Proceedings of the ASME 2018
Dynamic Systems and Control Conference
DSCC2018
September 30-October 3, 2018, Atlanta, Georgia, USA

DSCC2018-9000

A MULTIBODY TOOLBOX FOR HYBRID DYNAMIC SYSTEM MODELING BASED ON NONHOLONOMIC SYMBOLIC FORMALISM

Jiamin Wang
Robotics and Mechatronics Lab
Department of Mechanical Engineering.
Virginia Tech
Blacksburg, VA, USA

Vinay R Kamidi
Robotics and Mechatronics Lab
Department of Mechanical Engineering.
Virginia Tech
Blacksburg, VA, USA

Pinhas Ben-Tzvi
Robotics and Mechatronics Lab
Department of Mechanical Engineering.
Virginia Tech
Blacksburg, VA, USA
bentzvi@vt.edu

ABSTRACT

This paper proposes a hybrid multibody dynamics formalism with a symbolical multibody toolbox developed in MATLAB Environment. The toolbox can generate the dynamic model of a multibody system with hybrid and nonholonomic dynamic properties. The framework and software structure of the toolbox are briefly demonstrated. The paper discusses the recursive kinematics and modular modeling theories that help improve the modeling performance and offer accessibility into the dynamic elements. The formalism that offers a symbolic solution to nonholonomic and constrained dynamics is explained in detail. The toolbox also provides design tools and auto-compilation of hybrid automata. Two exemplary models and their simulations are presented to verify the feasibility of the formalism and demonstrate the performance of the toolbox.

NOMENCLATURE

EOM – Equation of Motion
DOF – Degrees of Freedom
COM – Center of Mass
ODE – Ordinary Differential Equation
$R_{(Y)}^{(X)}$ – Rotation Matrix (3×3) from Frame Y to X
$r_{(Y)}^{(X)}$ – Displacement Vector (3×1) from Frame Y to X
$v^{(X)}$ – Translational Velocity (3×1) in Frame X
$\mathrm{a}^{(X)}$ – Translational Acceleration (3×1) in Frame X
$\omega^{(X)}$ – Angular Velocity (3×1) in Frame X
$\alpha^{(X)}$ – Angular Acceleration (3×1) in Frame X
$\tilde{e}^{(X)}$ – Skew Matrix of the Vector $e^{(X)}$
f – System ODE (ODE of Flow)
g – Guard and Reset Map (Jump)
M – Inertial Matrix
G – Generalized Force
Λ – Virtual Power
Γ – Constraint Vector
q – Generalized Coordinates (Continuous States)
p – Discrete States
u – System Input
σ – Nonholonomic Signal
λ – Lagrangian Multiplier
m – Mass (1×1)
$I^{(X)}$ – Moment of Inertia (3×3) in Frame X
$F^{(X)}$ – Force represented (3×1) in Frame X
$T^{(X)}$ – Torque represented (3×1) in Frame X
Φ – The World Coordinate Frame

1. INTRODUCTION

Standard process-orientated calculation of a system with multiple bodies with dynamic effects, constraints, and switched/hybrid dynamic properties often yields a laborious and complicated analysis problem. With the advancement of computer technology, multibody dynamic software toolboxes are developed to provide assistance to researchers in multibody dynamics with objective-oriented programming approaches.

The multibody dynamic formalism determines the software structure of a modeling toolbox. While there are different formalisms developed by researchers over years, the principles of these multibody formalisms are based on a combination of the Lagrange's equations, Newton-Euler principle and d'Alembert's or Jourdain's principles also denoted as Kane's method [1]. Apart from formalisms, algorithms may vary based on different programming languages and design method.

1.1. Background and Related Work

The majority of the existing toolboxes adopt numerical formalisms. Over the past three decades, numerical algorithms

for physical/dynamic effects are designed and optimized to provide reliable simulation solution. Apart from the physics engines designed for multimedia entertainments [2], many scientific and research numerical multibody software toolboxes are developed for different modeling purposes such as ADAMS [3] for rigid mechanics and Simbody [4] for biomedical research modeling.

While modeling based on numerical methods are efficient in visualization and real-time interactivities, analytical solution and symbolic expression of dynamic elements are more helpful in the detailed study of a multibody system. The development of symbolic calculation engine [5] has led to the emergence of symbolic modeling toolboxes, which includes the ones designed by researchers led by A. Kecskemethy [6]; E. Kreuzer [7]; T. Kurz [8]; and B. Bitterner [9] respectively.

Switched and hybrid systems [10] have been a popular topic of study in many fields of research such as electronics, fluid dynamics, and robotics. The analysis of hybrid dynamics is crucial to the design and control of the system. However, the development of a hybrid dynamic modeling formalism has been a challenge due to the introduction of discrete properties. Therefore, not all of the formalisms support modeling of hybrid dynamics. In recent years, there have been several toolboxes designed with hybrid system features, such as Drake [11] and FROST [12].

1.2. Motivation

Most symbolic multibody toolboxes cannot support the modeling of nonholonomic properties, such as non-sliding rolling motion and floating joint. The expression of nonholonomic properties follows the form $\dot{\sigma} = f_\sigma(x)$. Here, x are the variables that describes the derivative of signal σ, while the expression of σ cannot be acquired through integration, which has result in the difficulty of analytically describing the dynamics of a system.

Some symbolic modeling formalisms try to circumvent the problem. However, the vector spaces of the circumvention and of the original properties may not be homomorphic, such as the 'Gimbal Lock' problem experienced using Euler Angle for rotation parameters.

As researchers in mechatronics and robotics, we often face tasks in dynamic related design optimization and controller design for novel designs, which requires symbolic modeling of nonholonomic, constrained and hybrid dynamics. While the previously introduced toolboxes are powerful, they do not offer solutions to all of the features. This motivated us to develop an efficient and capable symbolic toolbox which can provide solutions to these problems.

In this paper, we present a multibody formalism which offers a solution to most of the problems that may occur in rigid multibody modeling as mentioned above. The formalism is presented in the form of an objective-oriented symbolical multibody research software named "ANDY". The toolbox is developed based on the MATLAB environment, which the symbolic mathematics toolbox and other data structure classes. The features of the ANDY include:

(1) Application of graph theory for kinematic chain hierarchy and hybrid dynamic topology to help the user with their modeling process.
(2) Adoption of frame transformation based recursive kinematic algorithms to improve modeling performance.
(3) Applying formality to system elements to support modular dynamic modeling, which provides insight into the dynamic properties of system elements.
(4) Capability of modeling system with constrained and nonholonomic properties.
(5) Compilation and generation of hybrid automata codes for simulation purposes which supports reset map and Zeno/multi-jump detection.
(6) Providing tools for visualization and 3D animation of the multibody system.

The development and theory behind the toolbox will be discussed in detail. Several experimental verifications will be carried out to verify the performance of the toolbox.

2. FRAMEWORK AND ALGORITHM

The software framework of ANDY is presented in Fig. 1. For ANDY, the `System` is the base object for the whole multibody dynamic model. Before the establishment of the kinematics or dynamics, the system variables need to be declared. There are three basic categories of variables:

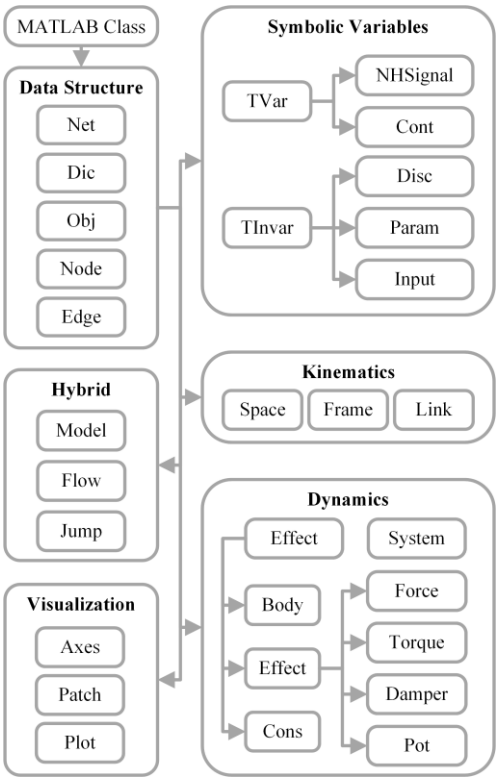

Figure 1. Software framework of ANDY

(1) Constants (`Param`): System internal parameters such as constant dimensions and inertial properties.
(2) Time Independents Signal (`Input`, `Disc`): These variables that is independent of time or external to the system, which includes discrete state, input signals, noises, etc.
(3) Time-Dependent Signal (`NHSignal`, `Cont`): Time-dependent variables in the system which are nonholonomic and calculated through numerical methods. Time-dependent signals are defined with three properties – the differential order, initial condition, and ODE.

Coordinate `Frame` and the transformation `Link` are established in `Space` graph network. While the derivation hierarchy of coordinate frames has the tree topological structure, basic graph theories can be used to retrieve the path of links between coordinate frames, which determines the relationship of transformations.

Basic system elements include `Body`, `Force`, `Torque`, `Damper` and Potential Energy (`Pot`). Some libraries adopt the notion to describe relationships between bodies with joints, which are inherited on and derived from bodies. Unlike those libraries, according to ANDY's framework, all the system elements are established based on the coordinate frames. This provides the accessibility to the dynamics of each element individually and means to generate dynamic equations in a modular way. The unconstrained system dynamics can then be generated with these elements.

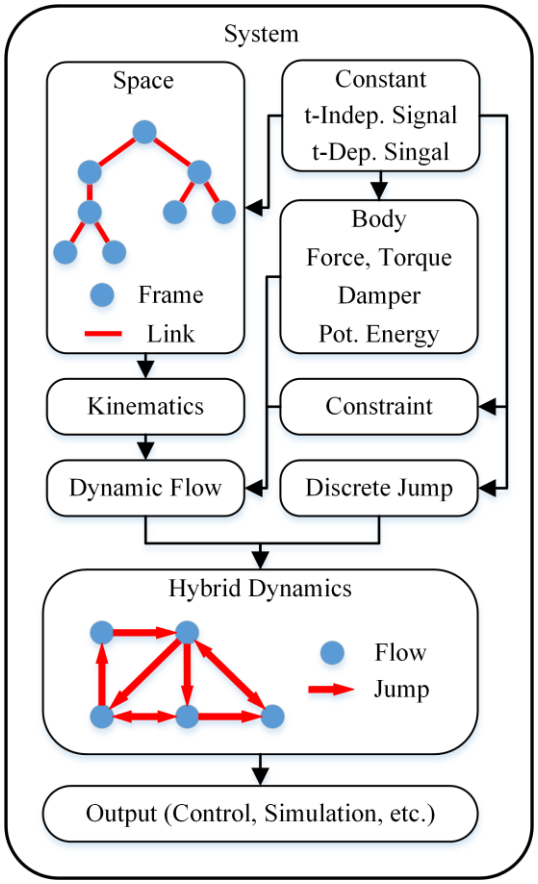

Figure 2. The process of multibody formalism

Constraints (`Cons`) are separated from the basic system elements since they will introduce loss of DOF to a system based on an open kinematic chain. Since the constraint effect in a model has also been modularized, the constrained EOM can be generated simply based on the unconstrained EOM and the selected constraints.

The hybrid dynamic `Model` is another structure based on graph theory, which contains `Flow` as its nodes and `Jumps` as its edges. For each flow, there is a set of EOM of the system. `Jump` contains the guard and resets map between flows.

Finally, visualization tool `Axes` provides `Patch` and `Plot` for 3D model animation and animated 3D trajectory printing.

The formalism process of ANDY based on the previously discussed framework is depicted in Fig. 2, which describes the procedure it takes to model a system. The algorithms will be discussed in detail in the following sections.

2.1. Recursive and Nonholonomic Kinematics

The hierarchy of coordinate frames follows the principle of tree topology. In ANDY, the code design extended the existing graph toolbox utilities to construct the space of a system. Each space has a static root coordinate frame that sets the reference for the whole system space. Kinematic links describe the relationships between coordinate frames.

The theory of recursive kinematics has been well developed, which can be applied for varies modeling problems [13][14]. Assume the links between coordinate frame Ψ_1 and coordinate frame Ψ_n are presented as $\Psi_1 \to \Psi_2 \to \cdots \to \Psi_{i-2} \to \Psi_{i-1} \to \Psi_i \to \cdots \to \Psi_n$. For each link, the known information includes the relative displacements, velocities, and accelerations of the child frame in their parent frame: $r_i^{(i-1)}, v_i^{(i-1)}, a_i^{(i-1)}, R_i^{(i-1)}, \omega_i^{(i-1)}, \alpha_i^{(i-1)}$. Assume the kinematic properties of Ψ_i presented in Ψ_{h+1} are acquired as $r_i^{(h+1)}$, $v_i^{(h+1)}$, $\omega_i^{(h+1)}$, $a_i^{(h+1)}$ and $\alpha_i^{(h+1)}$, the presentation of these properties in Ψ_h can be calculated as:

$$
\begin{aligned}
r_i^{(h)} &= R_{(h+1)}^{(h)} r_i^{(h+1)} + r_{(h+1)}^{(h)} \\
v_i^{(h)} &= R_{(h+1)}^{(h)} v_i^{(h+1)} + v_{h+1}^{(h)} + \tilde{\omega}_{h+1}^{(h)} R_{(h+1)}^{(h)} r_i^{(h+1)} \\
\omega_i^{(h)} &= R_{(h+1)}^{(h)} \omega_i^{(h+1)} + \omega_{h+1}^{(h)} \\
a_i^{(h)} &= R_{(h+1)}^{(h)} a_i^{(h+1)} + a_{h+1}^{(h)} + \tilde{\omega}_{h+1}^{(h)} [\omega_{h+1}^{(h)} (R_{h+1}^{(h)} r_i^{(h+1)})] + \\
&\quad 2\tilde{\omega}_{h+1}^{(h)} (R_{h+1}^{(h)} v_i^{(h+1)}) + \tilde{\alpha}_{h+1}^{(h)} (R_{(h+1)}^{(h)} r_i^{(h+1)}) \\
\alpha^{(h)} &= R_{(h+1)}^{(h)} \alpha_i^{(h+1)} + \tilde{\omega}_{(h+1)}^{(h)} R_{(h+1)}^{(h)} \omega_i^{(h+1)} + \alpha_{h+1}^{(h)}
\end{aligned}
\tag{1}
$$

The process in (1) outlines the symbolical recursive kinematics algorithm in ANDY. We can also acquire the Jacobian matrixes of the velocity properties of the coordinate frames with a similar procedure:

$$
J_{v_i}^{(h)} = R_{(h+1)}^{(h)} J_{v_i}^{(h+1)} + J_{v_{h+1}}^{(h)} - (R_{(h+1)}^{(h)} \tilde{r}_i^{(h+1)}) J_{\omega_{h+1}}^{(h)} \tag{2}
$$

$$
J_{\omega_i}^{(h)} = R_{(h+1)}^{(h)} J_{\omega_i}^{(h+1)} + J_{\omega_{h+1}}^{(h)} \tag{3}
$$

The recursive algorithm has a higher efficiency compared with symbolical differentiation of a complicated symbolic

expression since the velocities and accelerations of different frames can be acquired separated beforehand. Coordinate transformation can be nonholonomic. A nonholonomic kinematic link can be described as $r(0) = r_0$, $R(0) = R_0$, $\dot{r}=v$ and $\dot{R} = \tilde{\omega}R$. Here, r_0 and R_0 are the initial condition of the translational displacement and rotation matrix. The analytical solution of r and R are not available. In ANDY, the solution is offered by introducing nonholonomic signals. Whenever nonholonomic kinematic relationships are declared, the system will automatically define the corresponding nonholonomic signals that satisfy:

$$\begin{aligned} r &= \sigma_{\mathrm{r}} \\ R &= \sigma_{\mathrm{R}} \\ \dot{\sigma}_{\mathrm{r}} &= v \\ \dot{\sigma}_{\mathrm{R}} &= \tilde{\omega}R \end{aligned} \tag{4}$$

Nonholonomic signals will introduce new dynamics, which will be distinguished and treated separately. Compared to the joint based kinematics used in other symbolic toolboxes, which can only describe holonomic transformations, frame-based kinematics offered solutions to possible modeling problems.

2.2. Modular Dynamic Modeling

In our formalism, system elements are established on coordinate frames. The system objects are inherent to their kinematics, which leads to the convenient implementation of Kane's Method [15]. The Equation of Motion of a multibody system can normally be presented in the form of $M\ddot{q} = G$. Based on such presentation, the elements in a system with open kinematic chains are divided into two categories – Inertial Elements and Generalized Force Elements.

Particles and bodies are categorized into inertial elements. Providing that a body P_k has a mass of m_k and an angular inertia of I_k. The COM of the body is located at the origin of frame Ψ_k. The inertial forces and torques are presented as $F_k^{(\Phi)} = -m_k a_k^{(\Phi)}$ and $T_k^{(\Phi)} = -I_k^{(\Phi)}\alpha_k^{(\Phi)} - \tilde{\omega}_k^{(\Phi)} I_k^{(\Phi)} \omega_k^{(\Phi)}$.

According to Kane's Method, we can calculate the virtual power generated by the inertial force is given as $\Lambda_{\mathrm{k}}=v_{\mathrm{k}}^{(\Phi)^{\mathrm{T}}} F_{\mathrm{k}}^{(\Phi)}+\omega_{\mathrm{k}}^{(\Phi)^{\mathrm{T}}} T_{\mathrm{k}}^{(\Phi)}$, which can also be expressed in term as $\Lambda_{\mathrm{k}}= \dot{q}^{\mathrm{T}}(J_{v\mathrm{k}}^{\mathrm{T}} F_{\mathrm{k}}^{(\Phi)}+J_{\omega\mathrm{k}}^{\mathrm{T}} T_{\mathrm{k}}^{(\Phi)})$. Here, $J_{v_{\mathrm{k}}}^{\mathrm{T}}$ and $J_{\omega_{\mathrm{k}}}^{\mathrm{T}}$ are the Jacobian Matrices that map $\dot{q}$ to $v_{\mathrm{k}}^{(\Phi)}$ and $\omega_{\mathrm{k}}^{(\Phi)}$, respectively. Therefore, the inertial matrix and generalized force introduced by the body to the system are calculated as:

$$M_K = J_{v\mathrm{k}}^{\mathrm{T}} m_{\mathrm{k}} J_{v\mathrm{k}} + J_{\omega\mathrm{k}}^{\mathrm{T}} I_{\mathrm{k}}^{(\Phi)} J_{\omega\mathrm{k}} \tag{5}$$

$$G_{\mathrm{Ik}} = -J_{v\mathrm{k}}^{\mathrm{T}} m_{\mathrm{k}} (\dot{J}_{v\mathrm{k}} \dot{q}) - J_{\omega\mathrm{k}}^{\mathrm{T}} (I_{\mathrm{k}}^{(\Phi)} \dot{J}_{\omega\mathrm{k}} \dot{q} + \tilde{\omega}_{\mathrm{k}}^{(\Phi)} I_{\mathrm{k}}^{(\Phi)} \omega_{\mathrm{k}}^{(\Phi)}) \tag{6}$$

The Coriolis and Centripetal forces of the inertial elements will be introduced to generalized force to the system. The virtual power of forces, torques, dampers, potential energies and other effects can all be converted into the following standard form $\Lambda=\upsilon^{\mathrm{T}} K\zeta$. We define υ as the directional vector (a 1st order time derivative vector), K as the parameter matrix and ζ as the magnitude vector. The assignment of the variable is flexible as long as the unit of $\upsilon^{\mathrm{T}} K\zeta$ is equivalent to Watt.

Table 1. Generalized Force Elements Examples

Type	υ	K	ζ
Force	v	I	F
Torque	ω	I	T
Linear Damper	χ	$-b$	χ
Quadratic Damper	χ	$-b$	$\mathrm{sign}(\chi)\chi^2$
Spring	$\dot{x}$	$-k$	x
Gravity	v	mI	a_{g}

Table. 1 presents some typical generalized force elements and how they fit into the standard form. Here, χ is the velocity constraint for damper and x is the position constraint for the spring. Once the form is established, the generalized force can be calculated by:

$$G_{\mathrm{F}} = J_{\mathrm{v}}^{\mathrm{T}} K\zeta \tag{7}$$

$J_{\upsilon}^{\mathrm{T}}$ is the Jacobian Matrix that maps $\dot{q}$ to υ. The generalized force of the elements in the same system will therefore possess the same dimension.

Based on the previous conclusions (7)-(9), the EOM of an unconstrained system with m inertial elements and n generalized force elements can be presented as:

$$(\sum_{\mathrm{i}=1}^{\mathrm{m}} M_{\mathrm{i}})\ddot{q} = \sum_{\mathrm{i}=1}^{\mathrm{m}} G_{\mathrm{I_i}} + \sum_{\mathrm{j}=1}^{\mathrm{n}} G_{\mathrm{F_j}} \tag{8}$$

If the modeling process is reasonable and M matrix has a full rank, indicating that there are no redundant generalized coordinates, the ODE of the system model can be generated by:

$$\ddot{q} := f(q,\dot{q}) = M^{-1}G + M^{-1}J_{\Gamma}^{\mathrm{T}}\lambda \tag{9}$$

Through modular dynamic modeling, the complicated dynamic model can be segmented, which may facilitate the parallel computing of the modeling and simulation process. The dynamic properties of each element can be inspected individually for study purpose or modeling inspection.

2.3. Constrained Dynamics

Constraints will introduce new dynamic characteristics and complexity to the base system. The nonholonomic constraints have the standard form $\dot{\Gamma} = 0$. If the integral of a nonholonomic constraint can be obtained analytically, the constraint becomes holonomic $\Gamma = 0$. There are various methods available to integrate constraints into an unconstrained system. The constraints that have an explicit analytical solution can be solved and substitute into the system directly. However, the loss of DOF will lead to the reduction of system states, making it inconsistent with the unconstrained base model. A commonly adopted

method to apply constraint is through Lagrangian Multiplier. The EOM of the constrained system is given by

$$M\ddot{q} = G + J_\Gamma^{\mathrm{T}}\lambda \tag{10}$$

The Lagrangian Multipliers λ are the constraint forces. The Jacobian Matrix J_Γ is the mapping from $\dot{q}$ to the constraint vector $\dot{\Gamma}$, which satisfies:

$$\dot{\Gamma} = J_\Gamma\dot{q}; \ddot{\Gamma} = J_\Gamma\ddot{q} + \dot{J}_\Gamma\dot{q} \tag{11}$$

Therefore, by substituting $\ddot{q}$ from (11) into (12) yields:

$$\lambda = (J_\Gamma M^{-1} J_\Gamma^{\mathrm{T}})(-J_\Gamma M^{-1} G - \dot{J}_\Gamma\dot{q}) \tag{12}$$

The effect of constraint force will lead to loss of DOF without affecting the consistency between models. The constraint forces can also monitor unilateral constraints [16]. However, numerical error accumulated during the numerical integration may result in the constraint drifting apart. Another possibility to apply the constraint to a system is through the "Soft Constraint" method. Soft constraints are applied to the system as generalized force elements such as springs and dampers. When the parameters are selected properly, the effect of these springs and dampers will be very similar to the rigid constraints.

For a holonomic constraint, the virtual power of the generalized force elements added to the system as in (17).

$$\Lambda = -\dot{\Gamma}^{\mathrm{T}} K_s\Gamma - \dot{\Gamma}^{\mathrm{T}} K_d\dot{\Gamma} \tag{13}$$

In this case, $K_s\Gamma$ is similar to spring force and $K_d\dot{\Gamma}$ is similar to damper force. For a nonholonomic constraint, K_s is equal to zero. The generalized force is then given by

$$G_\Gamma = -J_\Gamma^{\mathrm{T}}(K_s\Gamma + K_d\dot{\Gamma}) \tag{14}$$

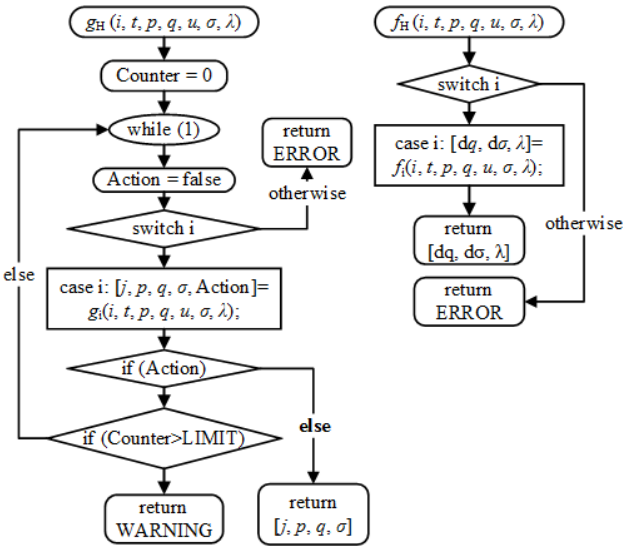

Figure 3. Flow charts of f_H and g_H

Compared with the Lagrangian Multiplier method, the calculation of the generalized force of soft constraints does not require inverse calculation of the inertia matrix, which to some extent simplified the modeling process. Soft constraints can also avoid the constrained dynamics from drifting due to error accumulation.

Soft constraints can mimic the effect of rigid constraints, but they are not rigorous since they change the dynamics of the original system. Another disadvantage becomes prominent when parameters have very large norms, which leads to the increase in the natural frequency of the system since the sampling frequency of a simulation needs to be higher than the highest natural frequency of the system to provide a reliable result. In ANDY, the Lagrangian Multiplier is used as the main constraint method with the auxiliary soft constraints [17]:

$$G_\Gamma = J_\Gamma^{\mathrm{T}}\lambda - J_\Gamma^{\mathrm{T}}(K_s\Gamma + K_d\dot{\Gamma}) \tag{15}$$

The combination of the two methods will prevent the constraints from drifting in a long run. The parameters of the soft constraint forces can be chosen relatively smaller which prevents the high system frequency.

2.4. Hybrid Automata

Previous sections explained the formulation of EOM of single system flows. The result can be used for controller design or exported to other applications. In addition to that, ANDY also supports the design and compilation of hybrid automata.

Switched systems and hybrid systems consist of multiple continuous and discrete system states. The flows and the jumps are presented in the form below [18] $[\dot{q}, \dot{\sigma}, \lambda] = f_i(t, p, q, u, \sigma, \lambda)$ and $[j, p, q, \sigma] = g_{ij}(t, p, q, u, \sigma, \lambda)$. Here f_i is the continuous dynamics of flow i; g_{ij} is the jump from flow i to flow j, which contains the guard and reset map. While flows can be compiled and generated automatically based on the algorithms discussed in the previous sections, the jumps still need to be manually designed. ANDY offers the assistive tool of hybrid system building with graph theory. The jump functions can be coded by modifying the generated templates. Once all settings are complete, ANDY will compile the hybrid automata into two state machines:

$$[j, p, q, \sigma] = g_H(i, t, p, q, u, \sigma, \lambda) \tag{16}$$

$$[\dot{q}, \dot{\sigma}, \lambda] = f_H(j, t, p, q, u, \sigma, \lambda) \tag{17}$$

Here i is the initialflow of the system; provides the ODE of the time dependent variablesin flow j; g_H provides the jump between i and j. The flow charts for g_H and f_H are presented in Fig. 3. A counter mechanism in g_H will detect multiple jumping scenarios. For cases such as zeno and infinite jump loop, the function will stop after the multiple jumping limits are reached and return the warning and the jumping trajectory data.

2.5. Other Toolbox Features

ANDY can generate precompiled models with MATLAB code generator. The precompiled codes have high efficiency and can be used for real-time simulation and implementation.

The precompiled code is compatible with all MATLAB ODE solvers for offline simulations. For real-time simulation, `rk4Hybrid` – a solver based on Runge-Kutta 4th is provided to allow input into the system.

3. IMPLEMENTATION AND CASE STUDY

The process of building a model with ANDY includes the following steps:

(1) Define the time variable and system object, declare the time-continuous states and any input variables, discrete states and nonholonomic signals to be used in the modeling.
(2) Construct the space and coordinate frame hierarchy. Define holonomic or nonholonomic kinematic links between coordinate frames according to model setup.
(3) Establish bodies and other system elements on coordinate frames. The inertia properties of the body will be defined at the origin of the base frame. The action points of forces and torques will be defined at the origin of the base frame, while the directions of the vectors can be defined in alternate reference frames. Dampers and potential energies do not require coordinate frames.
(4) Declare holonomic or nonholonomic constraints and name the corresponding Lagrangian multipliers.
(5) Initialize the model, build the hybrid system network. Define the constraints for different flows and write the jump function based on the template.
(6) Compile and output. The symbolic dynamics can be obtained from the model for dynamics and control study. The precompiled output function can be used for simulation and implementation.

Two cases are demonstrated to verify the performance of ANDY. The examples will showcase ANDY's capability of solving nonholonomic, constrained and hybrid multibody dynamic modeling problems.

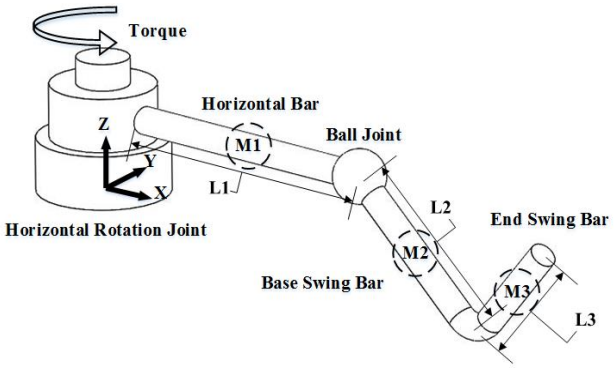

Figure 4. Swing bar mechanism

Table 2. Parameters and Initial Conditions of Swing Bar

Parameter	Value	Parameter	Value
L_1	$1m$	M_1	$1kg$
L_2	$1m$	M_2	$1kg$
L_3	$0.5\ m$	M_3	$0.5kg$
g	$9.81\ m/s^2$	b	$0.2\ Ns/m$
T	$2\ \ N$	$[q_{\mathrm{H}}, \dot{q}_{\mathrm{H}}]$	$[0, -5]$
$[q_{\mathrm{x}}, q_{\mathrm{y}}, q_{\mathrm{z}}]$	$[0, 0, 0]$	S_{j}	$[\sqrt{2}/2, 0, \sqrt{2}/2, 0]$

3.1. The Swing Bar Case

The first example features a simple 3D swing bar mechanisms as shown in Fig. 4. The ball joint between the two first two bars has introduced a nonholonomic kinematic link. A torque T is exerted at the horizontal bar. The joint between the swing bars is a fix joint.

The system, therefore, has four degrees of freedoms. Four continuous states are declared, in which q_h represents the angle of the horizontal bar; $\dot{q}_x, \dot{q}_y, \dot{q}_z$ represent the angular velocities of the ball joint. The quaternion $s_j = [\sigma_w, \sigma_x, \sigma_y, \sigma_z]$ describes the rotation of the ball joint is automatically generated by the setup command of nonholonomic link [19].

The kinematic hierarchy is presented in Fig. 5. Here, $DH\,(d, \theta, a, \alpha)$ stands for standard Denavit–Hartenberg transformation [20]. The three bars are established on the COM Frames. The system also involves dampers that dissipates the kinetic energy based on the global damping factor b.

With the system parameter, initial conditions and simulation parameters chosen from Table. 2, the simulation outcome is shown in Fig. 6. A 3D trajectory in the world frame is plotted in the top subplot. The quaternion vector trajectory of the ball joint rotation is visualized in the second subplot, which indicates that the system enters stability after the instantaneous response has

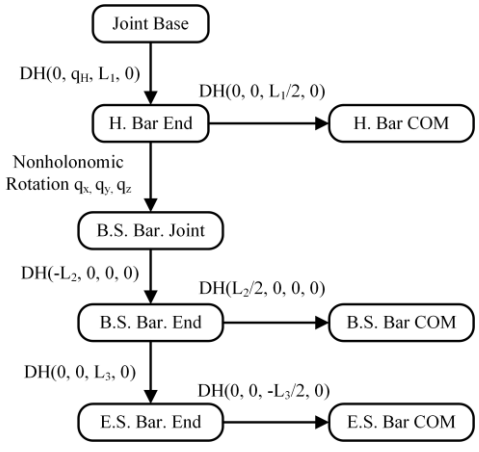

Figure 5. Coordinate frame hierarchy of swing bar system

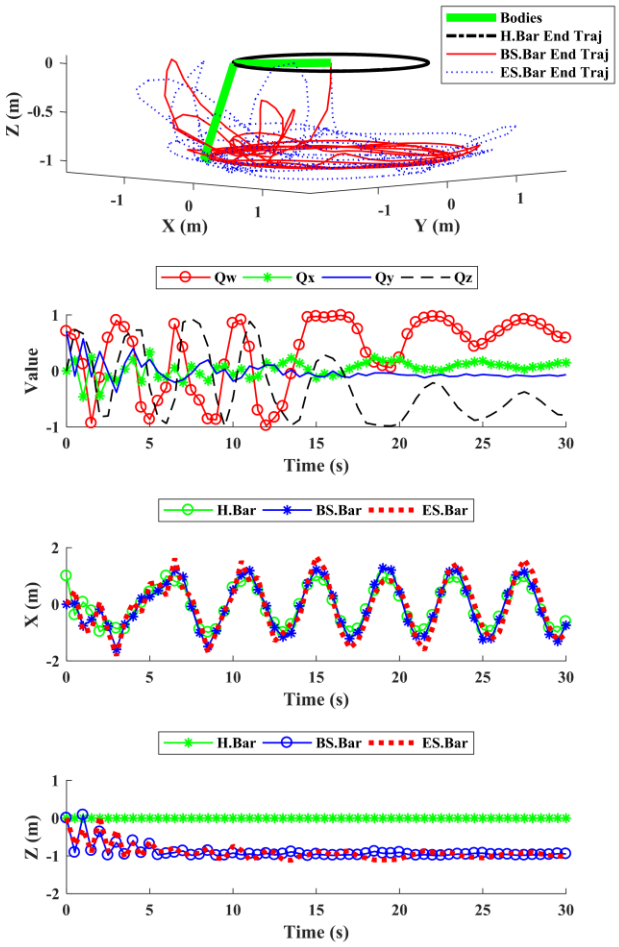

Figure *6*. Swing bar simulation result

been damped out. Further, the X and the Z coordinate trajectory of bar end points shown in the bottom two subplots corroborate with each other, indicating the steady state convergence of all three bars.

The results show that the ANDY can solve nonholonomic problems effectively, which verifies the feasibility of the

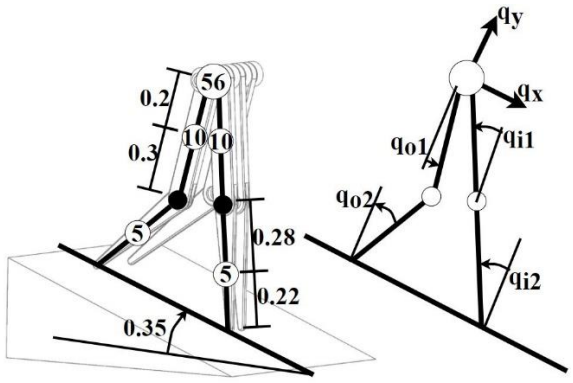

Figure 7. The layout of a passive walker with knee

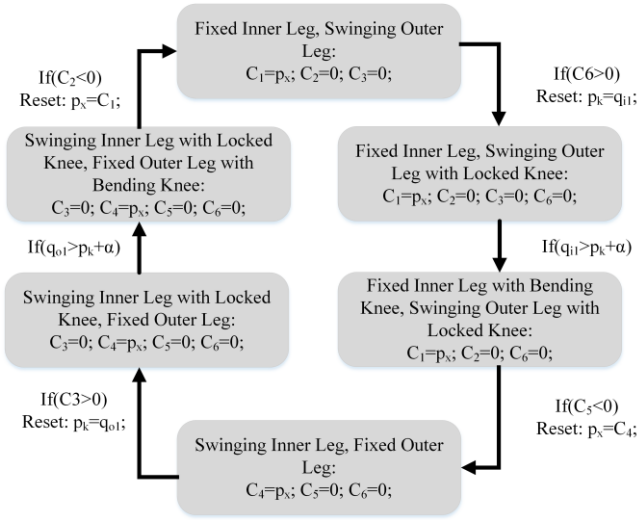

Figure 8. Hybrid transition map of passive walker

nonholonomic formalism. While the introduction of nonholonomic properties adds to the complexity of EOM (resulting in 9 ODEs in total), the `rk4Hybrid` solving frequency of precompiled ODE function for simulation can reach approximately $2.4e4$ HZ (single-threaded with Intel i7-7700 CPU).

3.2. The Passive Walker Case

Passive walker with knees is a classic constrained hybrid multibody model. Since a passive walker's stability is critically related to the design of the system, we referred to the modeling process of the knee bending passive walker presented in the paper by An and Chen [21]. The layout of the system is shown in Fig. 7.

We selected the identical system parameter from the reference paper [21]. However, instead of following the hybrid modeling process that remaps properties between the two legs, we have used a float base model to prevent discontinuity in system states. Therefore, the system has six holonomic constraints which can be expressed by the following expressions:

$$\begin{aligned} C_1 &= q_x + 0.5(\sin(q_{i1}) + \sin(q_{i2})) \\ C_2 &= q_y - 0.5(\cos(q_{i1}) + \cos(q_{i2})) \\ C_3 &= q_{i2} - q_{i1} \\ C_4 &= q_x + 0.5(\sin(q_{o1}) + \sin(q_{o2})) \\ C_5 &= q_y - 0.5(\cos(q_{o1}) + \cos(q_{o2})) \\ C_6 &= q_{o2} - q_{o1} \end{aligned} \tag{18}$$

The hybrid system also has in total 6 different phases, each of the phases possesses a flow with certain constraints. The hybrid transition map is illustrated in Fig. 8.

The discrete states used in the model include the knee locking angle p_k and foot fixing position p_x, which are used

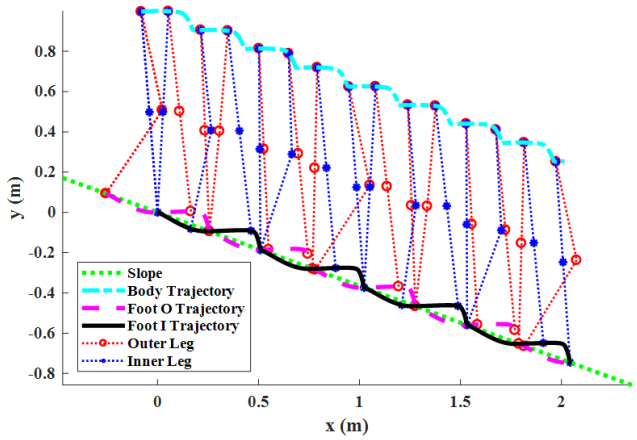

Figure 9. Walking trajectory of passive walker

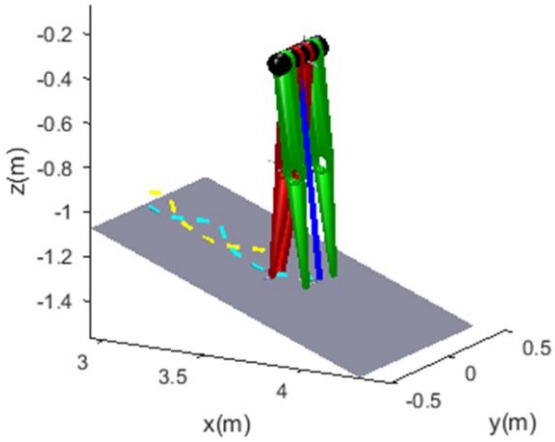

Figure 11. 3D animation of passive walker

for implementing the constraints. The angle α is the knee bending angle introduced to allow the stance leg to bend after the swing leg's knee is locked. The reset maps for fixing feet and locking knees includes impact map and state reset. The impact map [22] is realized with the mapping of continuous states:

$$\dot{q}^{+} = (1 - M^{-1} J_{\Gamma}^{\mathrm{T}} (J_{\Gamma} M^{-1} J_{\Gamma}^{\mathrm{T}}) J_{\Gamma}) \dot{q}^{-} \qquad (19)$$

While the hybrid automata for these systems are complicated. ANDY will auto-compile the flow functions based on your constraint settings at each phase. ANDY will also generate the jump function based on a prepared template, which allows you to fill in the reset maps in addition to the impact map.

By giving the same initial condition used in the reference paper [21], we have achieved the identical system walking performance, which is presented in Fig. 9. Since any small change in a stable passive walker model may result in instability, the result shows that the model established by ANDY is identical to the one modeled in the reference paper.

The angle and jumping trajectory of the system is presented in Fig. 10. While the trajectories look slightly different from the plot in the reference model (where there are discrete jumps) due to the model setup differences, the simulation data is identical.

The frequency of jump function and flow ODE solving, based on the same setup as in the previous example, can be evaluated at the frequency of approximately $1.7e4$ HZ. A snapshot of the 3D animation visualization is presented in Fig. 11. The animation of the 3D model that includes 10 bodies can maintain a framerate suitable for real-time simulation.

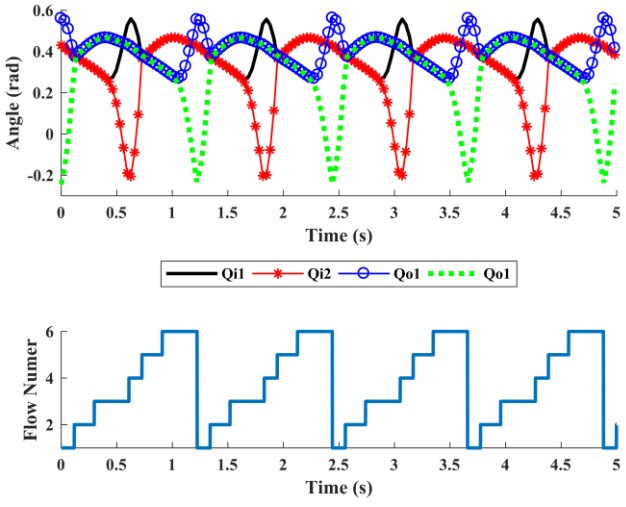

Figure 10. Angle and jump trajectories of passive walker

4. CONCLUSION AND FUTURE WORK

The new nonholonomic hybrid multibody dynamic formalism proposed in the paper has been explained and verified. The software ANDY based on the formalism has been proved successful in modeling a variety of multibody system with nonholonomic, constrained and hybrid features. The framework of ANDY also allows development of add-on features or specialized software tools.

Table 3. Comparison of symbolic modeling toolboxes

Features	ANDY	FROST* [12]	Neweul-M^2 [8]
Nonholonomic Modeling	Yes	No	No
Hybrid Modeling	Yes	Yes	No
Constrained Dynamics	Yes	Yes	Yes
Visualization Utilities	Yes	Yes	Yes
Graphic UI	No	No	Yes
Analysis Utilities	No	Yes	Yes

A small comparison against the available toolboxes for dynamic modeling can be seen in Table 3. As evident, ANDY supports nonholonomic equations and hybrid dynamics found in many dynamic models. While the basic algorithm for the multibody formalism is complete, ANDY is only in its early version. Future improvements will be made to improve calculation efficiency, analytical utilities, and user-friendliness.

ANDY[1] will serve as a quick dynamic modeling tool for future projects and researches, such as the control study and design optimization of mechatronic and robotic systems (legged robots, UAV/ROV, etc.). The toolbox will provide a balanced performance between symbolic modeling flexibility and computation efficiency. We hope the formalism and the toolbox can provide convenience to more people during their researches.

ACKNOWLEDGMENTS

The authors would like to acknowledge the effort of Mr. Yujiong Liu for his support during research.

REFERENCES

[1] Kane, Thomas R., and David A. Levinson. “Dynamics, theory and applications.” McGraw Hill, (1985).
[2] Dalmau, Daniel Sanchez-Crespo. “Core techniques and algorithms in game programming.” New Riders, 2004.
[3] A. Arbor. ADAMS/Solver Primer (2004).
[4] Sherman, Michael A., Ajay Seth, and Scott L. Delp. "Simbody: multibody dynamics for biomedical research." Procedia Iutam2 (2011): 241-261.
[5] Meurer, Aaron, et al. "SymPy: symbolic computing in Python." PeerJ Computer Science 3 (2017): e103.
[6] Kecskeméthy, Andrés, and Manfred Hiller. "An object-oriented approach for an effective formulation of multibody dynamics." Computer Methods in Applied Mechanics and Engineering 115.3-4 (1994): 287-314.
[7] Kreuzer, E., and W. Schiehlen. "NEWEUL—Software for the generation of symbolical equations of motion." Multibody systems handbook. Springer, Berlin, Heidelberg, (1990). 181-202.
[8] Kurz T, Eberhard P, Henninger C, et al. From Neweul to Neweul-M 2: symbolical equations of motion for multibody system analysis and synthesis[J]. Multibody System Dynamics, 2010, 24(1): 25-41.
[9] Bittner, Brian, and Koushil Sreenath. "Symbolic computation of dynamics on smooth manifolds." Workshop on Algorithmic Foundations of Robotics. 2016.
[10] Van Der Schaft, Arjan J., and Johannes Maria Schumacher. “An introduction to hybrid dynamical systems.” Vol. 251. London: Springer, 2000.
[11] R. Tedrake, “Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems.” http://drake.mit.edu, 2016.
[12] Hereid, Ayonga, and Aaron D. Ames. "FROST*: Fast Robot Optimization and Simulation Toolkit." (2017): 719-726.
[13] Sayers, Michael W. "Symbolic vector/dyadic multibody formalism for tree-topology systems." Journal of Guidance, Control, and Dynamics 14.6 (1991): 1240-1250.
[14] Rone, William S., and Pinhas Ben-Tzvi. "Continuum robot dynamics utilizing the principle of virtual power." IEEE Transactions on Robotics 30.1 (2014): 275-287.
[15] Kane, Thomas R., and David A. Levinson. Dynamics, theory and applications. McGraw Hill, 1985.
[16] Remy, C. David. “Optimal exploitation of natural dynamics in legged locomotion.” Diss. ETH Zurich, 2011.
[17] Witkin, Andrew. "An introduction to physically based modeling: Constrained dynamics." Robotics Institute (1997).
[18] Sanfelice, Ricardo, David Copp, and Pablo Nanez. "A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid Equations (HyEQ) Toolbox." Proceedings of the 16th international conference on Hybrid systems: computation and control. ACM, 2013.
[19] Diebel, James. "Representing attitude: Euler angles, unit quaternions, and rotation vectors." Matrix 58.15-16 (2006): 1-35.
[20] Craig, John J. Introduction to robotics: mechanics and control. Vol. 3. Upper Saddle River, NJ, USA:: Pearson/Prentice Hall, 2005.
[21] An, Kang, and Qijun Chen. "A passive dynamic walking model based on knee-bend behaviour: stability and adaptability for walking down steep slopes." International Journal of Advanced Robotic Systems 10.10 (2013): 365.
[22] Hurmuzlu, Yildirim, and Dan B. Marghitu. "Rigid body collisions of planar kinematic chains with multiple contact points." The international journal of robotics research 13.1 (1994): 82-92.

[1] ANDY is desired to be open source and is available at: https://github.com/RM-Lab/ANDY. It comes packaged with examples highlighted in this manuscript.