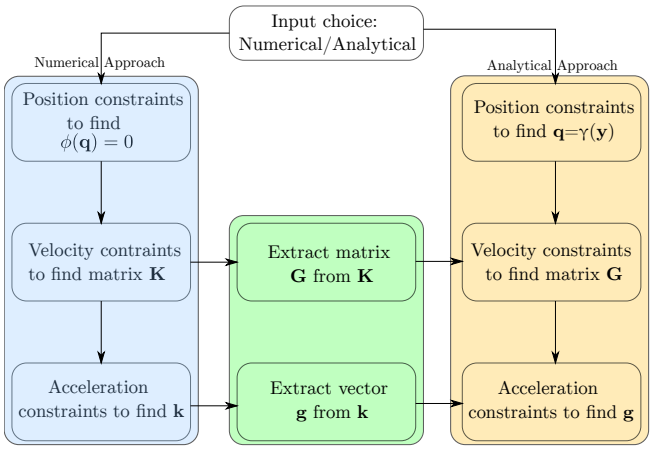


Fig. 1: Modular approach to obtain explicit constraints

for resolving the loop closures. The green layer between them depicts the idea of deriving the explicit constraints from implicit constraints numerically. This modular approach gives freedom to a user to choose a numerical or analytical approach for a parallel submechanism module. Hence, if the constraints are requested to be resolved numerically, they are expressed in explicit form using the green layer in the Fig. 1, following the methodology from Sec. II. On the contrary, if analytical approach is requested by the user, symbolic expressions are required for that particular parallel module.

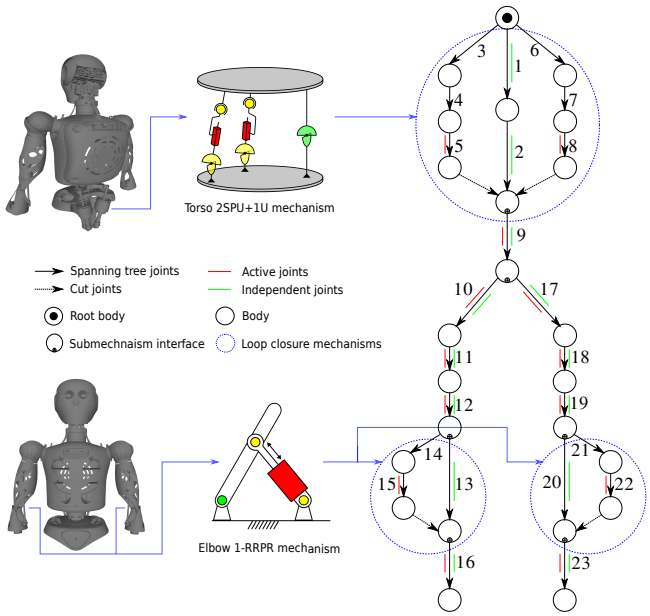


Fig. 2: Reduced version of RH5 Manus robot

C. Example

Let us consider a reduced version of RH5 Manus robot [13] as an example of series-parallel hybrid robot. In Fig. 2, the hybrid system is represented as a serial composition of eight submechanisms, defined by the user. Among these, blue circled are closed loop submechanisms while others are serial chain subsystems. The first closed loop submechanism

connected to the root of graph represents torso of the robot. It is a multi-loop mechanism of type 2SPU+1U [15] actuated by two prismatic joints. Additionally, the elbow of robot is realized with a closed loop mechanism of type RRPR actuated with a prismatic joint. If the analytical solutions to the torso mechanism is unknown and elbow mechanism is known, then the loop closure function of this hybrid robot is given by

$$\gamma = \begin{bmatrix} \gamma_{1,num}^T & \gamma_2^T & \dots & \gamma_8^T \end{bmatrix}^T \tag{9}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{1,num} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_2 & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{G}_8 \end{bmatrix} \tag{10}$$

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_{1,num}^T & \mathbf{g}_2^T & \dots & \mathbf{g}_8^T \end{bmatrix}^T. \tag{11}$$

In above equations, the first submechanism is solved numerically i.e. $\gamma_{1,num}, \mathbf{G}_{1,num}$, and $\mathbf{g}_{1,num}$ are computed using Eqs. 5 and 8. Remaining submechanisms are solved analytically using symbolic expressions.

D. Software Implementation

The proposed method exploits the numerical resolution of loop closure constraints in the RBDL [2]. The quantities $\mathbf{K}$ and $\mathbf{k}$ are obtained from RBDL and Eqs. 5 and 8 are used to get the explicit form of velocity and acceleration level constraints expressed as $\mathbf{G}_{num}$ and $\mathbf{g}_{num}$ respectively. The position constraints of the parallel kinematic chain is computed using iterative methods where the independent coordinates of the parallel kinematic chain is fixed and remaining spanning tree joint variables are allowed to change to find local feasible function ϕ. When the function is combined with independent coordinates, it is expressed as γ_{num}. On the other hand, for analytical expressions, HyRoDyn library [5] is used. The benefit of using HyRoDyn is its ability to reuse the submechanism libraries like RRPR, 2SPRR+1U, 6-UPS etc., which can be specified by the user for analytical approach.

IV. Recursive Forward Dynamics in Minimal Coordinates Using Constraint Embedding

This section presents a reformulation of the constraint embedding formulation for ABA proposed by Jain [7] which yields a recursive forward dynamics algorithm in minimal coordinates for series parallel hybrid robots. It adapts the concepts from Spatial Operator Algebra (SOA) introduced by Jain into standard Lie group concepts and root to tip regular numbering scheme from [8] which is more frequently used in the robotics community. The ABA algorithm calculates forward dynamics by computing the articulated body inertia [8]. The algorithm in [2], [8] makes three passes over the tree-type system that are

1) The first pass goes from base to tip of the kinematic tree and is responsible for calculating the velocity and bias terms of each node in the kinematic tree.
2) The second pass runs from tip to base and calculates the articulated body inertia and bias forces.
3) The third pass goes from base to tip and computes the accelerations.

A. Recursive formulations

As ABA is a well known recursive algorithm, the main algorithm is skipped here for brevity and can be followed in [7], [8]. This section only provides the adapted quantities to consider loop closure constraints. As discussed in [7], the spanning tree can be modified by considering the parallel kinematic chain as a node. This converts the spanning tree into a new spanning tree with the submechanism captured in a new node. The approach removes the bodies involved in the parallel kinematic chain and introduces a new submechanism node in the graph, $\mathscr{G}$, as shown in Fig. 3. In the figure, p denotes the parent of $\mathscr{G}$, and c denotes the child of $\mathscr{G}$ The explicit loop closure constraints shall be captured in $\mathscr{G}$. The

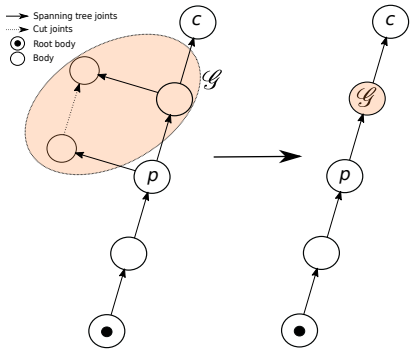


Fig. 3: Introduction of a submechanism node in the tree, adapted from [7]

constraints should be embedded in the spanning tree when going from $p \to \mathscr{G} \to c$. To compute the variables in the algorithm, screw theory and Lie group concepts are used that are well discussed in literature [8], [16]–[18]

The explicit Jacobian matrix, $\mathbf{G}_{\mathscr{G}}$, and explicit bias acceleration vector, $\mathbf{g}_{\mathscr{G}}$, can be computed for a parallel submechanism in the tree from Section III. The explicit Jacobian matrix in reduced coordinates $\mathbf{J}_{\mathscr{G}}$ needs to be computed. The new submechanism node $\mathscr{G}$ contains parallel kinematic chain bodies from the kinematic tree. The reduced Jacobian matrix with the explicit jacobian matrix $\mathbf{G}_{\mathscr{G}}$ is given by

$$\mathbf{J}_{\mathscr{G}} = \mathbf{A}_{\mathscr{G}} \mathbf{X}_{\mathscr{G}} \mathbf{G}_{\mathscr{G}} \tag{12}$$

where

$$\mathbf{A}_{\mathscr{G}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{Ad}_{T_{2,1}} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{Ad}_{T_{3,1}} & \mathbf{Ad}_{T_{3,2}} & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Ad}_{T_{n,1}} & \mathbf{Ad}_{T_{n,2}} & \dots & \mathbf{Ad}_{T_{n,n-1}} & \mathbf{I} \end{bmatrix} \tag{13}$$

with $\mathbf{Ad}_{T_i}$ is the adjoint transformation matrix of size (6×6) defined as

$$\mathbf{Ad}_{T_{i,j}} = \begin{bmatrix} \mathbf{R}_{i,j} & \mathbf{0} \\ {}^i\tilde{\mathbf{r}}_{i,j}\mathbf{R}_{i,j} & \mathbf{R}_{i,j} \end{bmatrix} \tag{14}$$

where $\mathbf{R}_{i,j} \in SO(3)$ is the rotation matrix and ${}^{i}\tilde{\mathbf{r}}_{i,j}$ is the skew symmetric matrix of position vector in frame i.

$$\mathbf{X}_{\mathscr{G}} = diag({}^{1}\mathbf{X}_1, {}^{2}\mathbf{X}_2, \ldots, {}^{n}\mathbf{X}_n) \tag{15}$$

where ${}^{i}\mathbf{X}_i$ is the screw coordinates of the vector, of joint frame i represented in body-fixed frame i.

The new graph introduces connector blocks for computing articulated body inertia and bias force for the submechanism node $\mathscr{G}$ defined as:

1) Connecting matrix from $p \to \mathscr{G}$ called as $\mathbf{A}(p,\mathscr{G})$ which maps the nodes in $\mathscr{G}$ to p.
2) Connecting matrix from $\mathscr{G} \to c$ called as $\mathbf{A}(\mathscr{G},c)$ which maps the nodes in $\mathscr{G}$ to c.

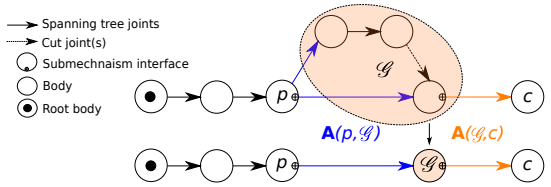


Fig. 4: Connecting matrices for new submechanism node

The connecting matrices are depicted in Fig. 4. The blue colored joints in the graph are the components used in building the connecting matrix $\mathbf{A}(p,\mathscr{G})$. Similarly, orange colored joints are the components used in building the connecting matrix $\mathbf{A}(\mathscr{G},c)$. Considering n bodies and m independent coordinates in submechanism node $\mathscr{G}$, $\mathbf{A}(p,\mathscr{G}) \in \mathbb{R}^{6n\times 6}$, and $\mathbf{A}(\mathscr{G},c) \in \mathbb{R}^{6\times 6n}$ The recursive formulation when encountering a closed loop in the system can be formulated as

1) First Pass: The first pass in the algorithm remains the same and computes velocities and bias terms.

2) Second Pass: The second loop is modified when moving from tip to base in the kinematic tree. The modifications are listed below.

From child node c to submechanism node $\mathscr{G}$ $(c \to \mathscr{G})$

$$\begin{aligned}
\mathbf{I}^{A}_{\mathscr{G}} &= \mathbf{I}_{\mathscr{G}} + \sum_{\forall c\in\mu(\mathscr{G})} \mathbf{A}^{T}(\mathscr{G},c)\mathbf{I}^{a}_{c}\mathbf{A}(\mathscr{G},c)\\
\mathbf{p}^{A}_{\mathscr{G}} &= \mathbf{p}_{\mathscr{G}} + \sum_{\forall c\in\mu(\mathscr{G})} \mathbf{A}^{T}(\mathscr{G},c)\mathbf{p}^{a}_{c}\\
\mathbf{U}_{\mathscr{G}} &= \mathbf{I}^{A}_{\mathscr{G}}\mathbf{J}_{\mathscr{G}}\\
\mathbf{D}_{\mathscr{G}} &= \mathbf{J}^{T}_{\mathscr{G}}\mathbf{U}_{\mathscr{G}}\\
\mathbf{u}_{\mathscr{G}} &= \tau_{\mathscr{G}y} - \mathbf{J}^{T}_{\mathscr{G}}\mathbf{p}^{A}_{\mathscr{G}}\\
\mathbf{c}'_{\mathscr{G}} &= \mathbf{A}_{\mathscr{G}}\mathbf{c}_{\mathscr{G}} + \mathbf{A}_{\mathscr{G}}\mathbf{X}_{\mathscr{G}}\mathbf{g}_{\mathscr{G}}\\
\mathbf{I}^{a}_{\mathscr{G}} &= \mathbf{I}_{\mathscr{G}} - \mathbf{U}_{\mathscr{G}}\mathbf{D}^{-1}_{\mathscr{G}}\mathbf{U}^{T}_{\mathscr{G}}\\
\mathbf{p}^{a}_{\mathscr{G}} &= \mathbf{p}^{A}_{\mathscr{G}} + \mathbf{I}^{a}_{\mathscr{G}}\mathbf{c}'_{\mathscr{G}} + \mathbf{U}_{\mathscr{G}}\mathbf{D}^{-1}_{\mathscr{G}}\mathbf{u}_{\mathscr{G}}
\end{aligned}$$

From submechanism node $\mathscr{G}$ to parent node p $(\mathscr{G} \to p)$

$$\begin{aligned}
\mathbf{I}^{A}_{p} &= \mathbf{A}^{T}(p,\mathscr{G})\mathbf{I}^{a}_{\mathscr{G}}\mathbf{A}(p,\mathscr{G}) + \mathbf{I}_{p}\\
\mathbf{p}^{A}_{p} &= \mathbf{A}^{T}(p,\mathscr{G})\mathbf{p}^{a}_{\mathscr{G}} + \mathbf{p}_{p}
\end{aligned}$$

where

- $\mathbf{I}_{\mathscr{G}} \in \mathbb{R}^{6n\times 6n}$ contain diagonal terms as spatial mass-inertia matrix of each body in the node $\mathscr{G}$.
- $\mathbf{I}^{A}_{\mathscr{G}} \in \mathbb{R}^{6n\times 6n}$ is the articulated body inertia of node $\mathscr{G}$. Connecting matrix $\mathbf{A}(\mathscr{G},c)$ maps child to the connecting body in node $\mathscr{G}$.
- $\mathbf{p}_{\mathscr{G}} \in \mathbb{R}^{6n}$ is the articulated bias forces of the node $\mathscr{G}$.
- $\mathbf{J}_{\mathscr{G}} \in \mathbb{R}^{6n\times m}$ is the reduced Jacobian matrix in explicit form for node $\mathscr{G}$, as in Eq. 12.
- $\tau_{\mathscr{G}y} \in \mathbb{R}^{m}$ are the generalized forces of the node $\mathscr{G}$ in independent coordinates.

Note the extra step that is introduced for the velocity product of the submechanism node, $\mathbf{c}'_{\mathscr{G}} \in \mathbb{R}^{6n}$ by combining it with the explicit constraints.

3) Third Pass: The next modifications in third loop are given below when moving from base to tip.

From parent node p to submechanism node $\mathscr{G}$ $(p \to \mathscr{G})$

$$\begin{aligned}
\mathbf{a}'_{\mathscr{G}} &= \mathbf{A}(p,\mathscr{G})\mathbf{a}_{p} + \mathbf{c}'_{\mathscr{G}}\\
\ddot{\mathbf{y}}_{\mathscr{G}} &= \mathbf{D}^{-1}_{\mathscr{G}}(\mathbf{u}_{\mathscr{G}} - \mathbf{U}^{T}_{\mathscr{G}}\mathbf{a}'_{\mathscr{G}})\\
\ddot{\mathbf{q}}_{\mathscr{G}} &= \mathbf{G}_{\mathscr{G}}\ddot{\mathbf{y}}_{\mathscr{G}} + \mathbf{g}_{\mathscr{G}}\\
\mathbf{a}_{\mathscr{G}} &= \mathbf{a}'_{\mathscr{G}} + \mathbf{J}_{\mathscr{G}}\ddot{\mathbf{y}}_{\mathscr{G}}
\end{aligned}$$

From submechanism node $\mathscr{G}$ to child node c $(\mathscr{G} \to c)$

$$\begin{aligned}
\mathbf{a}'_{c} &= \mathbf{A}(\mathscr{G},c)\mathbf{a}_{\mathscr{G}} + \mathbf{c}_{c}\\
\ddot{\mathbf{q}}_{c} &= \mathbf{D}^{-1}_{c}(\mathbf{u}_{c} - \mathbf{U}^{T}_{c}\mathbf{a}'_{c})
\end{aligned}$$

In this loop, also note the extra steps that map the explicit constraints to obtain the accelerations of all bodies in the submechanism node.

B. Mass Matrix Factorization and Inversion

The closed-form of forward dynamics uses the mass matrix factorization and inversion, inspired from [7], [19]–[21]. Considering a system with $c \to \mathscr{G} \to p$, the properties can be defined as

$$\mathsf{A} = \begin{bmatrix} \mathbf{A}_{c} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{\mathscr{G}}\mathbf{A}^{T}(\mathscr{G},c)\mathbf{A}_{c} & \mathbf{A}_{\mathscr{G}} & \mathbf{0} \\ \mathbf{A}_{p}\mathbf{A}^{T}(p,\mathscr{G})\mathbf{A}^{T}(\mathscr{G},c)\mathbf{A}_{c} & \mathbf{A}_{p}\mathbf{A}^{T}(p,\mathscr{G}) & \mathbf{A}_{p} \end{bmatrix} \tag{16}$$

Similarly, X and M are defined as

$$\mathsf{X} = \begin{bmatrix} \mathbf{X}_{c} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_{\mathscr{G}}\mathbf{G}_{\mathscr{G}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_{p} \end{bmatrix}, \quad \mathsf{M} = \begin{bmatrix} \mathbf{I}_{c} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\mathscr{G}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{p} \end{bmatrix} \tag{17}$$

The components used in recursive formulations can be written as

$$\mathsf{U} = \begin{bmatrix} \mathbf{U}_{c} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_{\mathscr{G}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{p} \end{bmatrix}, \quad \mathsf{D} = \begin{bmatrix} \mathbf{D}_{c} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\mathscr{G}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_{p} \end{bmatrix} \tag{18}$$

New system quantities, ψ and κ are defined using Eqs. 16, 17, and 18 , adapted from [7].

$$\psi = [\mathsf{I} - \mathsf{X}(\mathsf{U}\mathsf{D}^{-1})^{T}]\mathsf{A} \tag{19}$$

$$\kappa = (\mathsf{U}\mathsf{D}^{-1})^{T}\mathsf{A}_{D} \tag{20}$$

where A_D is nilpotent satisfying von-Neuman series $\mathsf{A}_D = (\mathsf{I} - \mathsf{A}^{-1})$. They satisfy the property,

$$\psi^{-1}\mathsf{A} = \mathsf{I} + \mathsf{X}\kappa\mathsf{A} \tag{21}$$

The mass-inertia matrix can be written as

$$\mathsf{M}_y = \mathsf{J}^T \mathsf{M} \mathsf{J} \tag{22}$$

where $\mathsf{J} = \mathsf{A}\mathsf{X}$, and M is defined in Eq. 17. Introducing $(\psi\psi^{-1})$ in above equation,

$$\begin{aligned} \mathsf{M}_y &= (\mathsf{A}\mathsf{X})^T (\psi\psi^{-1})^T \mathsf{M} \psi\psi^{-1} \mathsf{A}\mathsf{X} \\ \mathsf{M}_y &= \mathsf{X}^T (\psi^{-1}\mathsf{A})^T \psi^T \mathsf{M} \psi (\psi^{-1}\mathsf{A}) \mathsf{X} \end{aligned} \tag{23}$$

Using Eq. 21,

$$\begin{aligned} \mathsf{M}_y &= \mathsf{X}^T [\mathsf{I} + \mathsf{X}\kappa\mathsf{A}]^T \psi^T \mathsf{M} \psi [\mathsf{I} + \mathsf{X}\kappa\mathsf{A}] \mathsf{X} \\ \mathsf{M}_y &= [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^T \mathsf{X}^T \psi^T \mathsf{M} \psi \mathsf{X} [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}] \\ \mathsf{M}_y &= [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^T \mathsf{D} [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}] \end{aligned} \tag{24}$$

where $\mathsf{D} = \mathsf{X}^T \psi^T \mathsf{M} \psi \mathsf{X}$. Therefore, inverting Eq. 24

$$\mathsf{M}_y^{-1} = [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^{-1} \mathsf{D}^{-1} [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^{-T} \tag{25}$$

From standard matrix identity $[\mathsf{I} + \mathsf{A}\mathsf{B}]^{-1} = \mathsf{I} - \mathsf{A}[\mathsf{I} + \mathsf{B}\mathsf{A}]^{-1}\mathsf{B}$,

$$\begin{aligned} [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^{-1} &= \mathsf{I} - \kappa\mathsf{A}[\mathsf{I} + \mathsf{X}\kappa\mathsf{A}]^{-1}\mathsf{X} \\ [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^{-1} &= \mathsf{I} - \kappa\mathsf{A}(\psi^{-1}\mathsf{A})^{-1}\mathsf{X} \\ [\mathsf{I} + \kappa\mathsf{A}\mathsf{X}]^{-1} &= \mathsf{I} - \kappa\psi\mathsf{X} \end{aligned} \tag{26}$$

Therefore, using Eq. 26 and substituting in Eq. 25, mass-matrix inversion can be written using factorization as

$$\mathsf{M}_y^{-1} = [\mathsf{I} - \kappa\psi\mathsf{X}] \mathsf{D}^{-1} [\mathsf{I} - \kappa\psi\mathsf{X}]^T \tag{27}$$

The mass-inertia matrix factorization and inversion is captured in Eq. 27.

V. RESULTS

This section presents the numerical verification, experimental validation and a case study on computational efficiency of the proposed approach using the example of an upper body series-parallel hybrid RH5 Manus robot [1] which is the successor of RH5 Humanoid [22] (also see the accompanying video).

A. Verification of Hybrid Numerical-Analytical Approach

The modular and hybrid implementation of explicit constraints is verified by comparing it with analytical solutions of the closed loops. In order to do this, the reduced version of the upper body of RH5 Manus is considered as in Fig. 2. The robot model consists of $m = 13$ independent joints, $n = 23$ spanning tree joints, and 13 actuated joints. We consider a case where torso submechanism is solved using numerical approach and remaining loop closures with analytical solutions. A total of $n^c = 6$ constraints need to be defined for numerically solving this loop closure in the system. Cycloidal trajectories [23] are defined for the independent joints. For verifying the numerical resolution of loop closures, kinematic analysis and inverse dynamics are performed. In the actuation space of numerically resolved parallel torso mechanism, we have two prismatic actuators and the root mean squared error (RMSE) are reported in Table II. From these results, it can be concluded that explicit constraint resolution from numerical approach is feasible and does not compromise on accuracies.

[1]URDF models are released at: `https://github.com/dfki-ric-underactuated-lab/hybrid_numerical_analytical_approach_case_study` for benchmarking.

TABLE II: RMSE between proposed and analytical methods

Actuated joints	Torso left (Joint 5)	Torso right (Joint 8)
Position (m)	2.36e-08	3.77e-08
Velocity (m/s)	1.14e-09	1.11e-09
Acceleration (m/s^2)	2.47e-11	4.64e-11
Force (N)	1.15e-04	1.13e-04

B. Experimental Results

For experimental verification, the reduced version of the upper body of RH5 Manus is considered as before. The independent joint trajectories are provided for boxing motion generated using optimal control [13]. In Fig. 5, the robot is shown in action for the boxing motion trajectories.

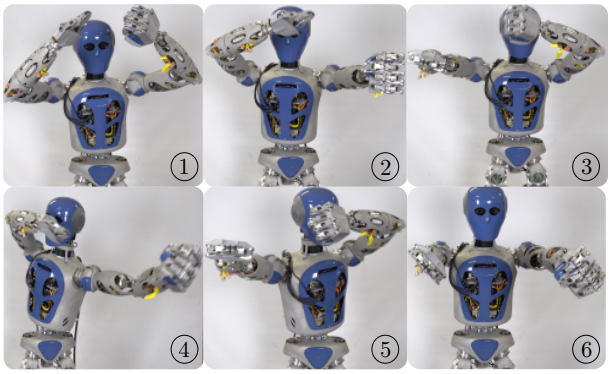
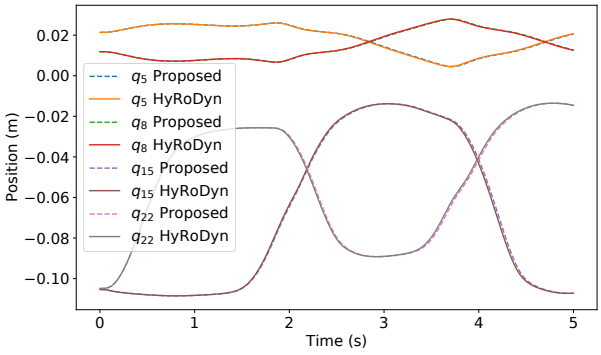


Fig. 5: RH5 Manus robot in boxing motion

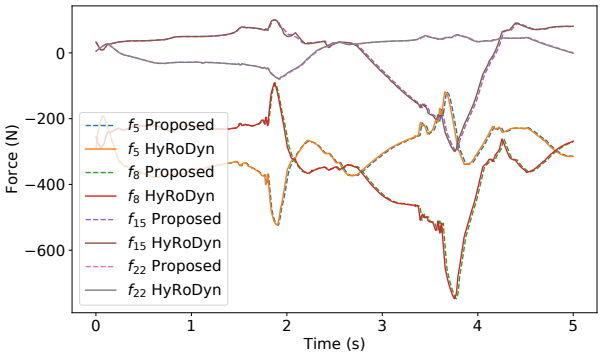
The proposed method is used to map the independent joint movements to the actuation space at the position, velocity, acceleration, and torque levels. For experimental verification, the data is collected from both approaches, i.e. analytical approach from HyRoDyn and the proposed hybrid approach. In Fig. 6, it can be seen that the trajectories can be tracked with the proposed method without any significant loss of performance. Hence, the proposed approach is also suitable for real time control of complex series-parallel hybrid robots.

C. Computational Performance

a) Reduced RH5 Manus Robot Model: The computational performance is measured in CPU time using CHRONO library in C++ for the reduced version of the RH5 Manus robot. For this robot with $n = 23$ spanning tree joints, CPU time is noted for position, velocity, acceleration, and torque analysis for randomly generated trajectories. A total number of 10000 calls were made to solve the full system state (position, velocity, acceleration), inverse dynamics, and forward dynamics. The program is run on a standard laptop with Ubuntu 20.04 operating system and Intel Core i9-11950H CPU @ 2.6 GHz. For computational performance, the following three cases are studied.



(a) Position plots



(b) Force plots

Fig. 6: Experimental results

1) The full system is solved through numerical approach using RBDL ($n^c = 10$).
2) The full system is solved through analytical approach using HyRoDyn ($n^c = 0$).
3) The full system is solved through proposed approach where one closed loop mechanism (torso mechanism in Fig. 2) is solved numerically and other parallel mechanisms are solved analytically ($n^c = 6$).

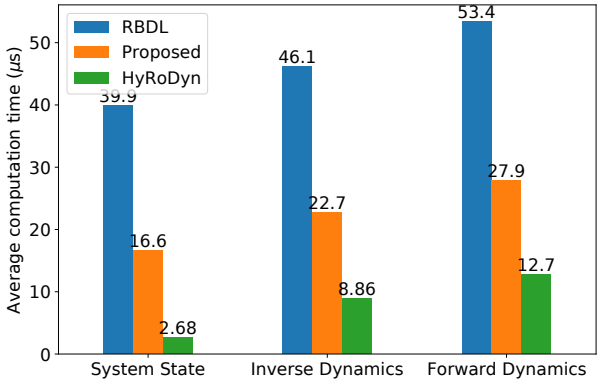


Fig. 7: Comparison of CPU times between HyRoDyn, RBDL, and proposed methods for reduced RH5 Manus robot

From Fig. 7, the computational time for inverse dynamics of the numerical approach (RBDL) is ≈ 5 times slower than the analytical approach (HyRoDyn). For the same, it can be noted that proposed numerical-analytical approach performs ≈ 2 times better than the numerical approach (RBDL) which shows the advantage of the proposed method.

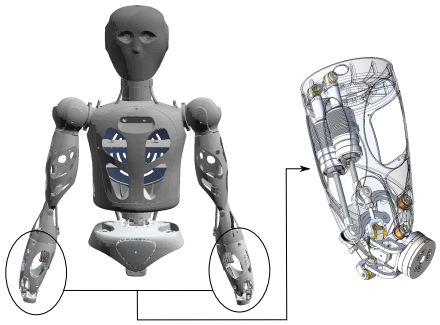


Fig. 8: Wrist mechanism in RH5 Manus robot [24]

b) Full Upper body of RH5 Manus Model: The computational performance is studied for the fixed-base full upper body of RH5 Manus robot model ($n = 61$ spanning tree joints, $m = 20$ independent joints, and $p = 20$ actuated joints) with a parallel wrist mechanism whose symbolic expressions are not yet available in the database of HyRoDyn software. The mechanism, as seen in Fig. 8, is complex in terms of loop closures where multiple cut joints define the constraints. The mechanism consists of 18 spanning tree joints, 2 independent joints, and 2 prismatic actuated joints. It is of type 2SU[RRPR]+1U [24] and imposes a total number of $n^c = 10$ constraints. The whole complex system of the upper body of RH5 Manus robot model now require $n^c = 30$ constraints to be defined. Following cases are studied:

1) The full system is solved through numerical approach from RBDL including the wrist ($n^c = 30$).
2) The full system is solved through proposed approach ($n^c = 20$).

From Fig. 9, the computational performance for the proposed numerical-analytical approach is ≈ 1.3 times better when compared to RBDL, for computing the inverse dynamics.

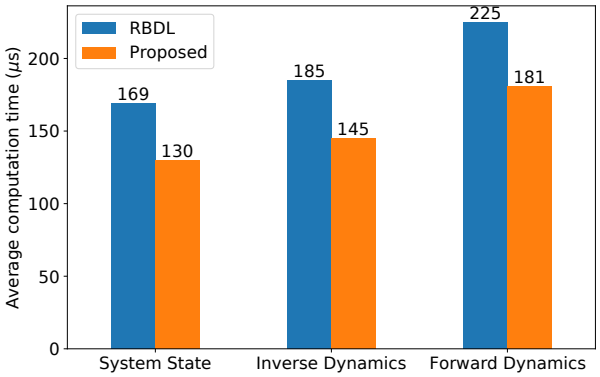


Fig. 9: Comparison of proposed and RBDL approaches for full upper body of RH5 Manus robot model

D. Validation of Constraint Embedding Approach

The constraint embedding approach is validated using the direct inversion of joint-space mass inertia matrix to solve the

forward dynamics of the whole system. The same example is considered and the input trajectories for positions and velocities are given by cycloidal trajectories. The accelerations are computed through both the approaches and the results are compared. In Fig. 10, the legend *direct* computes the

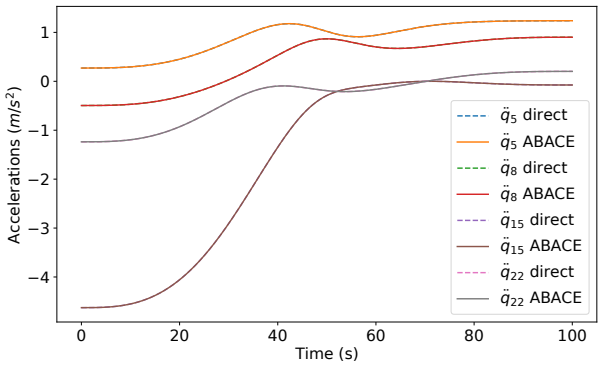


Fig. 10: Validation of constraint embedding method

acceleration using direct inversion of mass inertia matrix, and legend *ABACE* computes the acceleration using the constraint embedding approach in ABA. The root mean squared error for all the joints is computed to be zero, thus validating the constraint embedding reformulation.

VI. CONCLUSION

This paper introduces a case study on a modular and hybrid numerical-analytical approach for resolving loop closures in series-parallel hybrid robots which is more computationally efficient than a purely numerical approach. When there is a complex closed loop submechanism in the system whose symbolic solutions can be difficult to compute, the hybrid approach can act as a decent alternative. The proposed approach is applicable to all kind of systems (including floating base) and higher computational efficiency helps in improving model predictive and whole body control of such complex systems. The paper also presents a reformulation of the constraint embedding formulation for ABA proposed by Jain by translating SOA concepts into standard Lie group concepts and adapting the recursive algorithm for base to tip numbering scheme. This reformulation can benefit the robotics community by making this algorithm more accessible to developers who are comfortable with base to tip numbering scheme and Featherstone's notation. The future work include the implementation of contact dynamics.

REFERENCES

[1] S. Kumar, H. Whrle, J. de Gea Fernndez, A. Müller, and F. Kirchner, "A survey on modularity and distributivity in series-parallel hybrid robots," *Mechatronics*, vol. 68, p. 102367, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415820300477

[2] M. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," *Auton Robot 41*, vol. 322, no. 10, p. 495511, 2017.

[3] S. Delp, F. Anderson, A. Arnold, P. Loan, A. Habib, C. John, E. Guendelman, and D. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 1940–1950, 2007.

[4] S. Kumar, "Modular and analytical methods for solving kinematics and dynamics of series-parallel hybrid robots," Ph.D. dissertation, Universität Bremen, 01 2019.

[5] S. Kumar, K. von Szadkowski, A. Müller, and F. Kirchner, "An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots," *Journal of Mechanisms and Robotics (JMR)*, vol. 12, no. 2, pp. 1–12, 4 2020.

[6] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, pp. 826–834 vol.1.

[7] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*. Boston, MA: Springer US, 2011.

[8] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.

[9] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[10] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[11] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[12] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018. [Online]. Available: https://doi.org/10.21105/joss.00500

[13] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, and F. Kirchner, "Introducing RH5 Manus: A Powerful Humanoid Upper Body Design for Dynamic Movements," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8540–8546.

[14] S. A. Marques, F. and P. Flores, "On the constraints violation in forward dynamics of multibody systems," *Multibody System Dynamics.*, no. 39, p. 385419, 2017.

[15] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, and F. Kirchner, "Kinematic analysis of a novel parallel 2sprr+1u ankle mechanism in humanoid robot," in *Advances in Robot Kinematics 2018*, J. Lenarcic and V. Parenti-Castelli, Eds. Cham: Springer International Publishing, 2019, pp. 431–439.

[16] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st ed. USA: Cambridge University Press, 2017.

[17] A. Mueller, "Screw and lie group theory in multibody kinematics," *Multibody System Dynamics.*, no. 43, pp. 37–70, 2018.

[18] A. Mueller and S. Kumar, "Closed-form time derivatives of the equations of motion of rigid body systems," *Multibody System Dynamics.*, no. 53, pp. 257–273, 2021.

[19] G. Rodriguez, "Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics," *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 624–639, 1987.

[20] G. Rodriguez and K. Kreutz-Delgado, "Spatial operator factorization and inversion of the manipulator mass matrix," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 65–76, 1992.

[21] K. Kreutz-Delgado, A. Jain, and G. Rodriguez, "Recursive formulation of operational space control," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1750–1753.

[22] J. Eßer, S. Kumar, H. Peters, V. Bargsten, J. d. G. Fernandez, C. Mastalli, O. Stasse, and F. Kirchner, "Design, analysis and control of the series-parallel hybrid RH5 humanoid robot," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, pp. 400–407.

[23] S. Kumar, C. Rajeevlochana, and S. K. Saha, "Realistic modeling and dynamic simulation of kuka kr5 robot using recurdyn," *6th Asian Conference on Multi-Body Dynamics*, 08 2012.

[24] C. Stoeffler, A. del Rio Fernandez, H. Peters, M. Schilling, and S. Kumar, "Kinematic Analysis of a Novel Humanoid Wrist Parallel Mechanism," in *Advances in Robot Kinematics 2022*, O. Altuzarra and A. Kecskeméthy, Eds. Cham: Springer International Publishing, 2022, pp. 348–355.