2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

A Framework for Modeling Closed Kinematic Chains with a Focus on Legged Robots

Vinay R. Kamidi[1], Adam Williams[1] and Pinhas Ben-Tzvi[2], *Senior Member, IEEE*

***Abstract*— This paper presents the foundations of a MATLAB framework for dynamic modeling and simulation of closed kinematic chain (CKC) mechanisms, with a particular focus on implementation with legged locomotive mechanisms. As such, the framework supports both floating-base and fixed-base systems. Through the use of singular perturbation theory, various CKC mechanisms can be modeled so that constraint errors asymptotically converge to zero, thus avoiding the numerical drift that plagues commonly used methods. A functional API and the relevant core commands necessary to construct a model are presented. Two robotic legs incorporating CKC mechanisms are utilized as case studies, and simulations of each leg performing a dynamic monopedal gait are illustrated.**

I. INTRODUCTION

In dynamic environments and applications, CKC mechanisms are advantageous due to their superior performance as a consequence of the high rigidity these structures exhibit. Moreover, CKC mechanisms often result in lightweight architectures and as a result, a surge of CKC mechanisms can be found in a variety of fields [1]–[3]. However, the lack of robust foundational tools for modeling these mechanisms serves to create a barrier to systematic analyses, and therefore to even more widespread implementation, of CKC mechanisms. Our work thus aims to provide the community with a toolbox for handling the dynamics of CKC robots and enabling further adoption of these versatile systems in new and innovative applications.

A notable field in which CKC mechanisms provide significant benefits in design and operation is that of legged locomotion. High-profile legged robots that exhibit such structures include the KAIST Raptor [4], Minitaur [5], Super Mini Cheetah [6], SALTO [7] and ATRIAS [8]. Additionally, the author's Bio-inspired One-degree of freedom Leg for Trotting (BOLT) [9] incorporates CKCs, as seen in Fig. 1(a). These systems are all highly dynamic in nature and as such warrant extremely high torque motors. However, the inclusion of such large motors can lead to an unfortunate cycle in which the larger motors increase the system mass, thus requiring ever larger motors. The inclusion of CKCs enable proximally located actuation and exhibit high rigidity-to-weight ratios, thus reducing the reflected mass of the leg and aiding in fast locomotion. If a CKC consists of a parallelogram linkage geometry, such as that of ATRIAS,

*This material is based upon the work supported by the National Science Foundation under Grant No. 1557312

[1]The authors are with with the Robotics and Mechatronics Lab at Virginia Tech, Blacksburg, VA, USA. {vinay28,aw13}@vt.edu

[2] P. Ben-Tzvi is with the Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA, USA. bentzvi@vt.edu

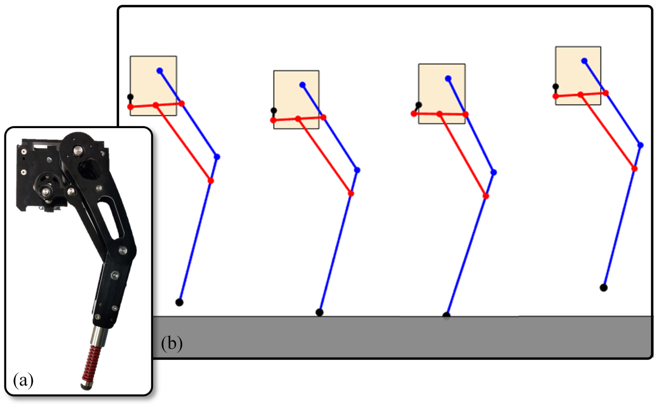


Fig. 1. (a) Bio-inspired One-degree of freedom Leg for Trotting (BOLT) (b) Portion of the hybrid dynamic cycle for a simulated monopedal running gait performed by BOLT.

it may be simplified to an open kinematic chain (OKC) model using common modeling methods. Once in OKC form, modeling can be conducted through any number of methods or softwares. However, for any other construction, modeling becomes highly implicit [10] due to the mechanism's characterization by a differential algebraic equation (DAE). The difficulties modeling DAEs and implementing them in real-time, model-based controllers is well documented [11]. For instance, while one can create models in commercial modeling software such as MSC ADAMS, real-time implementation requires plant dynamics that cannot be extracted from the software and therefore restricting its use to simulation environments.

Previously implemented packages for dynamic modeling of kinematic chains have primarily focused on OKCs. Prominent examples of such packages include proNEu, DRAKE, and FROST [12]–[14]. These packages have varying use cases, including legged locomotion. However, none of these platforms are equipped for efficient analysis of CKCs, and in some cases have no capability to model them at all. The closest one comes to achieving this is FROST, which models CKCs using the Lagrangian multiplier method. However, this method is prone to introduce numerical drift into the constraints, which can accumulate over time and thus cause the constraint to stray from its defined position.

While the numerical drift present in the commonly used modeling approaches can be addressed through methods such as Baumgarte's, this further increases complexity and requires parameter tuning [15]. Such intricacies render these approaches unfit for real-time co-simulation, model-based control, and field implementations; all domains into which legged robotic research is progressing. In response to these challenges, Singular Perturbed Formulation (SPF) [16] presents a superior approach. This method, when applied to CKCs, results in computationally lightweight modeling that asymptotically drives constraint drift to zero.

In this paper, a framework for modeling CKCs through the application of SPF is presented[1]. The methods are generalized for n-loop CKC modeling, and presented in a format conducive to dynamic modeling in research. The package is implemented in MATLAB due to its widespread use by academics. Direct knowledge of mechanism kinematics through precise modeling using the proposed framework can enable effective optimization of robot architectures. Minimal drift modeling will allow researchers to not only simulate their mechanism, but to use the tool as a sandbox for developing control architectures for robots implementing CKCs. For instance, controllers targeting the instability arising in CKC leg structures can be explored by enabling access to the full state of the robot through the accurately simulated dynamics. While the package is in active development to accommodate or port with software that supports OKC structures to achieve completeness, the authors are presenting in the hope that fellow researchers can benefit by allowing their focus to be on open problems in the CKC domain beyond dynamics.

In Section II of the paper, the underlying mathematics behind SPF are introduced, along with a short explanation of application of SPF to kinematic chains. Section III describes the package API and provides the core commands necessary to develop a dynamic model. This is followed by case studies in Section IV that demonstrate the framework on two CKC mechanisms, with the future plans for the framework concluding the paper in Section V.

II. Mathematical Foundation of Singularly Perturbed Formulation

The presented framework utilizes SPF in order to model the dynamics of closed kinematic chains. The method uses well-known Lagrangian method to develop basic dynamic equations, then approximates the algebraic constraint relations as ordinary differential equations (ODEs). Modeling of the system can then be performed through solution of a system of reduced order ODEs, rather than DAEs. Application of SPF results in a final system of equations of order corresponding to the active degrees of freedom.

A. First Order Dependencies

To arrive at a set of equations that characterize the system as a DAE, the first step is to separate the mechanism virtually [17]. The goal of virtual separation is to "cut" the mechanism into a minimal number of open or branched kinematic chains. An example is shown on the four-bar mechanism in Fig. 2. The separation point is seen in Fig. 2(b), labeled as e. Following separation, the resultant mechanism consists of two open-chain structures, shown in blue and red along with their respective link variables and angles with respect to the

[1]The framework is available at `https://github.com/RM-Lab/CKC_Framework`

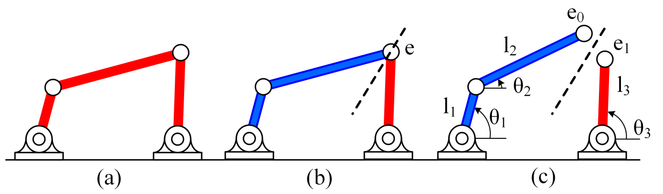


Fig. 2. Illustration of the virtual separation method on a (a) four-bar mechanism. (b) Identification of two distinct links and selection of a separation joint. (c) Two OKC chains are formed that can be modeled using existing methods.

horizontal in Fig. 2(c). In this case, the kinematic constraint relationships should satisfy the relation $e_0 = e_1$, in the planar frame. This is mathematically shown in (1)

$$\begin{bmatrix} l_1\cos(\theta_1)+l_2\cos(\theta_2)-l_3\cos(\theta_3) \\ l_1\sin(\theta_1)+l_2\sin(\theta_2)-l_3\sin(\theta_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1}$$

Following application of virtual separation, the generic DAE that describes the system dynamics can be generalized by (2).

$$\begin{cases} \mathbf{H}'(\mathbf{q}_\mathrm{d})\ddot{\mathbf{q}} + \mathbf{C}'(\mathbf{q}_\mathrm{d},\dot{\mathbf{q}}_\mathrm{d})\dot{\mathbf{q}}_\mathrm{d} + \mathbf{g}'(\mathbf{q}_\mathrm{d}) = \mathbf{B}\tau + \mathbf{F}_\mathrm{ext} \\ \phi(\mathbf{q}_\mathrm{d}) = 0 \end{cases} \tag{2}$$

where $\mathbf{q}_\mathrm{d}$ is the combination of independent and dependent variables required to completely describe the configuration space of a CKC, $\mathbf{H}'(\mathbf{q}_\mathrm{d}) \in \mathbb{R}^{n_{q_\mathrm{d}} \times n_{q_\mathrm{d}}}$ represents the generalized positive definite mass matrix, $\mathbf{C}'(\mathbf{q}_\mathrm{d},\dot{\mathbf{q}}_\mathrm{d}) \in \mathbb{R}^{n_{q_\mathrm{d}}}$ contains the Coriolis and centrifugal terms, $\mathbf{g}'(\mathbf{q}_\mathrm{d}) \in \mathbb{R}^{n_{q_\mathrm{d}}}$ is the gravitational term, $\mathbf{B}$ is the torque distribution matrix, $\tau \in \mathbb{R}^{n_{q_\mathrm{d}}}$ is the torque vector provided by the actuators, $\mathbf{F}_\mathrm{ext}$ is the external force mapped to the joint space, and $\phi(\mathbf{q}_\mathrm{d})$ contains the algebraic relations that encode the constraints. Further, $\mathbf{q}$ is a set of independent variables and the significance of the need of a separate vector is made clear later in this section. Note that the $'$ indicates the presence of dependent variables in the corresponding matrix.

The encoding of the algebraic constraints as ODEs are the driving motivation behind this method. Our primary goal is then to eliminate the coupling of the dependent variables in order to explicitly calculate the first order terms and remove the second order terms. Therefore, we divide the independent and dependent variables into $\mathbf{q}$ and $\mathbf{z}$ vectors, respectively. The first and second order dependencies are handled separately, with the initial focus on the first order coupling. To do so, we introduce a variable, $\mathbf{w} := \phi(\mathbf{q}_\mathrm{d})$, which records the constraint error. As $\mathbf{w}$ is a virtual variable, its inherent dynamics are flexible. Therefore, $\dot{\mathbf{w}}$ is chosen such that $\mathbf{w}$ is asymptotically driven to zero through relation:

$$\dot{\mathbf{w}} = -\frac{1}{\varepsilon}\mathbf{w} \tag{3}$$

where ε is a small quantity greater than 0. This then has the effect of driving the constraint error to exponential zero, controlled by the value of ε. By substituting and applying the definition of $\mathbf{w}$ to (3), the relation becomes:

$$\dot{\mathbf{z}} = -\mathbf{J}_\mathrm{z}^{-1}\left(\frac{1}{\varepsilon}\phi(\mathbf{q}_\mathrm{d}) + \mathbf{J}_\mathrm{q}\dot{\mathbf{q}}\right) \tag{4}$$

where $\mathbf{J}_z$ and $\mathbf{J}_q$ are the Jacobians corresponding to the dependent and independent variables, respectively. This equation encapsulates the SP dynamics, enabling the explicit calculation of $\dot{\mathbf{z}}$ and driving the constraint error to zero.

B. *Second Order Dependencies*

While the first order dependent states are explicitly calculated in the SP dynamics, the second order dependent states must still be dealt with. First, two selector matrices, $\mathbf{S}_\mathrm{q}$ and $\mathbf{S}_\mathrm{z}$, are utilized to encode the relation between $\mathbf{q}$ and $\mathbf{z}$, such that $[\mathbf{q}\ \mathbf{z}]^T = [\mathbf{S}_\mathrm{q}\ \mathbf{S}_\mathrm{z}]^T$. Next, two dimensionality reduction terms, $\Gamma(\mathbf{q}_\mathrm{d})$ and $\rho(\mathbf{q}_\mathrm{d})$, are introduced such that:

$$\Gamma(\mathbf{q}_\mathrm{d}) := \begin{bmatrix} \phi(\mathbf{q}_\mathrm{d}) \\ \mathbf{S}_\mathrm{q}(\mathbf{q}_\mathrm{d}) \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{q} \end{bmatrix} \tag{5}$$

and utilizing $\Gamma_{\mathrm{q_d}}(\mathbf{q}_\mathrm{d}) = {}^{\partial\Gamma(\mathbf{q}_\mathrm{d})}/_{\partial\mathbf{q}_\mathrm{d}}$.

$$\rho(\mathbf{q}_\mathrm{d}) := \Gamma_{\mathrm{q_d}}(\mathbf{q}_\mathrm{d})^{-1} \begin{bmatrix} 0 \\ \mathbf{I}_{n_\mathrm{q}\times n_\mathrm{q}} \end{bmatrix} \tag{6}$$

Utilizing these terms, dimensionality reduction is achieved through the application of the following relations.

$$\mathbf{H}(\mathbf{q},\mathbf{z}) = \rho(\mathbf{q}_\mathrm{d})^T \mathbf{H}'(\mathbf{q}_\mathrm{d}) \tag{7a}$$

$$\begin{aligned} \mathbf{C}(\mathbf{q}_\mathrm{d},\dot{\mathbf{q}}_\mathrm{d}) =& \rho(\mathbf{q}_\mathrm{d})^T \mathbf{C}'(\mathbf{q}_\mathrm{d},\dot{\mathbf{q}}_\mathrm{d})\rho(\mathbf{q}_\mathrm{d}) + \\ & \rho(\mathbf{q}_\mathrm{d})^T \mathbf{H}'(\mathbf{q}_\mathrm{d})\dot{\rho}(\mathbf{q}_\mathrm{d},\dot{\mathbf{q}}_\mathrm{d}) \end{aligned} \tag{7b}$$

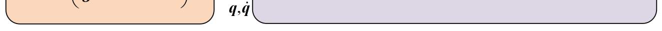
$$\mathbf{g}(\mathbf{q}_\mathrm{d}) = \rho(\mathbf{q}_\mathrm{d})^T \mathbf{g}'\mathbf{q}_\mathrm{d} \tag{7c}$$

The resulting real coordinate spaces $\mathbf{H}(\mathbf{q}_\mathrm{d}) \in \mathbb{R}^{n_\mathrm{q}\times n_\mathrm{q}}$, $\mathbf{C}(\mathbf{q}_\mathrm{d}) \in \mathbb{R}^{n_\mathrm{q}\times n_\mathrm{q}}$, $\mathbf{g}(\mathbf{q}_\mathrm{d}) \in \mathbb{R}^{n_\mathrm{q}\times n_\mathrm{q}}$ do not contain second order coupling terms. Combining these with the explicitly calculated first order dependent terms from (4) give the approximated ODE dynamic model for a CKC in (8). The mathematical model is illustrated in Fig. 3.

$$\begin{bmatrix} \mathbf{H}(\mathbf{q},\mathbf{z}) & 0_{n_\mathrm{q}\times 1} \\ 0_{n_\mathrm{z}\times 1} & \mathbf{J}_\mathrm{z}(\mathbf{q},\mathbf{z}) \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{z},\dot{\mathbf{z}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q},\mathbf{z}) + \mathbf{B}\tau \\ -\frac{1}{\varepsilon}\phi(\mathbf{q},\mathbf{z}) - \mathbf{J}_\mathrm{q}(\mathbf{q},\mathbf{z})\dot{\mathbf{q}} \end{bmatrix} \tag{8}$$

III. Implementation and Framework API

In order to input a model into the presented framework, it is first necessary to determine several requisite physical quantities, such as link lengths, mass properties, constraint equations, actuated and unactuated joints, and the initial state variable conditions. Additionally, the user must determine the virtual separation points and the corresponding kinematic relations. The output from the package is a MATLAB function file that encapsulates the code necessary to conduct a dynamic simulation of the mechanism, when coupled with a controller. If users so choose, the option is present to output the general and SP dynamic functions for implementation in other platforms or softwares.



A. *Hybrid Dynamic Modeling*

The SP formulation supports both fixed- and floating-base dynamics, and as such the framework here has the capacity to model both. However, the nature of floating-base dynamics warrants extra treatment that is provided by a hybrid dynamic modeling approach, similar to that found in [18].

Hybrid systems are characterized by behavior that can be divided into continuous and discrete dynamic states. In the case of a leg mechanism, when the leg is not in contact with the ground it is considered to be in the flight phase, while during contact the leg is in the stance phase. Both are continuous states. The transition between the two continuous phases is triggered by a discrete event, i.e. contacting or leaving contact with the ground. Modeling such dynamic systems can be achieved through the application of a state machine, where the continuous dynamics are treated as individual states that are switched between when a discrete trigger occurs.

In order to model floating-base dynamics in the presented package, the user can input any number of continuous states along with their corresponding uni-directional switching conditions. Using the example of a monopod hopper with point feet, the hybrid system is represented by the tuple, $\mathscr{H} = (D, S, \Delta, F)$, where D is the set of continuous states D_f and D_s, corresponding to the flight and stance respectively, S is the switching condition, Δ is the reset map, and F is the evolved initial conditions, all of which have the same inclusive sets as D. Thus given the switching conditions, S, the framework can then utilize the input mechanism properties to determine the unknowns in $\mathscr{H}$ in every cycle.

B. *Core API Classes*

The system dynamics are captured through the implementation of a series of core classes within the package.

Continuous Dynamics: The general dynamics of the system are contained by the class `ContinuousDynamics` in the package. The continuous states as described above are modeled using this class within the hybrid framework implementation. Each state has its inherent dynamics determined via the application of the Lagrangian formulation, calculated on the virtually separated model. The ODEs determined through the Lagrangian formulation are equal in number to the number of links present in the provided mechanism, n_{q_d}. The ODEs are coupled with the constraint equations provided by the user to arrive at the general DAE, (2). Additionally, the separation of the system variable vector, $\mathbf{q}_\mathrm{d}$, into its independent, $\mathbf{q}$, and dependent, $\mathbf{z}$, component vectors is implemented.

In order for the DAE to be approximated as a series of ODEs through the application of SPF, a `FastDynamics` sub-class is introduced. Within this sub-class, the constraint equations are abstracted by $\mathbf{w}$, and the chosen dynamics are imposed upon the system of equations, as in (3). The value of ε can be adjusted from the default by the user as required by their desired rate of convergence for the constraint error. Following the imposition of constraint dynamics, the first order dependencies are calculated as in (4). The sub-class has the facility to create an output function which provides the equations necessary to generate the vectors $\mathbf{q}, \dot{\mathbf{q}}, \mathbf{z}, \dot{\mathbf{z}}$ for implementation outside the provided framework.

To arrive at the reduced order ODEs, the sub-class `DimReduction` is applied to the $n_{q_d}-$dimensional ODEs generated from the Lagrangian formulation. The sub-class uses the results of the separation of independent and dependent variables to calculate (5) and (6) in order to apply (7a-c) to the original set of ODEs. The result is the n_q-dimensional system of ODEs. The sub-class also possesses the functionality to output a function with the reduced order dynamic matrices for external use, should the user so choose.

Finally, the `ContinuousDynamics` class utilizes the outputs of the two sub-classes to output the states of the systems given initial conditions, should the user choose to simulate within the provided framework.

***Discrete Dynamics*:** In order to simulate floating based system, a second class, `TransitionDynamics`, is included to generate a hybrid dynamic model. This class has two functionalities. The first is to output the matrices utilized to create a transition map at each switching condition. With knowledge of the continuous states, Δ, the post-switch map can be calculated as in [19]. Treating the pre-switch state as $\mathbf{q}^-$ and the post-switch state as $\mathbf{q}^+$, the two are considered continuous across both sides of the transition. Utilizing this continuity, the post-switch velocity, $\dot{\mathbf{q}}^+$, are determined through $x^+ = \Delta x^-$, where $x : \{\mathbf{q}, \dot{\mathbf{q}}\}$. In the hybrid system, $\mathbf{q}^+$ is equivalent to F.

The second functionality of the class manifests during simulation within the framework. It is continuously called to calculate the Δ and F required by the hybrid system and to generate the initial conditions required by the `ContinuousDynamics` class in the next iteration.

***Simulation*:** In the `Simulation` class, the current version of the framework does not contain a native control paradigm. This is by design, as the goal is to provide a testbed in which controllers can be developed and simulated for CKCs. Given the states, $\mathbf{q}$, output by the modeled dynamics, the user has the ability to write a function within the framework that provides a control input, τ.

The class will output a data file containing state data for the duration of simulated timespan. In addition, a primitive visualization using MATLAB's plotting capabilities to provide a validatory animation of the system. The layout of the framework can be seen in Fig. 4

C. User Inputs

The framework requires the user to provide an illustrative selection of properties and equations in order for it to successfully model the CKC mechanism of interest. Due to the lack of support for a user-friendly wrapper for mechanism property exportation capable of handling the self-referential nature of CKCs, a template file for the required user inputs is provided within the framework. As support for 3-dimensional models is under development, the following section describes the required inputs for planar mechanisms.

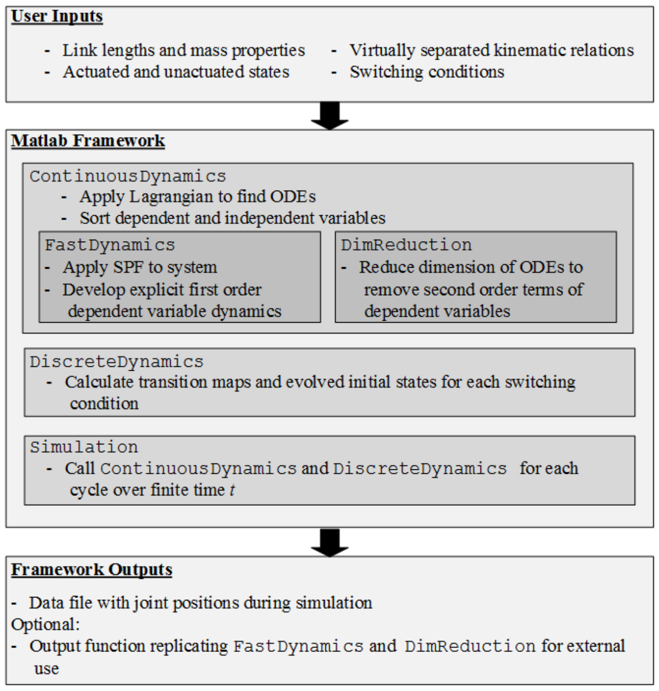

User Inputs
- Link lengths and mass properties
- Actuated and unactuated states
- Virtually separated kinematic relations
- Switching conditions
Matlab Framework
ContinuousDynamics
- Apply Lagrangian to find ODEs
- Sort dependent and independent variables
FastDynamics
- Apply SPF to system
- Develop explicit first order dependent variable dynamics
DimReduction
- Reduce dimension of ODEs to remove second order terms of dependent variables
DiscreteDynamics
- Calculate transition maps and evolved initial states for each switching condition
Simulation
- Call ContinuousDynamics and DiscreteDynamics for each cycle over finite time t
Framework Outputs
- Data file with joint positions during simulation
Optional:
- Output function replicating FastDynamics and DimReduction for external use

Fig. 4. Programmatic structure of CKC MATLAB framework. A template is provided to guide the formatting of the user inputs, and contains the necessary options for various configurations and outputs. During simulation, the continuous and discrete dynamics are iteratively evaluated to provide a hybrid dynamic environment.

Irrespective of whether the user's goal is simulation or extraction of dynamic functions, the user supplies the physical properties of the links, i.e. mass, length, inertia with respect to the axis of rotation, and location of center of mass along the link. Maintaining the link order used to determine the mass physical properties, the user supplies the vector of chosen state variables, $\mathbf{q}_\mathrm{d} = [q_1, q_2, ...q_n]$. Next, the user is prompted to highlight the actuated variables in a vector $\mathbf{q}_a$, drawn from a subset of $\mathbf{q}_\mathrm{d}$ unique to the mechanism of interest. The framework then further requires the symbolic kinematic relations following the application of virtual separation as illustrated in Fig. 2. The n_c constraints must be written into a vector of $2n_c \times 1$, as demonstrated with (1) in order to illustrate the constraint with respect to planar environment.

If the mechanism of interest is a floating-base system and an accompanying simulation is required, the switching conditions must be additionally provided to the framework. These conditions must be written in terms of the supplied state variables. Currently, a maximum of two states is supported in the hybrid dynamic model, however expansion to n-states is underway. For fixed-base systems, only one continuous state is present, thus no switching conditions are required

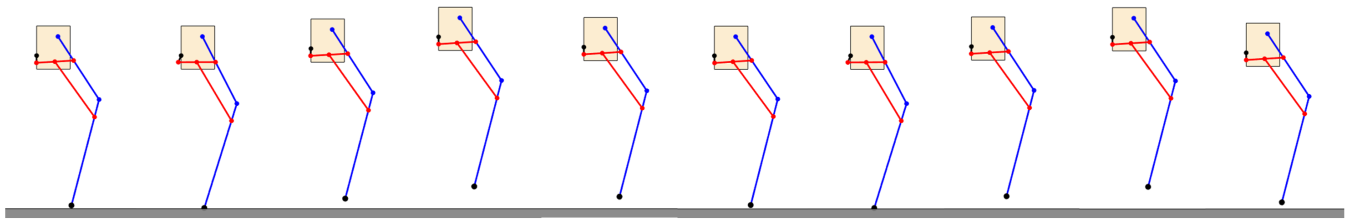


Fig. 5. Monopedal dynamic gait sequence simulated using BOLT, The driver link is shown in black while the driving loop is in red.

and the package automatically models the system within the hybrid domain as a single state.

IV. Case Studies

To provide illustrative examples of the dynamic simulation capabilities of the framework, simulations have been conducted for two distinct legged CKC mechanisms. The first is the author's BOLT, an evolution of the mechanism presented in [9]. The second example is the KAIST Raptor. In both case studies, the mechanisms are simulated performing a monopedal running gait, demonstrating the framework's capacity to simulate hybrid dynamic systems and characterize highly unstable mechanisms. Simulation of a simple static gait, while not pictured, can be achieved through application of the corresponding option within the framework.

A. BOLT

The single-degree of freedom leg, BOLT, is designed with the goal of simplifying legged locomotion while maintaining the full articulation found in biological legs, shown in Fig. 1(a). Incorporation of CKCs within the mechanical design is the most effective approach to achieve the necessary rigidity and weight distribution. As such, BOLT is characterized by a CKC consisting of two closed loops. As highlighted previously, such a mechanism is challenging to simulate numerically with conventional tools and methods.

Utilizing the presented framework, an accurate and drift-free numerical simulation was obtained. By enabling the simulation option, the visualization found in Fig. 5 was generated. The figure shows the leg transitioning between flight and stance phase, and maintains a periodic gait for the duration of the simulation. In demonstrating such behavior, the ability of the presented package to model multi-loop CKCs in hybrid dynamic environments is validated. All impacts are assumed to be rigid and instantaneous, as current framework does not yet contain a compliant contact model.

In Fig, 6, the prescribed asymptotic behavior of the constraint error is shown to converge to zero, indicating the method's validity for drift-free real time model-based control. As the motivation of this case study is to show the functionality of the presented framework to generate a dynamic simulation of a CKC mechanism-based leg, only the angle of attack is controlled with a proportional-derivative controller while the pitch angle is held constant. Due to the simplicity of the controller, the crank angle of the driver link can be seen to oscillate within a narrow band, rather than the more energetically efficient full rotation for which BOLT is designed.

B. KAIST Raptor

To illustrate the versatility of the framework, the next case study is conducted on an established CKC mechanism-based leg implemented on the KAIST Raptor. The leg as designed consists of a three-loop, single DOF mechanism that is integrated into a bipedal robot. The corresponding leg lengths and masses necessary for simulation are obtained from [4]. The mass properties required were approximated through application of the material attributes to flat link geometries, and the constraint equations were determined through the application of virtual separation.

The Raptor leg is simulated under the similar conditions to BOLT in the previous subsection. The results can be seen in Fig. 7. Whereas BOLT exhibited a monopedal forward running gait, the Raptor was simulated to perform a gait more reminiscent of hopping to demonstrate the package's ability to perform more varied gaits in simulation. Such flexibility serves to make the framework a stronger tool when utilized for exploring various controllers and architectures.

While the Raptor leg is more complex than BOLT due to its inclusion of an additional kinematic loop, the presented framework was able to accurately model the mechanism with asymptotically zero drift in the constraints, as seen in Fig. 8. Throughout, ode45 solver is utilized for numerical integration with a time step of 0.002s. The presented methods are capable of modeling CKCs with any number of loops, provided correct relations are supplied.

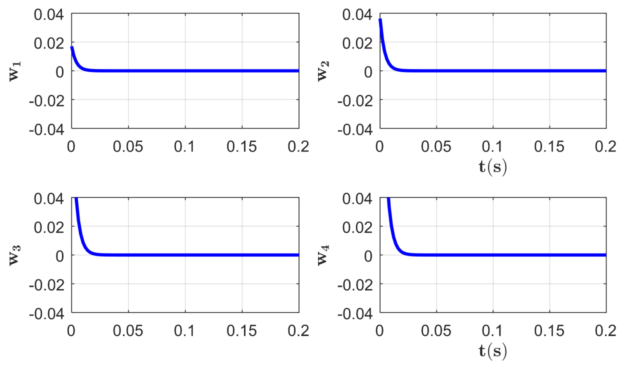


Fig. 6. Constraint errors for the BOLT simulation. Each kinematic loop has a corresponding constraint equation represented in both x and y.

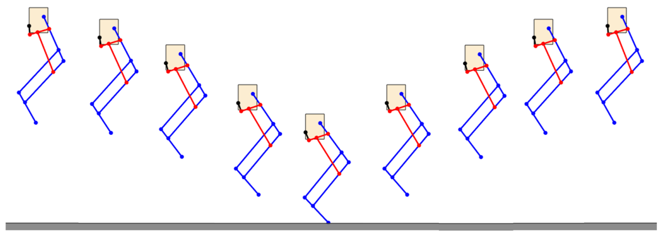


Fig. 7. Monopedal dynamic gait simulation using KAIST Raptor.

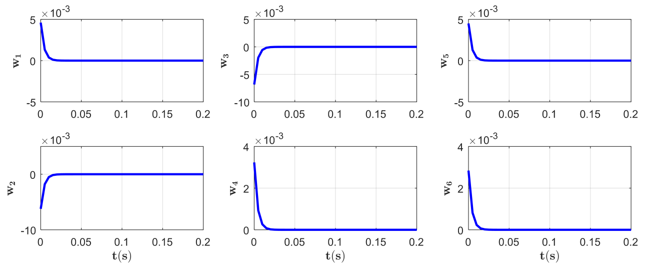


Fig. 8. Constraint errors for the KAIST Raptor simulation. In this case as the Raptor has three loops, there are 6 corresponding constraint equations.

V. CONCLUSIONS AND FUTURE WORK

The framework presented in this paper is capable of modeling, simulating, and providing functions for external implementation of kinematic chains. Special attention is devoted to the application of the package to CKCs incorporated in legged robotics, as the field of legged locomotion heavily relies on model-based control and thus requires accurate, lightweight dynamic modeling in order to develop controllers capable of real-time implementation. The use of this package to create such models can aid researchers in many areas to explore new applications and controls for complex mechanisms.

As a live tool for the modeling of kinematic chains, the presented framework will undergo continuous development. Further improvements will coincide with BOLT's development and integration, including addressing the necessary modeling needs required to support the implementation of highly-underactuated quadrupedal robots in the field. In the near future, the input modalities will be expanded to facilitate the interpretation of user-friendly file formats. In addition, integration with an off-the-shelf visualization package such as Gazebo will be explored. Further, work will be undertaken to expand the framework to 3D cases and to add open kinematic chain dependencies. The incorporation of these further aspects are aimed at the realization of a final product that will be utilized as a unified modeling framework for fixed and floating base kinematic chain dynamics.

ACKNOWLEDGMENT

The material presented in this paper was drawn from work supported through the National Science Foundation under Grant No. 1557312.

REFERENCES

[1] T. Huang, Z. Li, M. Li, D. G. Chetwynd, and C. M. Gosselin, "Conceptual Design and Dimensional Synthesis of a Novel 2-DOF Translational Parallel Robot for Pick-and-Place Operations," *Journal of Mechanical Design*, vol. 126, no. 3, p. 449, 2004.

[2] W. Saab and P. Ben-Tzvi, "Design and Analysis of a Robotic Modular Leg Mechanism," in *Proceedings of ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. ASME, aug 2016, p. V05AT07A062.

[3] E. Refour, B. Sebastian, and P. Ben-Tzvi, "Two-Digit Robotic Exoskeleton Glove Mechanism: Design and Integration," *Journal of Mechanisms and Robotics*, vol. 10, no. 2, p. 025002, jan 2018.

[4] J. Park, K.-S. Kim, and S. Kim, "Design of a cat-inspired robotic leg for fast running," *Advanced Robotics*, vol. 28, no. 23, pp. 1587–1598, 2014. [Online]. Available: http://dx.doi.org/10.1080/01691864.2014.968617

[5] G. Kenneally, A. De, and D. E. Koditschek, "Design Principles for a Family of Direct-Drive Legged Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, 2016.

[6] W. Bosworth, S. Kim, and N. Hogan, "The MIT super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion," in *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, oct 2015, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7443018/

[7] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2017, pp. 3345–3351. [Online]. Available: http://ieeexplore.ieee.org/document/8206172/

[8] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, and J. Hurst, "ATRIAS: Design and validation of a tether-free 3D-capable spring-mass bipedal robot," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1497–1521, oct 2016. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364916648388

[9] V. R. Kamidi, W. Saab, and P. Ben-Tzvi, "Design and analysis of a novel planar robotic leg for high-speed locomotion," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada: IEEE, sep 2017, pp. 6343–6348. [Online]. Available: http://ieeexplore.ieee.org/document/8206540/

[10] W. Saab, W. Rone, and P. Ben-Tzvi, "Robotic Modular Leg: Design, Analysis and Experimentation," *Journal of Mechanisms and Robotics*, no. c, 2017. [Online]. Available: http://mechanismsrobotics.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4035685

[11] Zhiyong Wang and F. Ghorbel, "Control of closed kinematic chains: a comparative study," in *2006 American Control Conference*. Minneapolis, Minnesota, USA: IEEE, 2006, p. 6 pp. [Online]. Available: http://ieeexplore.ieee.org/document/1656597/

[12] M. Hutter, C. Gehring, and R. Siegwart, "proNEu: Derivation of analytical kinematics and dynamics," ETH Zurich, Tech. Rep., 2011.

[13] R. Tedrake and Drake Development Team, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016. [Online]. Available: http://drake.mit.edu

[14] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2017, pp. 719–726. [Online]. Available: http://ieeexplore.ieee.org/document/8202230/

[15] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, jun 1972. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0045782572900187

[16] B. W. Gordon and S. Liu, "A Singular Perturbation Approach for Modeling Differential-Algebraic Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 120, no. 4, p. 541, 1998. [Online]. Available: http://dynamicsystems.asmedigitalcollection.asme.org/article.aspx?articleid=1407967

[17] J. Wittenburg, *Dynamics of Multibody Systems*, 2nd ed. Springer.

[18] J. W. Grizzle, C. Chevallereau, A. D. Ames, and R. W. Sinnet, "3D Bipedal Robotic Walking: Models, Feedback Control, and open problems," *IFAC Symposium on Nonlinear Control Systems*, pp. 505–532, 2010.

[19] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, and E. al., *Feedback Control of Dynamic Bipedal Robot Locomotion*, 2007.