Rapid Bipedal Gait Optimization in CasADi

Martin Fevre, Patrick M. Wensing, and James P. Schmiedeler

***Abstract*— This paper shows how CasADi's state-of-the-art implementation of algorithmic differentiation can be leveraged to formulate and efficiently solve gait optimization problems, enabling rapid gait design for high-dimensional biped robots. Comparative studies on a 7-DOF planar biped show that CasADi generates optimal gaits 4 times faster than another existing advanced optimization package. The framework is also applied to simultaneously generate a gait and a feedback controller for 2 spatial bipeds: a 12-DOF model and a 20-DOF model. Results suggest that CasADi's unprecedented efficiency could provide a practical path toward real-time gait optimization for high-dimensional biped robots.**

I. Introduction

Dynamic biped walking research began with planar models [1], but has been extended to 3D [2], where the control challenges are substantially more complex. Still, much research continues to limit biped locomotion to the plane because the easier analysis leads to faster simulations and optimizations. Indeed, the curse of dimensionality makes real-time gait optimization nearly impossible for high-dimensional bipeds, and reduced-order template models are often used instead to map simplified walking patterns onto the full dynamics [3], [4], [5]. While reduced-order models simplify motion planning, they unavoidably neglect important physical constraints, which may yield impractical gaits.

The open-source toolbox Drake provides an efficient C++ platform for rapid trajectory optimization of complex robotic systems such as bipeds [6]. Similarly, FROST [7] is an open-source MATLAB toolkit for modeling trajectory optimization of hybrid dynamical systems, including bipeds. More generally, the open-source CasADi [8] software allows for efficient modeling of optimal control problems [9], [10], [11], [12] and has been used to optimize stand-up and sit-down humanoid trajectories [13] and back-flip trajectories for the Mini Cheetah quadruped [14]. For optimization problems involving high-dimensional systems, CasADi's state-of-the-art implementation of algorithmic differentiation (AD) for computing and storing derivative information offers computational advantages. The symbolic expressions in CasADi are represented by directed graphs of symbolic primitives, rather than as tree structures like in conventional computer algebra systems. Further, the sparsity patterns that arise naturally in trajectory optimization problems can be generated efficiently and automatically by CasADi to accelerate optimization.

This work was supported in part by the National Science Foundation under Grants IIS-1734532, IIS-1527393, and CMMI-1835186. Corresponding author: mfevre@nd.edu

The authors are with the Department of Aerospace & Mechanical Engineering, University of Notre Dame, IN USA.

Fig. 1: **TR**ajectory **OP**timization **I**n **C**asADi is available at: https://github.com/fevrem/TROPIC

This paper introduces TROPIC (Fig. 1), an open-source CasADi-based optimization package that enables systematic gait design for high-dimensional biped robots in MATLAB. Analogous to FROST [7], TROPIC uses the hybrid zero dynamics (HZD) framework [15] to simultaneously generate gaits and feedback controllers via optimization of virtual constraints on the robot's configuration. This approach does not result in an optimal open-loop gait, but rather a closed-loop system that minimizes a given cost function along the periodic orbit. Formulating efficiently-solvable nonlinear programs (NLP) is critical in gait design for complex bipeds, and TROPIC employs the direct collocation method to leverage CasADi's unprecedented efficiency in generating large, but sparse NLP formulations.

Figure 2 depicts TROPIC's optimization process. Biped models are expressed as kinematic trees subject to contact classes (e.g., point, planar or spatial, with or without friction). The user can define constraints that are either physical or virtual. The friction cone (or pyramid) is an example of physical constraint automatically derived from the contact model. Virtual constraints used to synthesize feedback controllers as the output of HZD-based optimization can be omitted when generating open-loop gaits, but this does not guarantee existence of a stabilizing controller [15]. TROPIC uses MATLAB's interpreted language to enable seamless implementation of the biped model and user-defined constraints, and CasADi allows for efficient (in both time and memory) derivation and evaluation of the necessary symbolic expressions. While NLP solvers often require in-depth knowledge of optimization methods, CasADi provides a common interface to most off-the-shelf solvers.

In what follows, Section II describes the general form of biped models and gait design via optimization of virtual constraints in TROPIC. The optimization problem is automatically transcribed using direct collocation and the framework described in Section III. Section IV compares TROPIC's performance to FROST for examples of a 7-DOF planar biped and 12-DOF and 20-DOF spatial bipeds. Section V provides conclusions.

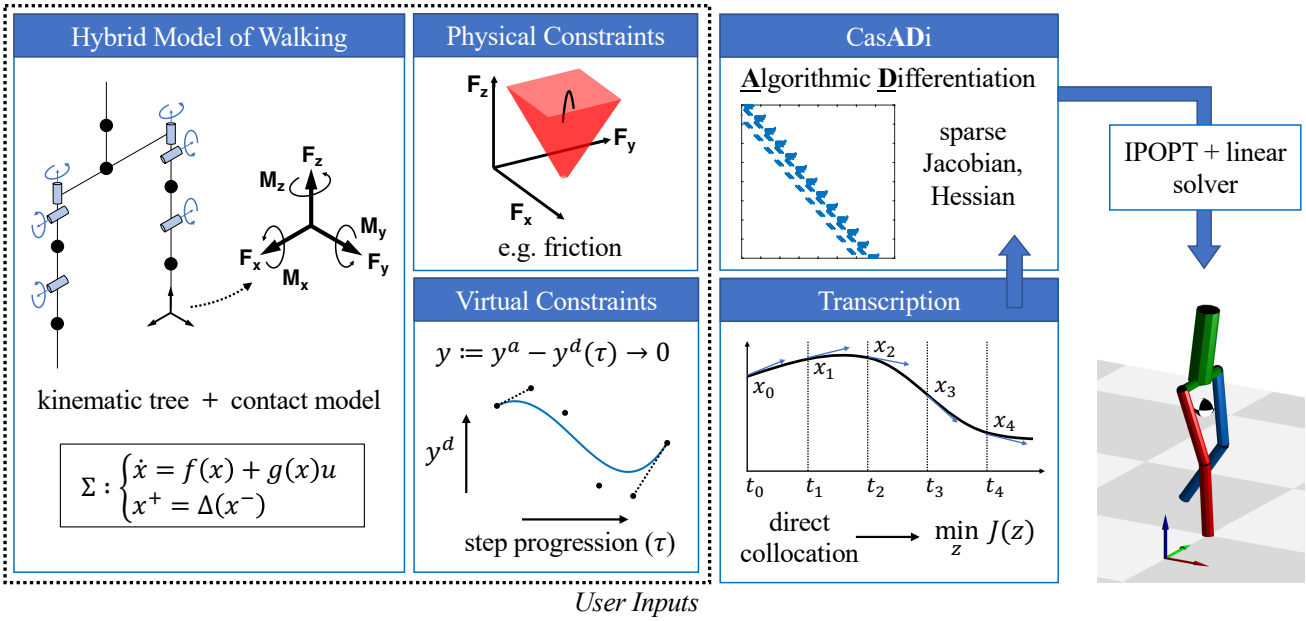


Fig. 2: Overview of gait optimization process in TROPIC

II. Dynamic Model of Walking

While TROPIC facilitates trajectory optimization of systems with or without impacts, this paper focuses on optimizing gaits for underactuated bipeds with hybrid dynamics and point contacts. The model in Fig. 2 is an example exhibiting alternating phases of single support and infinitesimally short ground impacts. Represented by a tree of rigid links, the model is left-right symmetric, so periodic gaits are sought by analyzing only a single step. With the legs terminating in point feet, there is no actuation at the end of the stance leg.

A. Continuous Single Support

Single support is modeled using continuous second-order ordinary differential equations,

$$H(q)\ddot{q} + C(q,\dot{q}) = Bu + J_c(q)^T F_c, \tag{1}$$

where $q \in \mathbb{R}^{n_d}$ represents the n_d generalized coordinates, $H(q) \in \mathbb{R}^{n_d \times n_d}$ is the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n_d}$ is a vector containing Coriolis and gravity terms, $B \in \mathbb{R}^{n_d \times n_a}$ is a selector matrix mapping n_a actuator torques to joint torques, $u \in \mathbb{R}^{n_a}$ is a vector of independent control inputs, $J_c(q) \in \mathbb{R}^{3 \times n_d}$ denotes the contact Jacobian, and $F_c \in \mathbb{R}^3$ represents the ground reaction forces. In single support, the stance foot is assumed not to slip. This physical contact with the ground introduces a holonomic constraint on the stance foot's Cartesian position in the inertial frame: ${}^0p_e$ is constant. The associated kinematic constraint is ${}^0v_e = J_c(q)\dot{q} = 0$, and the second derivative of the holonomic constraint

$$J_c(q)\ddot{q} + \dot{J}_c(q)\dot{q} = 0 \tag{2}$$

must be enforced and evaluated alongside Eq. (1) to determine the constrained continuous dynamics of single support.

B. Discrete Impact

Impacts (assumed purely inelastic) occur when the swing foot touches down with nonzero velocity. The hypersurface

$$\mathcal{S} = \left\{ (q,\dot{q}) \in \mathbb{R}^{n_d} \mid {}^0p_e(q) = 0,\ {}^0v_e(q,\dot{q}) < 0 \right\} \tag{3}$$

defines the limit of the continuous dynamics. While configuration is invariant through impact, the relabeling map $q^+ = R(q^-)$ swaps the right and left legs at each step, where q^- and q^+ represent the pre- and post-impact configuration states. The generalized velocities undergo a jump due to the instantaneous change in generalized momentum,

$$\dot{q}^+ = \Delta_q(q^-)\,\dot{q}^-, \tag{4}$$

where $\Delta_q(\cdot)$ is obtained by integrating Eqs. (1) and (2) over the duration of impact. The complete system with impulse effects may be rewritten in state-space form using coupled first-order differential equations,

$$\Sigma : \begin{cases} \dot{x} = f(x) + g(x)u, & x \notin \mathcal{S}, \\ x^+ = \Delta_x(x^-), & x \in \mathcal{S}, \end{cases} \tag{5}$$

where $x := (q\,;\dot{q}) \in \mathbb{R}^{2n_d}$.

C. Rigid Body Algorithms

TROPIC users build biped models from three rigid body and two joint types. The `slender-rod` represents most rigid links, whereas the `reaction-wheel` can be added to exert external torques through internal changes in angular momentum [16]. The `base` specifies the kinematic tree's base, and joints are either `prismatic` or `revolute`. The equations of motion are derived using spatial vector arithmetic with the Recursive-Newton-Euler Algorithm used

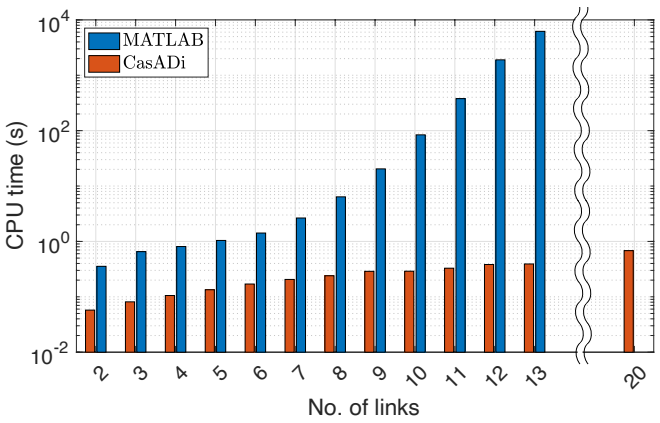


Fig. 3: CPU time spent to obtain $H(q)$ and $C(q, \dot{q})$ for an N-link pendulum using the same rigid body algorithms.

to compute $C(q, \dot{q})$ and the Composite-Rigid-Body Algorithm used to obtain the inertia matrix $H(q)$ [17]. CasADi features a sparse matrix representation of symbolic variables for which each symbolic expression is stored as a directed graph of primitive operations. Figure 3 compares CPU times to derive the equations of motion using CasADi symbolics and MATLAB's Symbolic Toolbox for a pendulum with increasing number of links on a Desktop computer with an Intel Core i7-3770 processor (3.4 GHz) and 16 GB of RAM. For 13 links, MATLAB required approximately 1.5 hours, whereas CasADi completed the task in 0.4 seconds. CasADi's derivation time increased to just 0.7 seconds for the 20-link case, showing no apparent limitations of the computer algebra system.

D. Virtual Constraints

Analogous to FROST, TROPIC employs HZD-based optimization to generate dynamic walking gaits using virtual constraints. While conventional optimization yields an optimal open-loop gait (state-control pair realization of Eq. (5)), HZD-based optimization generates a closed-loop system whose feedback controller minimizes a cost function along the periodic orbit. Virtual constraints are a special class of holonomic constraints enforced through feedback instead of physical connections. The controller outputs are the tracking errors between the actual $y^a(\cdot)$ and desired $y^d(\cdot)$ trajectories.

$$y(q, \alpha) = y^a(q) - y^d(\alpha, \tau), \tag{6}$$

where τ is a monotonically increasing phase variable and α is a set of scalar-valued parameters encoding the robot's desired configuration.

Feedback linearization is applied to obtain an input/output linear system.[1] The virtual constraints have relative degree 2, so the output must be differentiated twice before u appears explicitly. The feedback controller takes the common form

$$u = (L_g L_f y)^{-1}(v - L_f^2 y), \tag{7}$$

[1]The three biped models used in Section IV are underactuated and not full-state feedback linearizable since $\text{rank}(B) < n_d$.

where $L_f(\cdot)$ and $L_g(\cdot)$ represent the Lie derivative of $(\cdot)$ with respect to the drift and input vector fields, respectively. The control law

$$v = -\varepsilon^2 y(q, \alpha) - 2\varepsilon \dot{y}(q, \dot{q}, \alpha), \tag{8}$$

yields the linear output dynamics

$$\ddot{y}(q, \dot{q}, \ddot{q}, \alpha) = -\varepsilon^2 y(q, \alpha) - 2\varepsilon \dot{y}(q, \dot{q}, \alpha), \tag{9}$$

where $\varepsilon > 0$ is chosen to drive $y(q, \alpha)$ to 0 exponentially and restricts the system dynamics to evolve on the zero dynamics manifold $\mathcal{Z}$ during single support phases [18].

$$\mathcal{Z} = \left\{(q, \dot{q}) \in \mathbb{R}^{n_d} \mid y(q, \alpha) = 0,\ \dot{y}(q, \dot{q}, \alpha) = 0\right\} \tag{10}$$

Invariance of the zero dynamics through impact is further enforced by requiring that at the beginning of the step,

$$y(q, \alpha) = 0, \tag{11a}$$

$$\dot{y}(q, \dot{q}, \alpha) = 0. \tag{11b}$$

TROPIC expresses virtual constraints as Bézier polynomials.

$$y^d(\alpha, \tau) = \sum_{m=0}^{M} \frac{\alpha_m M!}{m!(M-m)!} \tau^m (1-\tau)^{M-m}, \tag{12}$$

where M is the user-selected order of the polynomial. This choice of virtual constraint offers both analytical and computational advantages since the hybrid invariance conditions in Eq. (11) can be derived, and therefore enforced, in closed-form (see [18], Corollary 6.1 for complete derivation).

III. Gait Optimization

A. Direct Collocation

Gait optimization problems are typically solved via direct shooting or collocation methods [19]. While generally simpler to model, shooting methods integrate the dynamics by *shooting* trajectories from starting points, so they require good initialization and often suffer from poor conditioning in complex systems. Conditioning becomes less of an issue in optimization problems with short horizons and fewer variables, which makes shooting methods well suited for model predictive control [20]. TROPIC employs direct collocation in which the trajectory is parameterized using discrete way-points that include the states and control inputs [21]. The main difference between the states and control inputs is that the states are differentiated variables in the dynamics equations, whereas the control variables only appear algebraically in the dynamics equations, resulting in a system of differential algebraic equations (DAE), which are distinct from ordinary differential equations (ODE) in that the Jacobian matrix of a DAE system is singular since its derivative with respect to some variables will be zero (*i.e.*, some equations are algebraic). The equations of motion must only be satisfied at the collocation points, resulting in a large but sparse Jacobian matrix that can be efficiently handled by large-scale NLP solvers.

TROPIC supports both trapezoidal and Hermite-Simpson collocation methods [21]. While the latter leads to more accurate solutions by approximating the system dynamics

using piecewise quadratic functions, the trapezoidal method is simpler and introduced herein for brevity. First, the continuous state variables $x(t)$, control inputs $u(t)$, and contact forces $F_c(t)$ are discretized using N_c collocation points.

$$\begin{aligned} t &\rightarrow [t^0, ..., t^k, ..., t^N], \\ x &\rightarrow [x^0, ..., x^k, ..., x^N], \\ u &\rightarrow [u^0, ..., u^k, ..., u^N], \\ F_c &\rightarrow [F_c^0, ..., F_c^k, ..., F_c^N], \end{aligned} \tag{13}$$

where k is the discretization index and $N = N_c - 1$ for the trapezoidal method. The system dynamics are then approximated between every pair of collocation points.

$$x^{k+1} = x^k + \frac{1}{2} l^k (\dot{x}^{k+1} + \dot{x}^k), \tag{14}$$

where $l^k = t^{k+1} - t^k$, and a new vector of decision variables is added at every collocation point,

$$z^k = (q^k; \dot{q}^k; \ddot{q}^k; u^k; F^k). \tag{15}$$

Obtaining an explicit representation of $\dot{x}^k$ in Eq. (14) typically requires inverting the inertia matrix, so it is not desirable. To avoid computing $f(x)$ and $g(x)$ symbolically in closed-form, TROPIC takes the approach of implicitly enforcing Eqs. (1)-(2) by introducing $\dot{x}^k := (\dot{q}^k; \ddot{q}^k)$ as a defect variable at each collocation point [22] and enforcing

$$H(q^k)\ddot{q}^k + C(q^k, \dot{q}^k) - Bu^k - J_c(q^k)^T F_c^k = 0, \tag{16}$$

$$J_c(q^k)\ddot{q}^k + \dot{J}_c(q^k)\dot{q}^k = 0. \tag{17}$$

The feedback law is also computationally expensive to enforce explicitly. On the zero dynamics manifold, however, the control u is uniquely determined by Eq. (7), so enforcing

$$\ddot{y}(q^k, \dot{q}^k, \ddot{q}^k, \alpha) + \varepsilon^2 y(q^k, \alpha) + 2\varepsilon \dot{y}(q^k, \dot{q}^k, \alpha) = 0, \tag{18}$$

at each collocation point calculates the feedback term accordingly. The invariance condition for the hybrid zero dynamics is particularly easy to enforce using collocation methods.

$$y(q^k, \alpha) = 0, \tag{19a}$$

$$\dot{y}(q^0, \dot{q}^0, \alpha) = 0. \tag{19b}$$

The dynamic equations assume no slip of the stance foot, so the contact forces must lie inside the friction cone.

$$\sqrt{(f_x^k)^2 + (f_y^k)^2} - \mu_s f_z^k < 0, \tag{20}$$

where $F_c := (f_x; f_y; f_z)$ and μ_s denotes the coefficient of static friction. TROPIC gives the user freedom to replace Eq. (20) with the more conservative linear friction pyramid

$$|f_x^k| < \frac{\mu_s}{\sqrt{2}} f_z^k \ , \qquad |f_y^k| < \frac{\mu_s}{\sqrt{2}} f_z^k \ . \tag{21}$$

To prevent flight phases, the normal component of the contact force must remain positive.

Foot touchdown is constrained to occur at the end of the discretized trajectory (*i.e.,* step), so the guard conditions of Eq. (4) must be enforced at the last collocation point,

$${}^0 p_e(q^N) = 0, \tag{22}$$

$${}^0 v_e(q^N, \dot{q}^N) < 0. \tag{23}$$

Upon impact, continuity of the position states is enforced using the relabeling map

$$q^0 - R(q^N) = 0. \tag{24}$$

The impact map for the velocities is added at the last collocation point to enforce periodicity.

$$H(q^0)(\dot{q}^0 - R(\dot{q}^N)) - J_c(q^0)^T \hat{F}_c = 0, \tag{25}$$

$$J_c(q^0)\dot{q}^0 = 0, \tag{26}$$

where $\hat{F}$ represents the integrated impulsive forces in units of N·s. A non-penetration holonomic constraint of the form

$$p_e(q^k) - p_e^d(\tau^k) \geq 0 \tag{27}$$

requires the distance to contact to be positive, ensuring sufficient swing foot ground clearance throughout the step.

B. NLP Formulation

Upon transcription of the gait optimization problem, the NLP has the general form

$$\begin{aligned} \min_{z} \quad & \mathcal{J}(z) \\ \text{subject to} \quad & \underline{z} \leq z \leq \overline{z}, \\ & w(z) \leq 0, \\ & h(z) = 0, \end{aligned} \tag{28}$$

where z is the set of decision variables bounded by $[\underline{z}, \overline{z}]$, $\mathcal{J}(\cdot)$ is the cost function to be minimized, and $h(\cdot)$ and $w(\cdot)$ represent the equality and inequality constraints, respectively. The Lagrangian corresponding to the constrained NLP is

$$\mathcal{L}(z, \mu, \lambda) = \mathcal{J}(z) + \mu^T w(z) + \lambda^T h(z), \tag{29}$$

where μ and λ are the Lagrange (or KKT) multipliers [23]. Obtaining symbolic expressions for the Jacobian and Hessian of Eq. (29) may prove critical to solve such problems efficiently. To derive the derivatives, TROPIC uses CasADi's implementation of AD. While conventional symbolic differentiation becomes intractable when computing the partial derivatives of a function with respect to many variables, AD uses the chain rule to break down the computation of the Jacobian into a sequence of atomic sub-operations. The Hessian is obtained by applying AD recursively.

The two most distinct features of CasADi's AD are a hybrid scheme of forward and reverse modes of AD and graph coloring. After breaking down the computation of the Jacobian, forward and reverse modes of AD provide two ways to compute the directional derivatives, and hybrid techniques often result in fewer sub-operations. The main idea behind graph coloring is to reconstruct the complete Jacobian using a minimal set of directional derivatives [24]. Trajectory optimization via direct collocation typically results in a large but sparse NLP, so CasADi's ability to recognize and store large sparsity patterns is critical for scaling to higher-dimensional bipeds.

IV. Results

A. Planar 7-DOF Biped Comparative Studies

TROPIC was used to optimize a 10-gait library with walking speeds from 0.15 to 1.05 m/s for RABBIT, a planar 5-link biped model with 7 DOF defined in [15]. RABBIT has 4 actuators, one for each hip and knee, so 4 time-dependent virtual constraints modulate their respective trajectories. Walking was parameterized using 25 collocation points and a uniform discretization grid, so $t^k = k\frac{T}{N}$, where T is the step period. The cost function is the integral of the squared torques over the step.

$$\mathcal{J} = \sum_{k=0}^{N} \sum_{i=1}^{4} \frac{l^k}{2} \left[(u_i^k)^2 + (u_i^{k+1})^2 \right] \tag{30}$$

when approximated using trapezoidal quadrature. This quantifies energetic efficiency, a critical aspect of bipedal gait design, because the input power of the motors in electrically actuated systems is proportional to the square of the armature current, which is itself linearly proportional to the torque output. The cost function penalizes large torques and tends to produce smooth solutions well-suited for walking robots.

TROPIC was compared to FROST first using FROST's formulation implemented in CasADi and then using TROPIC's formulation. The main difference is that FROST defines a new copy of the virtual constraint parameters α^k at each collocation point and enforces consistency throughout the trajectory by adding the constraint $\alpha^k = \alpha^{k+1}$. This results in a computationally efficient band structure in the Jacobian matrix of the constraints. In TROPIC, however, CasADi's graph coloring technique reconstructs the Jacobian using directed graphs of primitive variables and a minimal set of directional derivatives, which results in an efficient NLP without redundant variables.

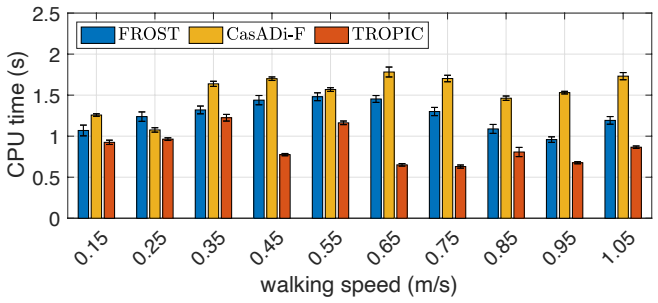
The initial guess, variable bounds, path constraints, and convergence tolerances were identical and selected to match FROST's default options. IPOPT was used to solve the resulting problems with the linear solver ma57 on the same processor used in Section II-C. This linear solver is well-suited for large optimization problems, but ma57 does not permit redistribution, so separate installation is required [25]. For each walking speed, FROST, CasADi's implementation of the FROST formulation, and TROPIC all converged to nearly identical gaits, with less than 0.5% difference in the optimal solutions. On average, FROST (which outperformed Drake while optimizing gaits for the ATLAS humanoid in less than 5 minutes [7]) generated gaits in 6.9 seconds, CasADi's implementation of it converged in 2.2 seconds, and TROPIC did so in 1.5 seconds, as summarized in Table I.

Figure 4 separates the duration of computation into time in IPOPT, which is primarily spent solving the KKT system at each iteration to determine the Newton step, and time evaluating the NLP functions, which includes the Jacobian and Hessian of the Lagrangian. The difference between "CasADi-F" and "TROPIC" in Fig. 4a highlights a major advantage provided by CasADi – it generates an efficient representation of the optimization problem regardless of how the user scatters variables and constraints. This stands in contrast with FROST in which they are set such that the Jacobian possesses a diagonal band structure.

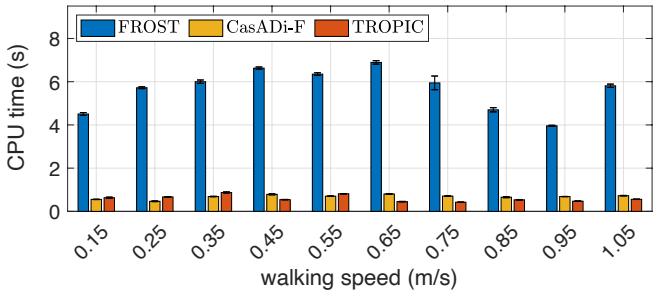
TABLE I: Average times for gait library optimization of planar 7-DOF biped model.

	FROST	CasADi-F	TROPIC
Variables	1355	1355	709
Constraints	1638	1638	891
Total CPU time (s)	6.9	2.2	1.5

As solving the KKT system requires comparable times for all three cases (within 1 second), the computational difference is mainly due to faster evaluation of the NLP functions in CasADi. Figure 4b shows that CasADi's performance in evaluating the NLP functions was unaffected when nearly doubling the number of variables and constraints. After transcription of the NLP, CasADi automatically encapsulates graph representations of the Jacobian and Hessian in function objects that are evaluated at run time using CasADi's virtual machines (VM) designed for high-speed and low overhead. Both FROST and TROPIC can ultimately achieve faster function evaluations by generating C code of the symbolic expressions, which is significantly faster than FROST's MATLAB functions and CasADi's VM. This approach is used in C-FROST [26], and the authors aim to add the same functionality in TROPIC.



(a) CPU time in IPOPT



(b) CPU time in NLP function evaluations

Fig. 4: Timings for gait optimization of planar 7-DOF biped model. "CasADi-F" denotes FROST formulation implemented in CasADi. Error bars indicate standard deviation for 10 optimizations run at each speed.

TABLE II: Optimization constraints & parameters.

Constraint description	Value
Actuator torques (N·m)	$\lvert u_i \rvert \leq 50, \quad i \in \{1, ..., 6\}$
Friction cone	$\mu = 0.6$
Step length (m)	[0.05, 0.35]
Step period (s)	[0.20, 1.00]
Mid-step foot clearance (m)	$p_e \geq 0.05$
Timing variable, τ (rad)	$\dot{\tau} > 0$
Control gain	ε = 10

B. Spatial 12-DOF Biped Implementation

TROPIC was employed to generate 0.50 m/s walking gaits for the 12-DOF spatial five-link biped model shown in Fig. 5. Each hip has two independently actuated DOFs, while each knee has just one. State-based virtual constraints on the actuated DOFs are posed as 5$^{\text{th}}$-order Bézier polynomials in τ, the angle between the ground and the line passing through the hip and stance foot in the sagittal plane. This mechanical phase variable has to increase monotonically over a step, so $\dot{\tau} > 0$ was added as a constraint. Table II lists the implemented constraints, including a coefficient of static friction of 0.6. Actuator limits of 50 N·m were selected based on the actual five-link biped robot ERNIE [27]. Joint limits on position and velocity were also enforced to generate realistic gaits. The cost function was again the integral of squared torques over the step. To test TROPIC's ability to solve larger gait optimization problems, the number of collocation points was increased from 25 to 50 and 100.

Without an initial guess, TROPIC automatically computes a *naive* seed by linearly interpolating between user-provided guesses of the robot's initial and final configurations [21]. The initial guess for the Bézier coefficients α's is systematically obtained using unconstrained least-squares fitting on the virtual constraints, and the control input is obtained from Eq. (7). Using this $naive$-seed approach and 25 collocation points, TROPIC required 6.1 seconds on average to converge to an optimal solution. Refining the discretization grid to 50 and 100 collocation points increased computational times to 18.5 seconds and 40.3 seconds, respectively, though the gaits were visually similar. The results are summarized in Table III, and a two-step snapshot of the resulting gait obtained with 100 collocation points is shown in Fig. 5.

TABLE III: Timings for optimization of 0.50-m/s walking gait for spatial 12-DOF biped model using different discretization mesh (CP = collocation points).

Process step	25 CP	50 CP	100 CP
Variables	1997	3947	7847
Constraints	2110	4185	8335
NLP transcription	0.8 s	1.4 s	2.3 s
IPOPT internal	3.3 s	10.2 s	28.8 s
NLP evaluation	2.8 s	8.3 s	11.5 s

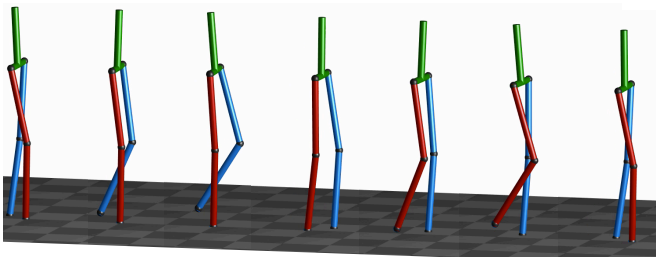


Fig. 5: Two-step snapshot of optimized 12-DOF biped gait. The model parameters are given in Table IV.

C. 20-DOF Biped Implementation

In this example, TROPIC was used to optimize a gait for a 20-DOF point-foot biped model with 14 actuators. This model is similar to the one shown in Fig. 5, with the addition of a head and upper limbs. In bipedal locomotion, arm motion can be used to improve balance and energy efficiency [28].

The friction and step constraints were kept identical from the previous section (Table II), and the cost function was the integral of squared torques over the step. This time, the Hermite-Simpson collocation method was employed to discretize the step with 25 finite elements, so there was a total of $2 \times 25 + 1 = 51$ collocation points. IPOPT and ma57 were again used to solve the resulting NLP. Using the *naive*-seed approach, TROPIC converged to a feasible solution in approximately 4 minutes. When using a relatively good initialization with results from the previous optimization, TROPIC needed less than 1 minute to find an optimized gait (22 seconds in IPOPT and 31 seconds in NLP function evaluations). A one-step snapshot of the resulting gait is shown in Figure 6, in which the biped model can be seen taking advantage of the extra DOFs in the upper limbs to swing the arms back and forth. TROPIC is able to optimize the full dynamics of the 20-DOF biped model for energetic efficiency, which produces a human-like gait with balancing arm motion.

TABLE IV: Geometric & mass parameters of 12-DOF and 20-DOF spatial bipeds used in Sections IV.B&C. The center of mass (CoM) location is measured from the proximal joint.

		Length (m)	CoM (m)	Mass (kg)
12-DOF Model	Lower leg	0.40	0.08	1
	Upper leg	0.40	0.08	1
	Pelvis	0.20	–	1
	Torso	0.25	0.0625	5
20-DOF Model	Lower leg	0.40	0.08	2.5
	Upper leg	0.40	0.08	3.5
	Pelvis	0.20	–	2
	Torso	0.45	0.1125	15
	Shoulder	0.35	–	1
	Head	0.10	0.05	2
	Upper arm	0.25	0.075	2.5
	Lower arm	0.30	0.12	2.5

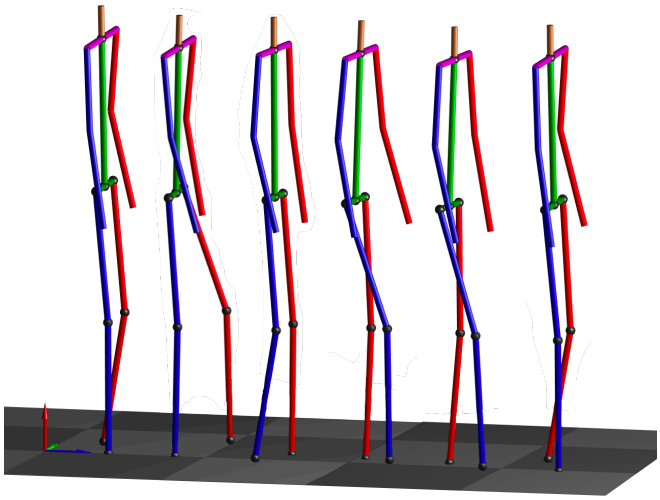


Fig. 6: Two-step snapshot of optimized 20-DOF biped gait. The total mass of the biped is 42 kg and model parameters are detailed in Table IV.

V. Conclusions

This paper introduced TROPIC, an open-source optimization package that allows systematic gait design for biped robots using MATLAB's interpreted language and CasADi's implementation of algorithmic differentiation. The approach, based on gait design via optimization of virtual constraints, generates a feedback controller for a given gait instead of a conventional open-loop state-control pair that can be hard to stabilize in practice. TROPIC's CasADi backbone enables fast and efficient transcription of the gait optimization problem, which becomes a large but sparse NLP using direct collocation methods. Comparative studies showed that TROPIC enables gait optimization 4 times faster than another state-of-the-art optimization package for a 7-DOF planar biped. The framework was also applied to 2 spatial bipeds: 12- and 20-DOF models, and results suggest that TROPIC's formulation is particularly well-suited to leverage CasADi's unprecedented efficiency into gait design for higher-dimensional biped robots.

Ongoing work aims to 1) generate C code expressions to further accelerate numerical evaluation and symbolic processing, 2) provide Universal Robot Description Format (URDF) parsing capabilities, and 3) use TROPIC for gait optimization of yet more complex legged systems, such as quadruped robots with multi-contact phases.

References

[1] T. McGeer, "Passive dynamic walking," *Int. J. Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.

[2] C. Chevallereau, J. W. Grizzle, and C. L. Shih, "Asymptotically stable walking of a five-link underactuated 3D bipedal robot," *IEEE Trans. Robotics*, vol. 25, no. 1, pp. 37–50, 2009.

[3] M. J. Powell and A. D. Ames, "Mechanics-based control of underactuated 3D robotic walking: Dynamic gait generation under torque constraints," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2016, pp. 555–560.

[4] Y. Liu, P. M. Wensing, J. P. Schmiedeler, and D. Orin, "Terrain-blind humanoid walking based on a 3D actuated dual-slip model," *IEEE Robotics & Automation Letters*, vol. 1, no. 2, pp. 1073–1080, 2016.

[5] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot Cassie." in *Proc. Robotics: Science and Systems*, 2018.

[6] R. Tedrake, "Drake: a planning, control, and analysis toolbox for nonlinear dynamical systems," *Available at: http://drake.mit.edu*, 2014.

[7] A. Hereid and A. D. Ames, "FROST*: Fast Robot Optimization and Simulation Toolkit," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2017, pp. 719–726.

[8] J. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[9] F. Magnusson and J. Åkesson, "Dynamic optimization in JModelica.org," *Processes*, vol. 3, no. 2, pp. 471–496, 2015.

[10] S. Lucia, A. Tătulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, pp. 51–62, 2017.

[11] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *Proc. European Control Conf.*, 2016, pp. 1586–1591.

[12] V. Look, "An optimal control toolbox for MATLAB based on CasADi," Ph.D. dissertation, Linköping University, 2016.

[13] T. Marcucci, M. Gabiccini, and A. Artoni, "A two-stage trajectory optimization strategy for articulated bodies with unscheduled contact sequences," *IEEE Robotics & Automation Letters*, vol. 2, no. 1, pp. 104–111, 2016.

[14] B. Katz, J. Di Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2019, pp. 6295–6301.

[15] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 42–56, 2003.

[16] T. L. Brown and J. P. Schmiedeler, "Reaction wheel actuation for improving planar biped walking efficiency," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1290–1297, 2016.

[17] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[18] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 2007.

[19] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[20] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[21] M. Kelly, "An introduction to trajectory optimization: how to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[22] J. Reher, E. A. Cousineau, A. Hereid, C. M. Hubicki, and A. D. Ames, "Realizing dynamic and efficient bipedal locomotion on the humanoid robot DURUS," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2016, pp. 1794–1801.

[23] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. SIAM, 2010.

[24] A. H. Gebremedhin, F. Manne, and A. Pothen, "What color is your Jacobian? Graph coloring for computing derivatives," *SIAM Review*, vol. 47, no. 4, pp. 629–705, 2005.

[25] HSL, "A collection of Fortran codes for large-scale scientific computation," *Available at http://www.hsl.rl.ac.uk*.

[26] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid trajectory optimization using C-FROST with illustration on a Cassie-series dynamic walking biped," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2019, pp. 4722–4729.

[27] M. Fevre, B. Goodwine, and J. P. Schmiedeler, "Terrain-blind walking of planar underactuated bipeds via velocity decomposition-enhanced control," *Int. J. Robotics Research*, vol. 38, no. 10–11, pp. 1307–1323, 2019.

[28] T. Kobayashi, K. Sekiyama, T. Aoyama, Y. Hasegawa, and T. Fukuda, "Selection of two arm-swing strategies for bipedal walking to enhance both stability and efficiency," *Advanced Robotics*, vol. 30, no. 6, pp. 386–401, 2016.