FROST*: Fast Robot Optimization and Simulation Toolkit

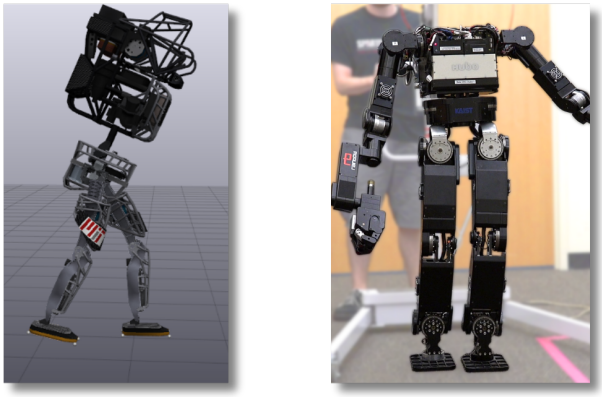
Ayonga Hereid[1] and Aaron D. Ames[2]

***Abstract*— This paper presents FROST, an open-source MATLAB toolkit for modeling, trajectory optimization and simulation of hybrid dynamical systems with a particular focus in dynamic locomotion. The design objective of FROST is to provide a unified software environment for developing model-based control and motion planning algorithms for robotic systems whose dynamics is hybrid in nature. In particular, FROST uses directed graphs to describe the underlying discrete structure of hybrid system models, which renders it capable of representing a wide variety of robotic systems. Equipped with a custom symbolic math toolbox in MATLAB using Wolfram Mathematica, one can rapidly prototype the mathematical model of robot kinematics and dynamics and generate optimized code of symbolic expressions to boost the speed of optimization and simulation in FROST. In favor of agile and dynamic behaviors, we utilize virtual constraint based motion planning and feedback controllers for robotic systems to exploit the full-order dynamics of the model. Moreover, FROST provides a fast and tractable framework for planning optimal trajectories of hybrid dynamical systems using advanced direct collocation algorithms. FROST has been successfully used to synthesize dynamic walking in multiple bipedal robots. Case studies of such applications are considered in this paper, wherein different types of walking gaits are generated for two specific humanoid robots and validated in simulation.**

I. Introduction

The rapid advancement of mechanical and actuation capabilities of modern robots has made increasingly more agile and robust locomotion on robots possible. To exploit the full dynamic capabilities of the machine, it is necessary for researchers to develop systematic approaches that consider the mathematical model of the robot when planning and controlling dynamic behaviors of a robot. In this paper, we introduce FROST—an open-source MATLAB toolkit aiming to provide an integrated software environment for developing model-based control and planning algorithms for robotic systems, specifically for dynamic legged locomotion, even for multi-contact foot behaviors and different actuation types.

In many applications of legged robots, heuristic-based simplified model techniques are still the most popular approaches due to their simplicity and maturity [16], [22]. While these simple models may provide intuitive understandings of a certain aspects of the system and often have many implementation advantages, they also limit the flexibility of the mechanical systems and often result in rather artificial behaviors. The development of more advanced optimization techniques in recent years has led to an increasing trend in the development of optimization based whole body planning algorithms which utilize more complicated dynamic models. [7], [8], [13], [17]. These applications typically use general-purpose nonlinear programming solvers, e.g., IPOPT [25] or SNOPT [9], to formulate the motion planning problems, which often requires experts' knowledge of trajectory optimization methods. Moreover, one may use state-of-the-art optimal control toolboxes, such as GPOPS [20], DIRCOL [24] or PSOPT [5], that come with advanced trajectory optimization algorithms. But, they either lack an effective framework for users to construct the robot model or are not capable of solving large-scale problems that are very common for high-dimensional robotic systems. Hence, our objective is to develop an integrated tool for modeling, optimization and simulation of robotics systems.



(a) ATLAS (b) DRC-HUBO

Fig. 1: The ATLAS and DRC-HUBO robots for which the FROST framework is applied in this work to generate walking motions.

The mathematical foundation of FROST is the hybrid dynamical system and virtual constraints [26] that were designed to mathematically synthesize stable controllers for realizing dynamical behaviors, such as bipedal locomotion. By enforcing virtual constraints that are invariant through impact via feedback controllers, the stability properties of the high-dimensional system are effectively captured in a lower-dimensional representation, termed the *hybrid zero dynamics*. If one can find such set of virtual constraints, then there exists a class of feedback controllers [4], [26] that can be used to stabilize the full-order robot dynamics. Therefore, optimal motion generation via virtual constraints

* The online documentation and source code of FROST can be found: http://ayonga.github.io/frost-dev/

[1]Ayonga Hereid is a postdoctoral research fellow of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48105, USA ayonga@umich.edu

[2]Aaron D. Ames is with the Faculty of Mechanical and Civil Engineering, Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA ames@cds.caltech.edu

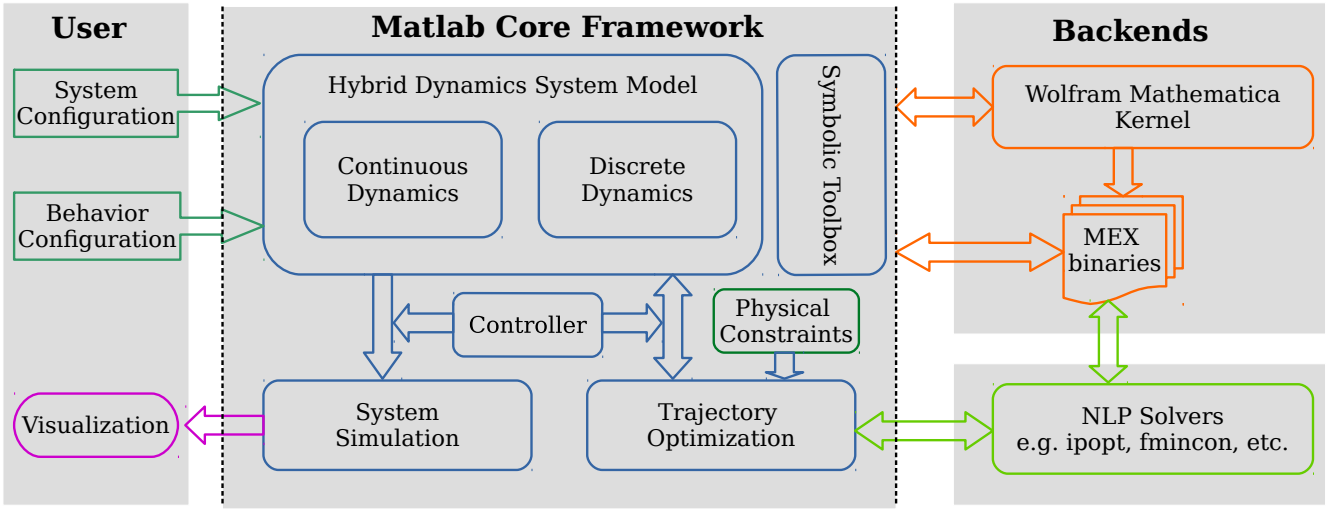


Fig. 2: The block diagram illustration of the FROST architecture.

unifies the motion planning and the stabilizing controller synthesis problem. In FROST, the motion planning problem becomes a nonlinear constrained optimization problem that determines an appropriate set of virtual constraints.

The FROST architecture is depicted in Figure 2. It consists of two main components: the core framework implemented in MATLAB and the back-end engines, including the Mathematica symbolic engine and the NLP solver. FROST provides a unified framework that is centered around a hybrid dynamical system model. FROST also utilizes the Mathematica kernel to compute the symbolic expressions for robot kinematics and dynamics to boost the computational speed of the optimization and simulation. More importantly, FROST automatically constructs a (hybrid) trajectory optimization problem for a (hybrid) dynamical system using advanced direct collocation methods. The objective of this paper is to introduce the key elements of FROST and illustrate it on a collection of case studies; in particular, generating 3D walking gaits for ATLAS [21] and DRC-HUBO [14] (shown in Figure 1).

The structure of this paper is as follows. Section II introduces the mathematical modeling of general hybrid dynamical systems, as well as rigid body kinematics and dynamics of robotics, Section III presents the virtual constraint based control design and the simulation of hybrid dynamical systems, and Section IV briefly discusses the hybrid trajectory optimization using direct collocation. Lastly, we present two specific applications of FROST for 3D humanoid locomotion in Section V followed by the discussion and conclusion in Section VI.

II. Modeling of Hybrid Dynamical System

FROST provides an abstract framework to formally construct hybrid dynamical system models. In this section, we discuss this feature in the context of robotic systems.

A. Hybrid System and Directed Graph

Hybrid systems are systems that exhibit both continuous and discrete dynamics, and thus have a wide range of applications to various types of physical systems undergoing impacts [11]. For instance, legged locomotion often consists of a collection of continuous phases, with discrete events triggering transitions between neighboring phases [10]. Specifically, we use the following definition to model a hybrid system in FROST.

Definition 1. A *hybrid system* is a tuple,

$$\mathscr{HC} = (\Gamma, \mathcal{D}, \mathcal{U}, S, \Delta, FG), \tag{1}$$

where $\Gamma = \{V, E\}$ is a *directed graph*, $\mathcal{D}$ is the admissible configuration of the continuous domains, $\mathcal{U}$ represents the admissible controllers, S determines guard conditions that trigger discrete transitions, and Δ and FG represent the discrete and continuous dynamics respectively [3].

In FROST, the discrete structure of a hybrid system is represented via a directed graph. The graph may be either cyclic or acyclic or a combination of both based on the definition of the behavior that the graph describes, see Figure 3. A directed graph consists of two elements: a set of vertices V that represent the continuous phases of a dynamical system and a set of edges E that represent the discrete events of the behavior. FROST allows dynamically adding or removing vertices and edges for the purpose of flexibility. When using a directed graph to model the ordering structure of a locomotion behavior, each vertex represents a admissible continuous phase (a.k.a. domain) determined by the Lagrangian model of the mechanical system and physical constraints of the robot (e.g., contacts); each edge represents a possible discrete transition between two continuous phases occuring at a change in the physical constraints, e.g., establishing new contacts or breaking existing contacts. For instance, Figure 3a could represent periodic locomotion and

Figure 3b represents a non-periodic gait, whereas Figure 3c may model the entire behavior of robot moving from the rest position to a periodic motion and ultimately to stopping.

(a) (b) (c)

Fig. 3: Illustration of different types of directed graphs. (a) a simple directed cycle; (b) a acyclic graph; (c) a general directed graph that contains cyclic and acyclic sub-graphs.

B. Continuous and Discrete Dynamics

Continuous Dynamics. The continuous state dynamics describe the evolution of system states governed by differential equations on a smooth manifold determined via a set of algebraic constraints. Let $x \in \mathcal{D}$ be a set of coordinates of the system, with $\mathcal{D} \subseteq \mathcal{X}$ being a smooth embedded manifold in the configuration space $\mathcal{X}$, the continuous dynamics equations can be given the following form of ordinary differential equations (ODEs):

$$\text{First-order ODEs:} \quad M(x)\dot{x} = F(x) + G(x,u) \tag{2}$$

$$\text{Second-order ODEs:} \quad M(x)\ddot{x} = F(x,\dot{x}) + G(x,u) \tag{3}$$

where $M(x)$ is the positive definite mass matrix, $F(x)$ or $F(x,\dot{x})$ is a set of drift vectors, and $G(x,u)$ is a set of input vectors with u being the system input variables. Input variables could be either control inputs of the actuators or other external inputs induced from the physical constraints of the system. FROST uses the `ContinuousDynamics` class to represent continuous state dynamical systems, allowing users to construct a system object by configuring the symbolic representation of states and inputs variables and mathematical expressions of each element of system ODEs. One specific example of such systems is a robot model consisting of rigid bodies—e.g., a biped—whose dynamics is governed by the Euler-Lagrangian equations of motion [19], in which the inertia matrix corresponds to the mass matrix, and the Coriolis matrix and gravity vectors correspond to the drift vectors.

Holonomic Constraints. In addition to the differential equations, the continuous dynamics is often constrained by a set of algebraic equations, resulting in a constrained dynamical system. These constraints could be either holonomic or nonholonomic, however, FROST currently only supports holonomic constraints given as $h_c(x) \equiv \text{constant}$. To impose holonomic constraints on a dynamical system, we enforce the second order derivatives of $h_c(x)$ must be zero, i.e.,

$$J_c(x)\ddot{x} + \dot{J}_c(x,\dot{x})\dot{x} = 0. \tag{4}$$

where $J_c(x) = \frac{\partial h_c(x)}{\partial x}$ is the Jacobian of the holonomic constraints. For each holonomic constraint, there exists a constraint wrench λ_c acting on the system with the associated input vector of this wrench being: $J_c^T(x)\lambda_c$. Because the constraint wrenches are the external inputs used to enforce the holonomic constraints, their magnitude will be determined from (4). FROST uses the `HolonomicConstraint` class to model holonomic constraints. Users can add or remove particular holonomic constraints objects to the system. The associated input vectors and constraint wrenches will be automatically configured during these process. For robotic systems, holonomic constraints may be used to model many physical constraints of the system. For instance, a fixed joint or a physically connected linkages such as a four-bar linkage can be expressed as a holonomic constraint.

Unilateral Constraints. We also use holonomic constraints to model rigid contacts of the robot with the environment. That is, the following complementarity conditions should hold: either body cannot exert a force when there is no contact and the bodies that constitute a contact cannot interpenetrate each other [15]. While a holonomic constraint describes how a contact effects the dynamics of the system, it ignores limitations of the complementarity conditions. For instance, the wrenches from the ground-foot contacts of a legged robot must be within the spatial friction cone and satisfy the zero moment point criteria, and the normal force must be positive. The distance between two bodies cannot be negative. To accommodate these limitations, the `UnilateralConstraint` class is introduced as an inequality condition of state or input variables. Unlike holonomic constraints, unilateral constraints are not strictly enforced in the system simulation, but are instead considered as necessary conditions for the control law design and trajectory optimization. Some unilateral conditions are also used as event functions to determine the triggering conditions for discrete events of the hybrid system model[1].

Discrete Dynamics. When the evolution of a system transitions from one continuous phasae into another, its states often undergo an instantaneous change. In FROST, we use the `DiscreteDynamics` class to characterize these instantaneous changes in state variables. A `DiscreteDynamics` object determines the followings: 1) the event condition that triggers the change, and 2) a *reset map* that represents the mathematical formulation of the change, which can be given expressed in the following manner:

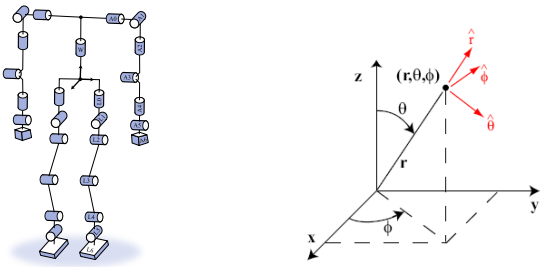
$$x^+ = \Delta_x(x^-), \tag{5}$$

where $x^- = \lim_{t \nearrow t_0} x(t)$ and $x^+ = \lim_{t \searrow t_0} x(t)$ with t_0 be the time instant at which the discrete dynamics occurs. For a second-order system, the first order derivative of x should also be considered, i.e.,

$$\dot{x}^+ = \Delta_{\dot{x}}(x^-)\dot{x}^-, \tag{6}$$

where $\dot{x}^- = \lim_{t \nearrow t_0} \dot{x}(t)$ and $x^+ = \lim_{t \searrow t_0} \dot{x}(t)$. For the sake of generalization, we consider a transition with no

[1]Ideally, all unilateral constraints are event functions. For simplicity, however, FROST simulation only monitors event functions that have associated edges predefined in the hybrid system model.



(a) Kinematic tree (b) Coordinate frame

Fig. 4: Illustration of a robot kinematic tree structure and coordinate frame.

instantaneous change of states as a discrete dynamics model with $\Delta = \boldsymbol{I}$ being an identity matrix.

C. Rigid Body Kinematics and Dynamics

With the goal of providing an user-friendly environment dedicated to the development of dynamic locomotion of legged robots, FROST features a rich set of functions and classes to model robotic systems in the context of hybrid dynamical systems.

Robot Kinematics. We use the Universal Robot Description Format (URDF) as a standard to specify the rigid body tree structure of a robot, see Figure 4a. Currently, FROST only supports the core components (e.g., joints, links, and transmissions) of URDF description of a robot. Moreover, the following group of classes are provided by FROST to construct the kinematic model from the robot's URDF file:

- `CoordinateFrame` defines an arbitrary coordinate frame in the $SO(3)$ space, as shown in Figure 4b. Each coordinate frame has a reference frame, and its configuration is determined by the offset from the origin of the reference and the rotation expressed in the reference frame. If the reference frame is empty, then it is assumed to be the world frame.
- `RigidJoint` defines a coordinate frame located at the origin of a joint and rigidly attached to its child link, with the previous joint in the kinematic tree as its reference. It also defines the joint type, rotation axis, child and parent link, limits of the joint, and actuator properties described in the URDF file.
- `RigidBody` defines a coordinate frame rigidly attached to a link and located at its center of mass, with its parent joint frame as its reference. It also defines the mass and inertia properties of the rigid link.
- `ContactFrame` defines a coordinate frame located at the contact point of the robot with the environment. It also defines the type of kinematic contacts, determines whether the contact is point, line, or planar contact. In particular, we always assume that the $z-$axis of the contact frame is normal to the contact surface.

Given a URDF file of a robot, FROST automatically parses its contents to construct coordinate frame objects for the joints and links. The joint and link objects are then used to construct an object of `RobotLinks` class inherited from the `ContinuousDynamics`. That is being said, this class represents a particular type of continuous dynamical system—rigid body dynamical system. Contacts can be added or removed after an object of such class is created. `RobotLinks` also provides methods for conveniently computing mathematical expression the forward kinematics and the spatial/body Jacobians etc. of a coordinate frame.

Robot Dynamics. Once the `RobotLinks` object is created, the mathematical expression of the Euler-Lagrangian equations of motion will be automatically computed symbolically using a custom Mathematica package based on the screw theory described in [19]. The dynamical equations are automatically then formulated as the ODEs in (3).

Rigid Impact. A rigid impact is a special type of discrete dynamics describing the impulsive contact between two rigid bodies, such as the robot feet striking the ground. Following the hypotheses in [15] and [10], we assume that the impact occurs instantaneously, and that the two impacted bodies stick to each other post-impact. Therefore, the robot configuration remains the same but the joint velocities will have impulsive changes. The reset map can be computed from the impact constraints. For more details, we refer the readers to [10].

III. Control Design and Simulation

The most distinctive feature of FROST is its use of virtual constraint based feedback control. In this section, we briefly introduce the default control design and the simulation of hybrid system in FROST.

A. Virtual Constraint based Feedback Controller

Analogous to holonomic constraints, virtual constraints are defined as functions of state variables that modulate the robot links in order to achieve a certain desired behavior, e.g., a walking gait, via feedback controllers [26]. In general, a virtual constraint is defined as the difference between the actual output $y^a(x)$ and the desired output $y^d(\alpha, \tau)$:

$$y(x, \alpha) := y^a(x) - y^d(\alpha, \tau) \tag{7}$$

where α is a set of parameters describing the virtual constraint, with τ a monotonically increasing or decreasing timing variable. FROST provides very flexible ways of defining virtual constraints. The default form used for desired outputs is the Bézier polynomial. Users have the freedom to choose the order of the polynomial or even use other function forms, such as canonical fuman walking function (CWF) used [3]. The phase variable τ can be either a time or state-based function. Users are also allowed to define the relative degree of the output. Due to the space limitations, we refer the readers to the online documentation of FROST for more details regarding the configuration of virtual constraints.

Given a virtual constraint y for a dynamical system, one can use classical feedback linearization control to drive $y \to 0$ [26]. Such controller has the following form:

$$u = -A^{-1}(L_f + \mu), \tag{8}$$

where A is a decoupling matrix, L_f represents the nonlinear dynamics of the system, and μ is the auxiliary input. If we assume that the virtual constraint has relative degree 2, then under this control we have:

$$\ddot{y} = -\mu. \tag{9}$$

One can choose the μ such that the linear dynamics in (9) is stable. In addition, advanced optimal controllers based on control Lyapunov functions (CLFs) and quadratic programming can be applied to stabilize the outputs [4]. As a result of the virtual constraint based methodology used by FROST, the end result is that the FROST automatically generates controllers to achieve dynamic motions either via feedback linearization or CLFs.

Remark 1. The idea of virtual constraints is based on the method of computed torques [26], the total number of outputs should not exceed the total degrees of actuation and total unconstrained degrees of freedom, whichever is smaller. If the number of outputs is more than the total unconstrained degrees of freedom—which is the total degrees of freedom minus the number of holonomic constraints—the system becomes an over-constrained system. Such over-constrained systems should be avoided in the control law design.

Algorithm 1 Hybrid Dynamics Simulation

```
parse:
    simulation options
initialize:
    x(0) ← x0
    vertex ← the starting vertex
while true do
    if vertex is empty or terminal then
        break
    end if
    if N_cycle ≥ max(cycle) then
        break
    end if
    do
     forward simulate the continuous phase
    while guard conditions not triggered
    update:
        edge ← the triggered guard
        vertex ← tar(edge)
    discrete dynamics: x(t+) ← Δ_e x(t−)
end while
```

B. Hybrid System Simulation

Once a hybrid system model of a specific behavior and the virtual constraints based feedback controller are constructed, FROST can simulate and validate the hybrid dynamical system. The simulated subject could be the entire directed graph of the system or a certain part of the graph — a sub-graph. The simulation of the sub-graph can be specified via the simulation option, in which a sub-graph can be either directly defined or constructed by specifying the starting and terminal vertices within the original graph. FROST also allows for one to specify the number of cycles if the simulated hybrid system model is a simple cyclic graph.

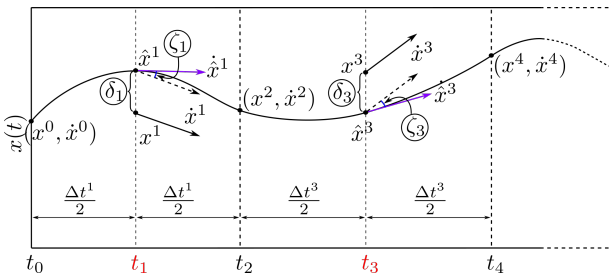


Fig. 5: Defect constraints for Hermite-Simpson collocation.

A simplified procedure for hybrid dynamics simulation is shown in Algorithm 1. The simulator starts with a certain vertex in the graph, and then iterates through the graph until it reaches a terminal vertex which has no successor or it reaches the specified number of cycles. The simulated trajectory of the hybrid dynamical system then can be visualized by 3D animators via RViz or a custom stick animator provided by FROST. The default 3D animator of FROST currently does not support parsing the mesh files defined in URDF files.

IV. Hybrid Trajectory Optimization

The main functionality of FROST is the fast and scalable trajectory optimization algorithm for hybrid dynamical systems. When virtual constraints are used to represent a certain behavior or a gait, the optimization not only generates an optimal trajectory, it also optimizes a set of virtual constraint parameters, and hence feedback controllers.

A. Direct Collocation Optimization

Given a hybrid dynamical system model, FROST automatically constructs a multi-phase hybrid trajectory optimization nonlinear programming (NLP) problem. Each phase itself is a single trajectory optimization problem representing either continuous or discrete dynamics[1].In particular, direct collocation methods are used to solve the continuous dynamics trajectory optimization problem [6]. FROST supports various types of direct collocation methods. Here we briefly introduce the default Hermite-Simpson collocation.

First, we uniformly discretize the continuous phases into $N \geq 0$ grids, then the even-numbered discrete nodes at the terminals of each grid are called *cardinal nodes* and the odd-numbered nodes at the middle of each grid are called *interior nodes* (see Figure 5). The state (including first and second order derivatives, if appropriate) and input variables of the continuous dynamics at these discrete nodes will be automatically introduced as optimization variables. If the defects δ and ζ at all interior nodes become zero and the discrete state and input variables satisfy system dynamics at all nodes, then there exist a piecewise continuous cubic polynomials determined by the discrete state variables accurately approximating the solution of differential equations

[1]Phases representing discrete dynamics have a time horizon of zero.

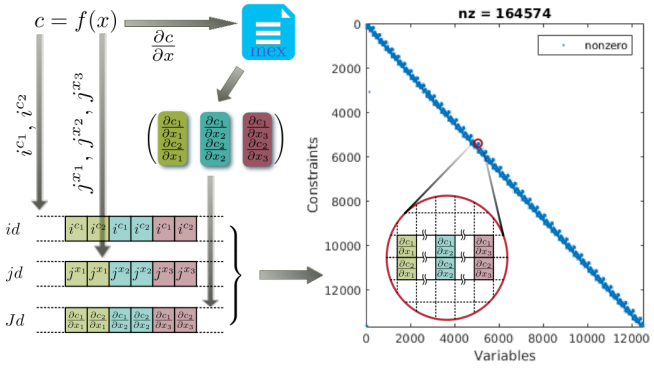


Fig. 6: Illustration of sparse Jacobian matrix construction.

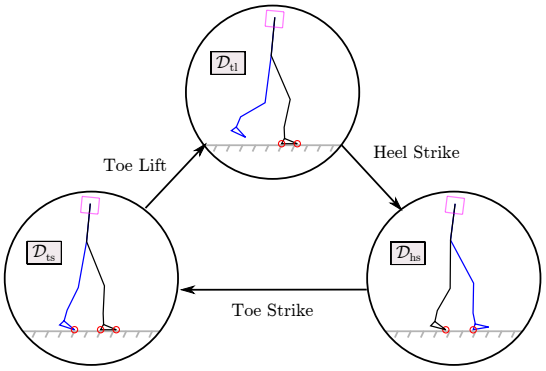


Fig. 7: The directed graph of the multi-contact periodic walking for ATLAS.

of the system [1]. Then the hybrid trajectory optimization problem that has N_p phases becomes a constrained nonlinear programming problem:

$$\text{argmin} \quad \sum_{i=1}^{N_p} \int_{t_0^i}^{t_f^i} \mathcal{L}_i(\cdot)dt + E_i(\cdot) \tag{10}$$

subject to defect constraints, system dynamics and other physical constraints, where $\mathcal{L}_i(\cdot)$ and $E_i(\cdot)$ are the running and terminal cost of each phase, respectively. FROST will automatically enforce the holonomic constraints, unilateral constraints and virtual constraints defined on a continuous phase as physical constraints of the optimization problem. For the detailed mathematical formulation of the problem, we refer the readers to [1].

Different from other direct collocation trajectory optimization toolboxes, FROST introduces *defect variables*, also called *slack variables*, to avoid computing the closed form system dynamics explicitly. Instead, we enforce the system dynamics in its raw form of differential algebraic equations (a combination of (3), (4), and (9)) directly. The idea of defect variables is very simple. Assume that the original constraint has a form: $a(b(x)) = 0$. This can be decoupled by introducing a defect variable y and re-formulate the constraint as:

$$a(b(x)) = 0 \quad \Longleftrightarrow \quad b(x) = y, \;\; a(y) = 0 \tag{11}$$

The introduction of the defect variables not only improves the local linearity of the constraints, but also makes it possible to compute the first and second order derivatives of the nonlinear constraints symbolically. FROST provides a custom symbolic math toolbox that uses Mathematica kernel as backend. Using this toolbox, we can compute the analytic gradients of a given nonlinear function and export them as executable MEX binaries for the solver to use.

B. Interface to NLP Solvers

Once the hybrid trajectory optimization NLP problem is formulated, it can be solved by state-of-the-art NLP solvers, such as IPOPT [25], SNOPT [9], or Fmincon, etc. In FROST, we design solver interface classes that transform the original NLP problem to a specific format compatible with the solver. With these interfaces, the same NLP problem can be seamlessly solved by different NLP solvers without changing the original problem. In particular, the interface first stacks all NLP variables, constraints and cost functions into 1-D arrays. During this process, variables, constraints and cost functions will be indexed based on their location in the corresponding array. To further expedite the run-time evaluation, we exploit the sparsity pattern of the Jacobian matrix. Thanks to the symbolic calculation of analytic Jacobians, we can determine the indices of all the non-zero entries of the Jacobians even before running the optimization. By only computing the value of these non-zero entries at runtime, one can easily construct a sparse Jacobian matrix using the pre-computed index information, as shown in Figure 6. Considering that the sparsity of the Jacobian matrix is often less than 1%, this process saves a significant amount of computation time.

V. Case Studies

In this section, we demonstrate the application of FROST for dynamic legged locomotion through two 3D humanoids.

A. ATLAS: Multi-Contact Periodic Walking

Modeling. In this example, we consider the lower-body ATLAS robot model described in [17], which has 6 degrees of freedom in each leg and 3 degrees of freedom in its upper body. The upper body joints are considered as fixed, which will be enforced as holonomic constraints. We consider a periodic walking gait with 3 different contact configurations, resulting in a multi-domain hybrid system model with a directed cycle $\Gamma = \{V, E\}$ as shown in Figure 7, given as

$$V = \{\text{ts}, \text{tl}, \text{hs}\}, \tag{12}$$

$$E = \{\text{ts} \to \text{tl}, \text{tl} \to \text{hs}, \text{hs} \to \text{ts}\}. \tag{13}$$

The contact configuration of each domain is defined as follows (assuming all contacts have friction):

- D_{ts} has a planar contact at the stance foot and a line contact at the non-stance toe;
- D_{tl} has a planar contact at the stance foot;
- D_{hs} has a line contact at the stance toe and a line contact at the non-stance heel.

In addition, we define the linearized forward velocity of the robot as our velocity-modulating output, and correspondingly, choose the linearized forward hip position as the

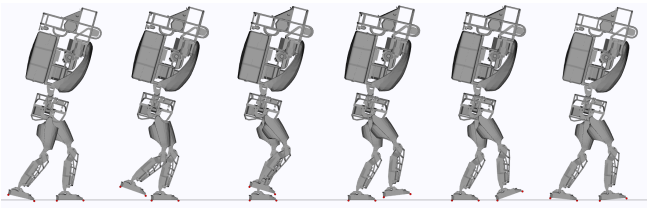


Fig. 8: Tiled still images from the simulation of Atlas multi-contact walking in 3D at $0.51m/s$.

phase variable [3]. The detailed selection of these position-modulating outputs is omitted in this paper due to space limitations, and refer readers to [1].

Gait Optimization. Given this hybrid system model object, FROST automatically formulates a hybrid trajectory optimization problem. The limits on joint torques, velocities, and position, and the unilateral conditions on the contact wrenches will be automatically enforced via FROST. In particular, we choose the cost function to be the mechanical cost of transport of the walking gait, which is given as the total mechanical work done by the actuators divided by the weight of the robot and the distance traveled during one step. Additional physical constraints, such as swing foot clearance, avoidance collision, etc., will be also considered in the gait optimization problem.

In this example, we use IPOPT to solve the NLP problem, the solver converged to a feasible solution after with an average of 988 seconds and 744 iterations when using a randomly generated initial guess. These numbers are reduced to 269 seconds and total of 246 iterations when seeded with relatively "good" initial guesses taken from the previous results. The convergence speed of FROST gait optimization is notably faster than another full-body dynamics open loop gait optimization algorithm—DIRCON—implemented in Drake, which requires about 10 minutes to two hours to converge for the same ATLAS application [21] using SNOPT. While the choice of different solvers could potentially affect the convergence time, the result shows FROST is able to provide the same or even marginally better performance than DIRCON, the state-of-the-art full-body dynamics gait generation framework, in this particular case.

Simulation. We also show the simulation result of one optimal walking gait in Figure 8. The step length of this gait is 0.48 m, the time duration of a complete step is 0.94 seconds and the mechanical cost of transport (COT) is 0.201. Because in the simulation we use the same feedback controllers as used in the optimization via the linear output dynamics constraints [1], there is no increase of the COT numbers in simulation. When using QP-based CLF feedback controllers in the simulation [4], we noticed a slight reduction in the COT number, yielding 0.19. Compared to the result in [21], the gait discussed in this work has slower walking speed but marginally smaller cost of transport number. More importantly, the COT number does not increase in simulation as it happens with DIRCON which constructs a LQR type feedback controller to stabilize the open-loop optimal trajectory [21].

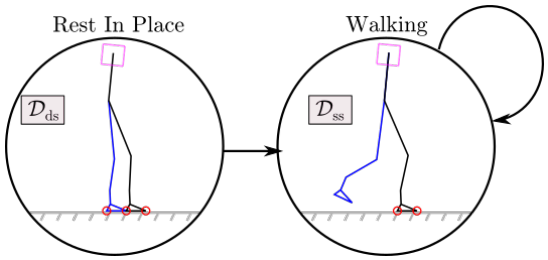


Fig. 9: The directed graph description of DRC-HUBO walking from the rest to a periodic gait.

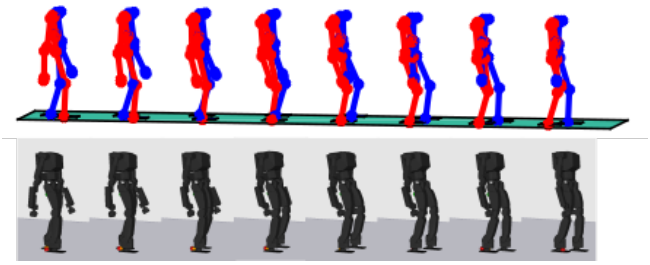


Fig. 10: Walking tiles of DRC-HUBO simulation.

B. DRC-HUBO: Dynamic Arm Swing

In this example, we consider the 3D full humanoid, DRC-HUBO, a fully-actuated humanoid with 27 actuators [14]. The kinematic model includes the upper body arm links, which are allowed to move while the robot is walking. In many gait planning algorithms, the arm motion is often ignored, which can be a missed opportunity. Beyond the obvious manipulation tasks, arms can be helpful for improving the balance and economy of locomotion by swinging them as part of a dynamic gait. As such, we use FROST to fully exploit the internal dynamics of the robot. Importantly, the humanoid swings its arms as a consequence of optimizing the dynamic gait for energy-efficient locomotion subject to no-net-moment constraints, not by *a priori* specification.

In particular, we generated a walking motion in FROST that starts from a rest position to a single-domain periodic gait. An acyclic hybrid system model was constructed to model such behavior, as shown in Figure 9. More liberal constraints on arm-joint velocities and effective foot size as well as slower stepping frequencies are allowed during the gait optimization process. As shown in Figure 10, this gait exhibits very natural-looking counter-rotating arm swing motion to minimize the total mechanical energy consumed. Typically such optimization uses 7-10 minutes on a laptop computer when using IPOPT with linear solver `ma57` [25]).

To validate the stability of the gait, we simulated the DRC-HUBO walking in DART simulation environment[1], which uses LCP-based contact models to perform more "real-world" simulation. An animation of DART simulation result is also shown in the bottom figure of Figure 10, The center of mass position evolution over time in both FROST and

[1]The DART simulation environment is available at https://github.com/dartsim/dart.

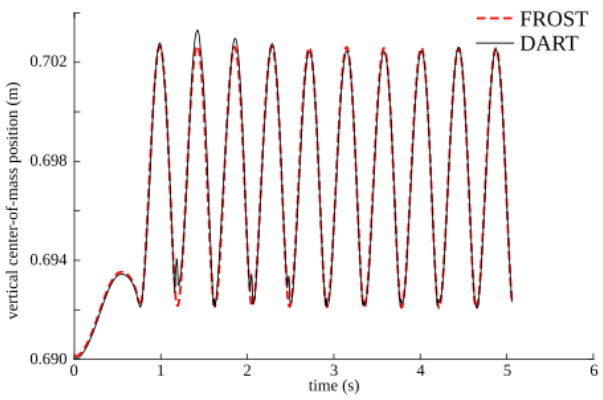


Fig. 11: Center-of-mass position over time for the simulated gait in FROST and DART.

DART simulation is shown in Figure 11, which shows a good consistency between these two simulation environments.

VI. Conclusions and Future Work

In this paper, we introduced FROST, an integrated software package for synthesizing highly dynamic legged locomotion within the HZD framework. The use of directed graph descriptions for hybrid system modeling allows FROST to model a wide range of dynamic behaviors on a given robot. By transforming the virtual constraint optimization into a large-scale hybrid trajectory optimization problem, FROST is able to generate versatile behaviors for various high-dimensional humanoid robots [1], [2], [12], [18], [23]. Further, FROST provides a unified and generalized software package for users who are less familiar with the HZD framework to design model-based gaits and stabilizing controllers for their robots. Future work involves faster optimization algorithms and advanced feedback controllers that allow versatile behaviors of robotic systems.

Acknowledgment

The authors would like to thank the members of Dr. Jessy Grizzle's research group at the University of Michigan for their feedback and discussion on dynamic gait optimization.

References

[1] Yongga A. *Dynamic Humanoid Locomotion: Hybrid Zero Dynamics Based Gait Optimization via Direct Collocation Methods*. PhD thesis, Georgia Institute of Technology, 2016.

[2] A. Agrawal, O. Harib, A. Hereid, S. Finet, M. Masselin, L. Praly, A. D. Ames, K. Sreenath, and J. W. Grizzle. First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons. *IEEE Access*, 5:9919–9934, 2017.

[3] A. D. Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, May 2014.

[4] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control (TAC)*, 59(4):876–891, April 2014.

[5] Victor M Becerra. Solving complex optimal control problems at no cost with PSOPT. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1391–1396. IEEE, 2010.

[6] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[7] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[8] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 295–302. IEEE, 2014.

[9] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[10] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames. Models, feedback control, and open problems of 3D bipedal robotic walking. *Automatica*, 50(8):1955 – 1988, 2014.

[11] J. Guckenheimer and S. Johnson. Planar hybrid systems. In *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 202–225. Springer Berlin Heidelberg, 1995.

[12] Ayonga Hereid, Shishir Kolathaya, and Aaron D Ames. Online hybrid zero dynamics optimal gait generation using Legendre pseudospectral optimization. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2016.

[13] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 874–880. IEEE, 2015.

[14] Christian Hubicki, Ayonga Hereid, Michael Grey, Andrea Thomaz, and Aaron Ames. Work those arms: Toward dynamic and stable humanoid walking that optimizes full-body motion. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[15] Y. Hurmuzlu, F. Génot, and B. Brogliato. Modeling, stability and control of biped robots–a general framework. *Automatica*, 40(10):1647–1664, 2004.

[16] S Kajita, F Kanehiro, K Kaneko, K Yokoi, and H Hirukawa. The 3D linear inverted pendulum model: a simple modeling for a biped walking pattern generation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 239–246, 2001.

[17] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Auton. Robots*, 40(3):429–455, March 2016.

[18] Wen-long Ma, Ayonga Hereid, Christian M. Hubicki, and Aaron. D. Ames. Efficient HZD gait generation for three-dimensional underactuated humanoid running. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2016.

[19] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[20] Michael Patterson and Anil Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[21] M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373, May 2016.

[22] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133, August 2012.

[23] Jacob Reher, Ayonga Hereid, Shishir Kolathaya, Christian M. Hubicki, and Aaron D. Ames. Algorithmic foundations of realizing multi-contact locomotion on the humanoid robot DURUS. In *the 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer, 2016.

[24] Oskar von Stryk. Users guide for DIRCOL version 2.1. *Simulation and Systems Optimization Group, Technische Universität Darmstadt, www. sim. informatik. tu-darmstadt. de/sw/dircol*, 1999.

[25] Andreas Wächter and T. Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.

[26] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press Boca Raton, 2007.