This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

A Springy Leg and a Double Backflip

Juan D. Gamba and Roy Featherstone

***Abstract*—This paper presents a simulation study of a planar monopod robot's motion search and control problems. The robot's passively spring-loaded prismatic joint in the lower body (between the foot and the shin) assists it in performing athletic motions. Systems of this kind can balance on a single point, hop, and produce complex movements with a suitable control strategy. We aim to investigate the application of direct orthogonal collocation (DOC) methods to nonlinear motion search problems for obtaining the continuity accuracy of using an ODE solver but exploiting the faster computation and convergence of collocation methods. Moreover, the presented controller allows the robot to get up from the ground, move to the ready position, and replicate the optimized motion (a double backflip) in a dynamic simulation.**

***Index Terms*—Underactuated Robots, Constrained Motion Planning, Integrated Planning and Control.**

I. Introduction

For acrobats and athletes, the success or failure of a maneuver depends on precise and accurate movements. In this sense, strategies for finding optimized ways of executing a task and approaches to implementing them in real systems are still an open problem in robotics. Launching into a hop without considering the takeoff velocities can cause head landings or crashes, as commented in [1]. Additionally, as the motion to perform is unknown and leads the system to an uncontrollable state (takeoff), finding the optimal amount of time the robot needs to achieve takeoff is nontrivial. [2] uses a sequential optimization approach for finding a launching motion for a vertical hop with a biped wheeled robot and concluded that by employing this strategy, it is necessary to manually adjust the motion duration t_{a_k} until finding the best instant, which is computationally inefficient and tedious. Analogous studies have been conducted to achieve similar motions. [3] presents a sequential motion planning method for generating somersaults on bipedal robots; the authors modified Cassie by adding a flywheel at the top of its pelvis to amplify its capability to control the momentum while flying. [4] uses a humanoid robot model to produce a jump of 16.4 cm height with a robot of 43.5 kg. And, [5] presents a similar work where a simple control algorithm enables a rigid-leg monopod robot to crouch, lift off, fly, and land during a single hop.

This paper extends and generalizes the application of the work presented in [5] for producing hops and balance with a monopod robot. This study uses a modified version of the dynamic software package developed in [6] to obtain an algebraic representation of the equations of motion for any robot applied to nonlinear trajectory optimization with casADI software [7] combined with an implicit integrator based on orthogonal collocation methods and Legendre-Gauss-Radau polynomials. A limitation of this approach is that it requires a continuous dynamical system, and as transitions between contact and no contact, or between slipping and not slipping, are discrete events that introduce discontinuities into the dynamics, it is necessary to split the full motion into a sequence of three phases (takeoff, flight, and landing), each having continuous dynamics.

After obtaining the optimized motion, we employ the balance theory [8] to design the proposed controller, which has demonstrated its effectiveness on physical systems [9], [10], [11]. Controllers based on this theory exhibit significantly faster and more accurate tracking of motion commands at the actuated joint than an LQR [12], nonlinear [13], and linear [14] controllers (e.g., see Fig. 12 of [8]). A similar performance is obtained in [15], [16], where the controller simultaneously maintains the balance, tracks a command signal, and suppresses the vibrations introduced by a passive spring using only one actuator. In contrast, motions like hopping require achieving certain dynamic conditions to takeoff rather than a specific joint configuration. This paper shifts the control problem to the center of mass (CoM) space following the commented balance theory to ensure the success of such motions.

The scope of this study goes beyond [5] and [2], [3] by introducing a strategy to successfully execute a double backflip using a planar spring-loaded monopod robot. The decision to study only a planar system rather than full 3D can be justified by observing that hopping is a nearly planar activity. It makes sense to design the robot so that the hopping movement is planar. The novel contributions of this paper are: (i) The robot employed is underactuated; we use only a single actuator to accomplish the entire sequence of movements (one less than Raibert's planar hopper [17]). (ii) The robot has actuated and spring-loaded joints. (iii) We present a highly dynamic movement requiring substantially different behavior patterns (balancing, launching, flying, landing), some of which rely on the spring to achieve high physical performance. (iv) We demonstrate the applicability of nonlinear multiphase trajectory search strategy with orthogonal collocation methods and Legendre-Gauss-Radau polynomials to simultaneously find complex motions, the best passive system parameters, and the optimal takeoff instant with relatively low computational cost, fast execution, and high accuracy. (v) Introducing a high-performance controller to track a given motion at the center of mass (CoM) level enabling the robot to accurately achieve

Manuscript received: February 2, 2023; Revised: April 3, 2023; Accepted May 19, 2023.

This paper was recommended for publication by Editor L. Pallottino upon evaluation of the Associate Editor and Reviewers' comments. (*Corresponding author: Juan D. Gamba*).

Juan D. Gamba is with Advanced Robotics and Dynamic Legged Systems Lab, Istituto Italiano di Tecnologia, Genoa, Italy. juan.gamba@iit.it

Roy Featherstone is with Advanced Robotics, Istituto Italiano di Tecnologia, Genoa, Italy. roy.featherstone@ieee.org

Digital Object Identifier (DOI): see top of this page.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

Link (i)	Mass (kg)	Length (m)	CoM (m)	Inertia at CoM (kgm^2)
1	0.1	0.2	0.1	0.0006
2	0.4	0.3	0.15	0.006
3	2	0.5	0.33	0.0800

TABLE I
LENGTH AND INERTIA PARAMETERS OF THE ROBOT SHOWN IN FIG.1.

the necessary momentum for performing a desired activity that requires a takeoff.

II. ROBOT MODEL

The springy-leg robot, shown in Fig.1, is a planar, three-link mechanism in which links 1, 2, and 3 are the foot, shin, and thigh, respectively. Table I shows the links' mass and length parameters. The symbols m_i, l_i, c_i, and I_i appearing below denote the mass, length, CoM, and rotational inertia about the CoM, respectively, of link i.

The joint variables vector is $q = [\bar{q}_t, q_t]^\top$, where $\bar{q}_t = [q_y, q_x]^\top$ and $q_t = [q_1, q_2, q_3]^\top$. q_x and q_y specify the position of the foot, and q_1, q_2 and q_3 specify the configuration of the robot. When all three joints in q_t are zero, the leg is vertical, the leg's length is $l_1 + l_2$, and the thigh is horizontal out to the right. The positive motion of a revolute joint i (q_1 and q_3) rotates link i counter-clockwise relative to link $i-1$, and the positive motion of joint 2 extends the leg (so the actual length of the leg is $l_1 + l_2 + q_2$). In Fig. 1, q_1 is positive and q_3 is negative. Joint 3 is actuated, and the torque at this joint is τ_3. The force at joint 2 comes from the spring and damper, and the torque at joint 1 is always zero.

The following dynamic equation governs the system

$$\begin{bmatrix} H_{yy} & H_{yx} & H_{y1} & H_{y2} & H_{y3} \\ H_{yx} & H_{xx} & H_{x1} & H_{x2} & H_{x3} \\ H_{y1} & H_{x1} & H_{11} & H_{12} & H_{13} \\ H_{y2} & H_{x2} & H_{12} & H_{22} & H_{23} \\ H_{y3} & H_{x3} & H_{13} & H_{23} & H_{33} \end{bmatrix} \begin{bmatrix} \ddot{q}_y \\ \ddot{q}_x \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} C_y \\ C_x \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} F_y \\ F_x \\ 0 \\ F_s \\ \tau_3 \end{bmatrix} \quad (1)$$

where H_{ij} are elements of the joint-space inertia matrix, $\ddot{q}_y$, $\ddot{q}_x$, $\ddot{q}_1$, $\ddot{q}_2$ and $\ddot{q}_3$ are the joint acceleration variables, C_y, C_x, C_1, C_2 and C_3 contain gravity and velocity terms, $F_s = -K_s q_2 - D_s \dot{q}_2$, K_s and D_s denote the stiffness and damping coefficients of the spring. F_y and F_x denote the ground contact forces along y and x axes, which are zero when the robot is flying and are calculated by the simulator in subsection V-C when the foot is in contact with the ground; for simplicity, these forces are neglected at the controller and trajectory search phases by imposing that q_y and q_x are fixed.

The values of K_s and D_s are not initially given but will be determined by the optimization process.

III. TRAJECTORY SEARCH

Optimization strategies have been extensively used by [18], [4], [19], [20] for solving motion planning problems. In this sense, the motion search exercise is solved by employing the system dynamics commented on in the previous section as the boundary value problem of a differential-algebraic equation (DAE). In this sense, the dynamic model explained in equation

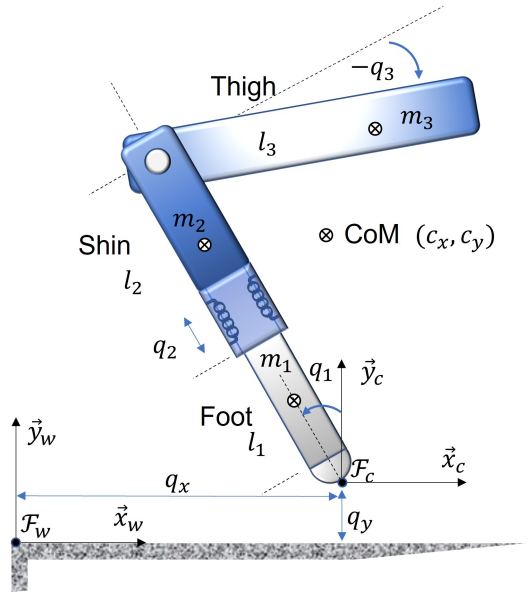

Fig. 1. Flying robot model. q_3 is negative in this configuration.

(1) is written as an ordinary system equation (ODE) with a set of constraints b:

$$\ddot{q} = H^{-1}(\tau - C) = f(q_t, \dot{q}_t, \tau_3) \qquad 0 = b(t, q_t, \dot{q}_t, \tau_3), \quad (2)$$

where $\ddot{q}$ is a nonlinear function of the robot's state $[q, \dot{q}]^\top$ and the control input τ_3. The main objective of this section is to find a control profile by evaluating the ODE system and satisfying the imposed set of boundaries b [21].

In this sense, the motion is divided in k time steps along $t \in (0, \infty] = [t_0, t_1, ..., t_k]$. Consequently, the problem is implemented as a nonlinear programming problem (NLP) to facilitate the addition of the following constraints:

- At every time step, the following nonlinear dynamic equality with a suitable integration method guarantees that the found motion matches the real system dynamics.

$$\dot{q}_{i+1} = \dot{q}_i + \int_{t_i}^{t_{i+1}} f(q_t, \dot{q}_t, \tau_3) dt, \; q_{i+1} = q_i + \int_{t_i}^{t_{i+1}} \dot{q}\, dt \quad (3)$$

- To bound the operating space of the joint positions due to the robot's kinematics, we implement mechanical inequalities $q_{min} \leq q \leq q_{max}$. In the presence of motors at the joint's axis, the joint's velocity can also be restricted to a maximum value $|\dot{q}| \leq \dot{q}_{max}$.
- Initial state equalities/inequalities ensure that the optimal solution starts at a specific value or under a specified region.
- Like initial equalities/inequalities, terminal state equalities/inequalities specify that the solution should finish in a specific value or under a particular region.

ODE systems can generally be solved using ***direct*** and ***indirect*** techniques. ***Direct*** approaches are usually used because they are simpler to set up and solve [22] than ***indirect*** methods, where it is necessary to construct the adjoint equations and their gradients to obtain more accurate metric for the solution [23]. ***Direct*** methods split into *sequential* and *simultaneous* methods. *Sequential* or *direct single shooting* methods solve

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

	DMS	DOC-LG	DOC-LGR
Number of Decision Variables	7505	25505	25505
Number of Constraints	6007	24007	24007
Number of Solver Iterations	45	166	43
Cost Function Value	1.92e+03	1.92e+03	1.92e+03
Execution Time (s)	58.91	70.98	20.16
Time per Iteration(s/it)	1.3091	0.4276	0.4688
Launching Instant (s)	0.8728	0.8728	0.8728

TABLE II
COMPARISON TABLE BETWEEN DMS, DOC WITH LEGENDRE-GAUSS POLYNOMIAL (DOC-LG), AND DOC WITH LEGENDRE-GAUSS-RADAU POLYNOMIAL (DOC-LGR).

the ODE model in an inner-loop (simulation), and the discretized control input τ_i is updated out of the simulation loop using an NLP solver. The outer-loop control update can use an analytical or numeric sensitivity analysis of the ODE model to calculate the objective function gradient concerning the control input [24]. These strategies are moderately easy to construct but require multiple integrations of the ODE system, which can be computationally costly and inefficient [2].

1) Direct Multiple Shooting (DMS): Between *sequential* and *simultaneous* methods, *direct multiple shooting* (DMS) approaches are capable of handling unstable DAE systems (not guaranteed by sequential schemes [25]). The method discretizes the control policy and states in k time steps and satisfies the system's dynamics by imposing the abovementioned constraints and integrating the system dynamics over each step i. The integration accuracy of equation (3) will depend on the integration method chosen. More straightforward integration techniques like Euler clearly demand less computational effort than more complex methods like Runge-Kutta or a DAE solver by compromising the solution's accuracy [24].

2) Collocation Methods: Direct Collocation Methods (DCMs), also known as *full-simultaneous* methods, approximate the continuous functions of the optimization problem in polynomials. This approach substitutes the integration method used in DMS by algebraic equality constraints enforced at the collocation points, which are distributed according to a first (direct collocation) or high (orthogonal collocation, DOC) order polynomial between the time steps t_{i-1} and t_i. In other words, it uses an implicit integrator that satisfies the system's dynamics at the collocation points instead of an explicit integrator. Trapezoidal or Hermite-Simpsons methods with relatively low-order polynomials are often used to implement the implicit integrator. As expected, the extra collocation constraints at every time step i increase the NLP problem dimension and sparsity, commonly exploited by Modern NLP, to reach solutions faster. Generally, collocation methods with higher-order polynomials, such as Legendre-Gauss-Radau, Legendre-Gauss, etc., can achieve higher accuracy than lower-order methods, such as linear and cubic splines, Chebyshev polynomials, etc. By considering the correct sets of orthogonal polynomials, it is possible to increase the order of accuracy of the collocation scheme (the result is still an approximation to the ODE solution) [24]. A more detailed explanation of orthogonal polynomials is in [26], [27].

3) Performance Comparison: Here we compare the DMS and DOC methods by producing a leap of $0.5\ \mathrm{m}$ with a launching velocity $v_0 = 2.0657\ \mathrm{m/s}$ between the CoM and the ground at $\theta = 0.5916\ \mathrm{rads}$ with the acrobot robot used in [5] using a Lenovo laptop with a Core i7-6500U CPU @ 2.5GHz and eight gigabytes of RAM. The DMS and DOC methods use an explicit and implicit 4th-order Runge-Kutta integrator. The orthogonal polynomials Legendre-Gauss (LG) and Legendre-Gauss-Radau (LGR) with three collocation points were used with the DOC method. The motion was implemented using CasADI and MATLAB and then solved using the IPOPT filter [28] and the default MA27 solver.

Table II shows how the number of decision variables in the DOC methods increased more than three times, and the number of constraints increased almost four times compared with the DMS approach. As expected, using an implicit solver increases the NLP problem size in DOC strategies. At the end of the optimization, the three approaches found the same launching motion and minimized cost at the end of the optimization. The DMS strategy took less time to find the optimal solution than DOC-LG, but the DOC-LG took three times more iterations than the DMS approach. On average, each DOC-LG iteration took three times less than the DMS iteration. The DOC-LGR approach stands out as the fastest method due to its superior stability properties compared to DOC-LG in applications of this nature. It efficiently solves the problem with a reduced number of iterations, resulting in faster convergence [24], [29].

IV. MULTI-PHASE OPTIMIZATION

Our strategy starts by splitting the entire motion into three phases (launch, flight, and landing). Then, we formulate a multiphase-optimization problem by concatenating the three nonlinear programming (NLP) problems (one for each stage) to obtain the entire motion profile for a double backflip with a spring-loaded monopod robot.

The three phases were solved using the *direct orthogonal collocation method* with Legendre-Gauss-Radau polynomials and seven collocation points. In this sense, the problem is formulated in each stage by specifying the state variables that describe the system's continuity during the motion. Then, the control variables are the parameters that need to be optimized for solving the motion (actuation profile $\tau_3(t)$, spring stiffness K_s, and damping D_s coefficients). The total task duration to perform the takeoff (t_{a_k}) and flight (t_{b_k}) are also added as control variables. The task steps k is fixed in this sense, but the sample times $\frac{t_{a_k}}{k}, \frac{t_{b_k}}{k}$ can vary with the phases duration t_{a_k} and t_{b_k}.

All the NLP problems were implemented using Matlab with the software package CasADI [7].

Launching and landing phases minimize the following objective function

$$J(t) = \int_0^{t_k} \tau_3(t)^2\, dt. \tag{4}$$

A. Launch Phase

The robot configuration when the robot takes off the ground is named the launching state. This state correlates the CoM's linear velocity to the coordinate frame and the centroidal

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

angular momentum ${}^cL$, which describes the angular momentum of the robot about the CoM. The desired takeoff linear velocities along the x and y can be obtained using parabolic-based physics equations for executing a leap; calculating the centroidal angular momentum is more complex because it is necessary to consider the robot's motion during the flight and the desired landing configuration. When the robot is flying, based on the conservation of momentum law, the robot can only modify its rotational velocity by varying the mass distribution about the CoM (given that the rotational momentum is constant during this phase). In this context, a successful landing depends on the launch instant and the change of centroidal rotational velocity during the flight.

1) Formulation: This motion was implemented using $a_k = 600$ steps, $T_a = [0, t_1, ..., t_{a_k}]$. It starts with the robot balancing on the ground with zero velocity and finishes when it's about to fly. The joint accelerations are obtained by solving equation (1), assuming that the foot is attached to the ground via a revolute joint, q_x and q_y are zero during the whole motion.

subject to:

$$0.2 \text{ sec} \leq t_{a_k} \leq 3 \text{ sec}, \tag{5}$$

$$1 \text{ Nm}^{-1} \leq K_s \leq 5{\times}10^4 \text{ Nm}^{-1}, \tag{6}$$

$$0 \text{ Nsm}^{-1} \leq D_s \leq 5{\times}10^4 \text{ Nsm}^{-1}, \tag{7}$$

initial constraints:

$$c_x(0) = \dot{c}_x(0) = \dot{c}_y(0) = 0, \tag{8}$$

$$F_s(0) = C_2(0), \tag{9}$$

loop constraints:

$$q_{min} \leq q_t(i) \leq q_{max}, \tag{10}$$

$$|\tau_3(i)| \leq 50 \text{ Nm}, \quad |F_x(i)| \leq 15 \text{ N}, \tag{11}$$

$$0 \text{ N} \leq F_y(i) \leq 350 \text{ N}, \tag{12}$$

$$\dot{q}_t(i+1) = \int_{t_{a(i)}}^{t_a(i+1)} \ddot{q}_t(t)dt, \quad q_t(i+1) = \int_{t_a(i)}^{t_a(i+1)} \dot{q}_t(t)dt, \tag{13}$$

terminal constraints:

$$1 \text{ ms}^{-1} \leq \dot{c}_y(a_k) \leq 10 \text{ ms}^{-1}, \quad F_y(a_k) = 0, \tag{14}$$

The spring parameters are added to the NLP problem in equations (6) and (7). The robot's initial conditions are in equations (8) and (9), where the CoM position c along the x axis, c_x, and the CoM's velocity $\dot{c}$ along the y and x axes, $\dot{c}_y$ and $\dot{c}_x$ have to be zero, and q_2 in its rest position. For simplicity, the loop constraints are only imposed at every step k and not at the collocation points. The joints position q are bounded (eq. (10)) based on the physical limits of the robot, where $q_{min} = [0, -0.2, -\pi/3]^\top$ and $q_{max} = [4\pi/9, 0, \pi/3]^\top$ in $\left[\text{rad} \quad \text{m} \quad \text{rad}\right]^\top$. The joint velocities $\dot{q}_i$ are assumed to be unbounded. (11) constrains the actuation torque τ_3. Forces F_x and F_y are constrained in equation (12) to ensure that the robot does not slip and not requires an unrealistic vertical force to takeoff. The system dynamics commented in equation (1) and the system's continuity are imposed as a constraint in (13). Finally, the terminal condition is to have a vertical velocity $\dot{c}_y$ bigger than $1\,\text{m/s}$ and zero vertical force at the ground contact point (F_y).

B. Flight Phase

The flight phase is implemented in $b_k = 200$ steps, $T_b = [t_{a_k}, t_{a_k+1}, ..., t_{b_k}]$ and starts when the robot loses contact with the ground until the touchdown. In this phase, the robot has no contact with the ground and has a mobile base rather than a fixed one. Moreover, during this phase, the robot can only modify its rotational velocity by changing its inertia (according to the conservation of momentum law). Like athletes, the robot should extend its leg at the takeoff and the touchdown instants and fold itself during the flight to achieve the necessary rotational velocity to perform the double backflip. The task is initialized with the last state of the takeoff phase $q = [0, 0, q_t(a_k)]$ and $\dot{q} = [0, 0, \dot{q}_t(a_k)]$.

$$J(t) = \int_0^T \tau_3(t)^2 + p_E(t)^2 \, dt. \tag{15}$$

subject to:

$$0.3 \text{ s} \leq t_{b_k} - t_{a_k} \leq 1.5 \text{ s}, \tag{16}$$

loop constraints:

$$q_{min} \leq q(i) \leq q_{max}, \tag{17}$$

$$|\tau_3(i)| \leq 50 \text{ Nm}, \tag{18}$$

$$\dot{q}(i+1) = \int_{t_b(i)}^{t_b(i+1)} \ddot{q}(t)dt, \quad q(i+1) = \int_{t_b(i)}^{t_b(i+1)} \dot{q}(t)dt, \tag{19}$$

terminal constraints:

$$-10 \text{ ms}^{-1} \leq \dot{c}_y(b_k) \leq 0 \text{ ms}^{-1}, \tag{20}$$

$$q_y(b_k) = 0 \text{ m}, \tag{21}$$

$$\pi/6 + 4\pi \text{ rad} \leq q_1(b_k) \leq 4\pi/9 + 4\pi \text{ rad}, \tag{22}$$

The objective function (15) is similar to (4) with an additional term p_E, which denotes the gravitational potential energy of the robot to obtain the motion with the smallest height. The flight duration is constrained in equation (16); in the loop constraints, the joints position q are delimited in equation (17), where $q_{min} = [-2, 0, -\pi/2, -0.2, -\pi/3]^\top$ and $q_{max} = [2, 15, \pi/2 + 4\pi, 0.2, \pi/3]^\top$ in $\left[\text{m, m, rad, m, rad}\right]^\top$, and the joints velocities $\dot{q}$ are assumed to be unbounded. The actuation torque τ_3 is bounded in equation (18). The system dynamics commented in equation (1) and the system's continuity are imposed as a constraint in (19). The terminal condition at the touchdown (eq.(21)) is to have a negative vertical velocity $\dot{c}_y$ (eq.(20)) and to have done a double backflip (eq.(22)).

C. Landing Phase

At this phase, the robot needs to go back to balance after the touchdown, requiring the minimum effort (eq.(4)). The task is implemented in $c_k = 300$ steps, $T_c = [t_{b_k}, t_{c_1}, ..., t_{c_k}]$. This NLP problem uses the same loop constraints imposed on the launch phase, from equation (10) to (13), with a modification at the bounds of joint one $4\pi \text{ rad} \leq q_1 \leq 4\pi/9 + 4\pi \text{ rad}$.

The phase starts at the touchdown, where we assume a plastic collision between the ground and the foot, the momentum

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

generated by $\dot{q}_x$ and $\dot{q}_y$ is mapped to the robot joints q_t as:

$$\dot{\tilde{q}}_t(t_{b_k}) = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{22} & H_{23} \\ H_{13} & H_{23} & H_{33} \end{bmatrix}^{-1} \begin{bmatrix} H_{y1} & H_{x1} \\ H_{y2} & H_{x2} \\ H_{y3} & H_{x3} \end{bmatrix} \begin{bmatrix} \dot{q}_y(t_{b_k}) \\ \dot{q}_x(t_{b_k}) \end{bmatrix} \quad (23)$$

then, the contribution $\dot{\tilde{q}}_t(b_k)$ due to $\dot{q}_x$ and $\dot{q}_y$ is added to the actual velocity of the joints $\dot{q}_t = \dot{\tilde{q}}_t(b_k) + \dot{q}_t(b_k)$.

subject to:

$$0.2\,\mathrm{s} \leq t_{c_k} - t_{b_k} \leq 3.5\,\mathrm{s}, \quad (24)$$

terminal constraints:

$$|\dot{q}_t(c_k)| = 0, \quad (25)$$

$$|c_x(c_k)| \leq 1\,\mathrm{mm}, \quad (26)$$

The task ensures that motions found in the previous phases are a solution for a smooth landing. In this sense, this NLP problem forces the optimizer to find the optimal spring parameters to minimize the impact force within the imposed mechanical limits (a spring compression of less than l_1).

D. Optimization Results

The full motion was solved using the IPOPT filter and MA86 solver from the Harwell Subroutine Library [30], which substantially reduces the execution time by solving the problem using multiple cores compared with the default solver MA27.

We started by finding a feasible trajectory for the takeoff and flight phases satisfying the imposed constraints. Then, we added the landing phase to the NLP problem to obtain the complete solution.

The found takeoff motion lasts $2.786\mathrm{s}$, the flight $0.821\,\mathrm{sec}$, and the landing $0.585\,\mathrm{sec}$. During the motion, the robot starts at a crouch configuration $q_t = [0.204\,\mathrm{rad}, -0.044\,\mathrm{m}, -1.046\,\mathrm{rad}]$ with zero velocity, then it tips itself backward (negative x direction) and forwards to launch into a double back-flip with a CoM position $c = [-0.013, 0.62]$ in m and a velocity of $\dot{c} = [0.677, 3.766]$ in ms^{-1} and a centroidal angular momentum of ${}^{c}L = 1.568\,\mathrm{kgm^2s^{-1}}$, during the flight the robot's CoM reaches $c_y = 1.4668\,\mathrm{m}$ height. The robot lands with a CoM position $c = [0.043, 0.41]$ in m and a velocity of $\dot{c} = [0.677, -4.288]$ in ms^{-1} and the same centroidal angular momentum, and stabilizes itself in a balanced configuration with $q_3 = -0.522$. At the touchdown, the CoM had a lower height than the takeoff, which explains the small increment in the magnitude of $\dot{c}_y$.

We have noticed that the spring parameters play a significant role in the landing phase for reducing the impact force generated after the touchdown and minimizing the required actuation effort to bring the robot back to balance. The spring-loaded monopod robot behaves similarly to a variable spring system, and it isn't easy to obtain its parameters analytically, as presented in [31]. The optimal spring parameters found with the optimization were $K_s = 519.046\,\mathrm{Nm^{-1}}$ and $D_s = 33.755\,\mathrm{Nsm^{-1}}$.

The optimal spring parameters also depend on the actuator's τ_3 ability to produce fast torque changes for efficiently storing and releasing the elastic energy produced by a given movement or impact while maintaining the robot balance.

V. Trajectory Tracking

The controller employed to track the optimized launching motion is based on the balance theory introduced in [15]. The controller is modified to control the CoM velocity $\dot{c}_x$ along the x-axis by driving ${}^{b}\ddot{L}$ through a tracking task.

A. Balance Theory

This section briefly introduces the analysis presented in [15]. The strategy assumes that the foot neither slips nor loses contact with the ground ($q_y = q_x = 0$) and computes a linearized model of the robot to obtain a virtual output ${}^{b}\dddot{L} = -mg\ddot{c}_x$ that later is transformed in τ_3 using the system dynamics introduced in (1).

The states vector of the linearized model is defined as $[{}^{b}\ddot{L}, {}^{b}\dot{L}, {}^{b}L, q_3]$, where ${}^{b}L = H_{11}\,\dot{q}_1 + H_{12}\,\dot{q}_2 + H_{13}\,\dot{q}_3$ is the angular momentum of the whole robot about the support point, which is proved in the appendix of [8]. ${}^{b}\dot{L} = -m\,g\,c_x$ is the moment of gravity about the support, where m is the total mass of the robot, g is the acceleration due to gravity (a positive number), and c_x is the x coordinate of the robot's CoM.

By assuming that q_y and q_x are zero during the whole task, it is possible to link the joint-space dynamics with the motion of the CoM, which leads us to:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{g\,D} \begin{bmatrix} -g\,H_{x3} & -H_{13} \\ g\,H_{x1} & H_{11} \end{bmatrix} \begin{bmatrix} {}^{b}L \\ {}^{b}\ddot{L} \end{bmatrix} \quad (27)$$

where $D = H_{x1}\,H_{13} - H_{11}\,H_{x3}$ assuming that the matrix is invertible (which it will be if the robot is physically capable of balancing [8]). Consequently $\dot{q}_3$ can be expressed as

$$\dot{q}_3 = Y_1\,{}^{b}L + Y_2\,{}^{b}\ddot{L} \quad (28)$$

where

$$Y_1 = \frac{H_{x1}}{D}, \qquad Y_2 = \frac{H_{11}}{g\,D} \quad (29)$$

Y_1 and Y_2 vary with configuration and can be expressed as simple functions of two physical properties of the mechanism: its time constant of toppling, T_{c}, which measures how quickly the robot falls if the controller does nothing, and its velocity gain [32], which measures the effect on the center of mass (CoM) velocity of a unit change in the velocity of the actuated joint.

B. Launch Controller

The system introduced in [15] is modified by controlling a motion (${}^{b}\ddot{L}_c$) in the CoM space instead of the actuated joint q_3 space, given that we are interested in controlling the robot's CoM to takeoff. After modifying the system's plant, the open-loop transfer function is

$$y(s) = \frac{1}{s}\,u(s). \quad (30)$$

In this form, the system does not present any zeros in the left semi-plane as commented in [8]. The system behaves like a simple integrator, and to ensure that the tracking error ${}^{b}e_L = {}^{b}$

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

$\dddot{L}_c - {}^b\dddot{L}$ goes to zero during the takeoff motion, the control law u is computed as:

$$ {}^b\dddot{L} := u = -k_{dd}({}^b\ddot{L} - {}^b\ddot{L}_c) + {}^b\dddot{L}_c, \tag{31}$$

where ${}^b\ddot{L}_c$ denotes the desired signal to be tracked, and as the launching motion is known, it is possible to obtain ${}^b\dddot{L}_c$ by differentiating ${}^b\ddot{L}_c$. Then, the closed-loop system looks like this:

$$\begin{bmatrix} {}^b\dddot{L} \\ {}^b\ddot{L} \\ {}^b\dot{L} \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} -k_{dd} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2(q) & 0 & Y_1(q) & 0 \end{bmatrix} \begin{bmatrix} {}^b\ddot{L} \\ {}^b\dot{L} \\ {}^bL \\ q_3 \end{bmatrix} + \begin{bmatrix} k_{dd}\,{}^b\ddot{L}_c + {}^b\dddot{L}_c \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{32}$$

and the transfer function of the closed-loop system is:

$$ {}^b\ddot{L}(s) = \frac{k_{dd}(1 + s/k_{dd})}{k_{dd} + s}\,{}^b\ddot{L}_c = {}^b\ddot{L}_c \tag{33}$$

which demonstrates that by only imposing a positive gain k_{dd} it is possible to drive ${}^b\ddot{L} \to {}^b\ddot{L}_c$.

To recover the robot from the ground and at the touchdown, the control law u was substituted by:

$$ {}^b\dddot{L} := k_{dd}\ddot{L} + k_d\dot{L} + k_L L + k_q q_3\,, \tag{34}$$

The feedback gains are obtained via pole placement as

$$\begin{aligned} k_{dd} &= -a_3 \qquad & k_d &= -a_2 + a_0 Y_2/Y_1 \\ k_L &= -a_1 \qquad & k_q &= -a_0/Y_1\,, \end{aligned} \tag{35}$$

where

$$\begin{aligned} a_0 &= \lambda_1\lambda_2\lambda_3\lambda_4 \\ a_1 &= -\lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_2\lambda_4 - \lambda_1\lambda_3\lambda_4 - \lambda_2\lambda_3\lambda_4 \\ a_2 &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 + \lambda_2\lambda_4 + \lambda_3\lambda_4 \\ a_3 &= -\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4 \end{aligned} \tag{36}$$

and $\lambda_1, \ldots, \lambda_4$ are the chosen values of the poles [33].

C. Simulation

In this simulation, we use the *ode23t* solver from MATLAB with relative tolerance set to 10^{-6}, and other parameters are chosen at their default values. Although the controller and launch optimization assume that the foot never loses contact nor slips with the ground, in the simulation, we have included the ground-contact model described in [34]. Contact forces acting on the foot in the normal and tangent directions are

$$\begin{aligned} F_y &= \max(0, K_n z^{3/2} + D_n z^{1/2}\dot{z}), \\ F_x &= \mathrm{clip}(K_t z^{1/2} u + D_t z^{1/2}\dot{u}, -\mu F_y, \mu F_y), \end{aligned} \tag{37}$$

where z and u are the ground compression and shear deformation, μ is the coefficient of friction, K_n and D_n are the normal, and K_t and D_t are the tangential stiffness and damping coefficients. The function $\mathrm{clip}(a, b, c)$ returns the value of a clipped to b and c. The parameter values used in the simulations are

$$\begin{aligned} K_t &= 12.7 \times 10^6 \quad D_t = 3.1 \times 10^5 \quad \mu = 1 \\ K_n &= 8.5 \times 10^6 \quad D_n = 3.1 \times 10^5 \end{aligned} \tag{38}$$

which are consistent with a hard floor and a high-friction hard rubber foot.

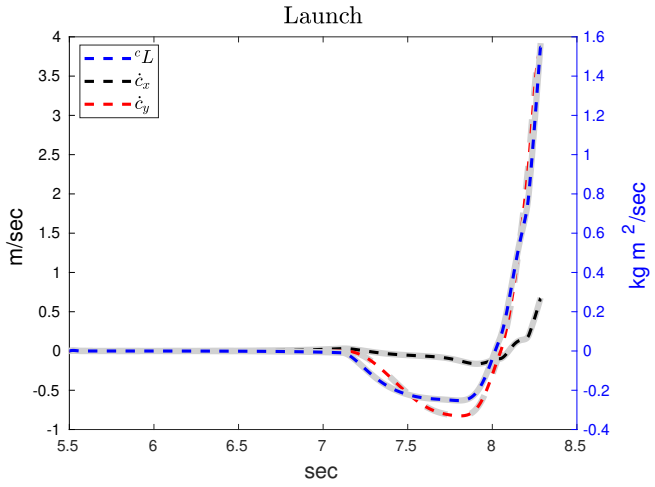

Fig. 2. Evolution of the robot's CoM's velocity $\dot{c}$, and centroidal angular momentum ${}^cL$ before the takeoff. The left-side scale applies to $\dot{c}_x$ and $\dot{c}_y$ and the right-side scale applies to ${}^cL$.

In the simulation, the springy-leg robot starts in an upright position laying on the ground $q_t = [-\pi/2\,\mathrm{rad}, 0\,\mathrm{m}, \pi/2\,\mathrm{rad}]$, it recovers to a balanced position by using a PID controller to move the actuated joint q_3 to -1.451 rad and uses the balance controller introduced in [15] with all the poles at $\lambda_i = -1/T_c\,\mathrm{rad\,s^{-1}}$ to balance the robot and get the crouch launching configuration with zero velocity. Then, the gain $k_{dd} = 60$ is set for tracking the takeoff motion $\dddot{L}_c$ with the control law introduced in equation (31). During the flight, the following controller is used

$$\tau_3 = C_3 + H_{33}\,(k_p\,e + k_d\,\dot{e}), \tag{39}$$

where C_3 and H_{33} were obtained from (1), $e = q_c - q_3$, q_c denotes the desired joint position, and k_p and k_d are positive gains. Once the robot touches the ground, the balance controller [15] is used to bring it back to balance with the poles at $\lambda_{1..3} = -10$ and $\lambda_4 = -1/T_c$ without following any trajectory. The whole motion can be seen in the accompanying video. The sequence of movements was obtained by employing the optimization strategy described in section IV. The system's evolution is presented from figures 2 to 7 in colored lines; the pale gray lines denote the optimized solution found by the optimizer, and all other lines show the simulation results.

In the figures, it is possible to observe that the strategy can accurately track and reproduce the motion obtained from the optimizer during the launch and flight phases, which corroborates the precision of the orthogonal collocation method employed during the optimization and that the controller is suitable for performing this kind of motion. We tried to execute the launch motion directly using the torque profile obtained from the optimization without any controller. However, the robot failed to achieve the liftoff by falling out of balance. The torque profile from the optimizer had an average difference of %4.37 compared with the torque profile obtained using the proposed controller.

The launch motion starts at $t = 5.5$s after successfully recovering from the ground and achieving the ready configuration to takeoff. Figure 2 shows the tracking results during the takeoff motion. Figure 3 shows a performance comparison

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

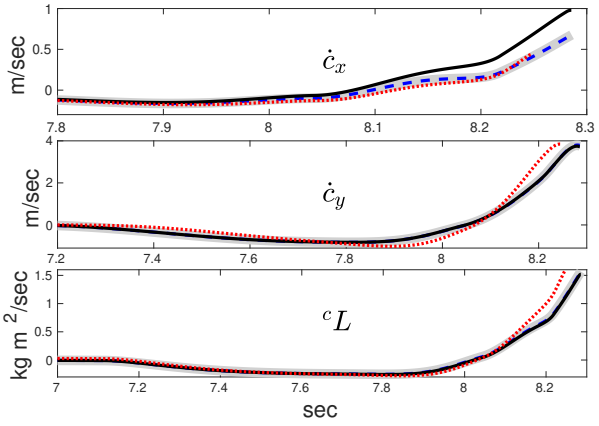

Fig. 3. Performance comparison between the proposed controller(blue), a PD controller (black) and Azad's controller (red) [5].

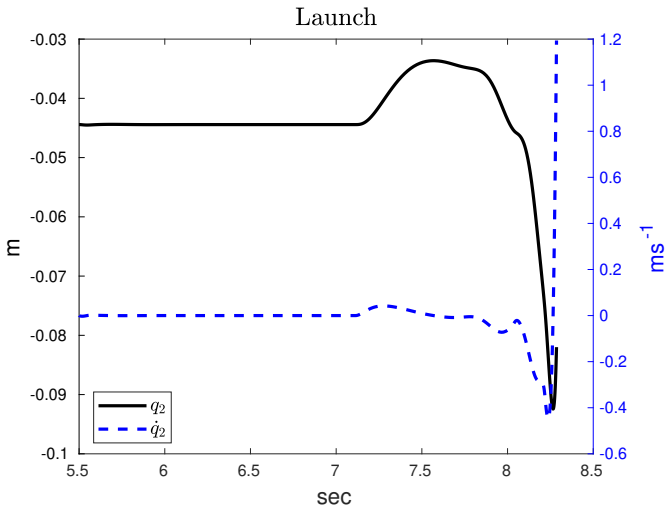

Fig. 4. Evolution of the spring-loaded joint position q_2 and velocity $\dot{q}_2$ before the takeoff. The left-side scale applies to q_2, and the right-side scale applies to $\dot{q}_2$.

between different controllers; the proposed controller in blue had an average error of 1% in achieving the desired launch found by the optimizer, the PD controller (eq. (39)) had a 16.33% and the controller presented in [5] obtained 12.76%. Figure 4 shows the evolution of the spring, where it stores a maximum of 2.218 J just before the robot leaves the ground. At the end of this phase, the controller had a tracking error of 1.386% at $\dot{c}_x$, 0.398% at $\dot{c}_y$ and 1.206% at ^cL related to the takeoff instant obtained from the optimization.

At $t = 8.286$ s, the robot is flying and can only modify the leg's position q_3 and velocity $\dot{q}_3$. Figure 5 shows the evolution of joint q_3 during the flight and the position of the CoM along the x and y axes related to the takeoff point.

At $t = 9.101$ s, the robot touches the ground after performing the double backflip. At the end of this phase, the controller had a tracking error of 0.38% at $\dot{c}_x$, 0.146% at $\dot{c}_y$ and 1.163% at ^cL related to the touch-down instant obtained from the optimization. Figure 6 shows the evolution of the CoM's position, velocity, angular momentum related to the contact point, and the q_3 position. Figure 7 shows the evolution of the spring-loaded joint q_2 at the impact. The spring absorbed

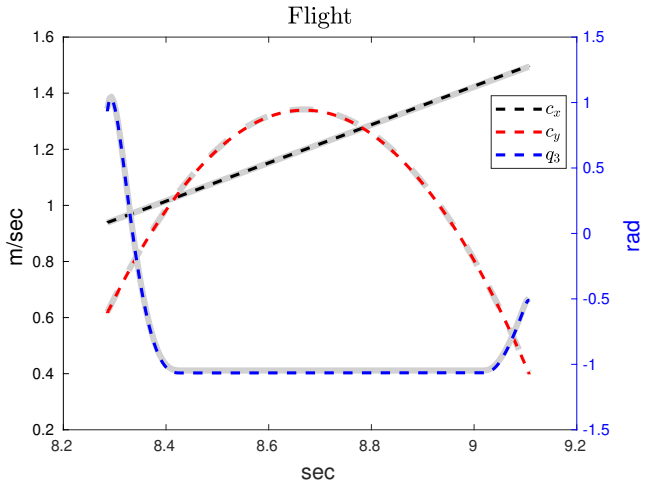

Fig. 5. Motion of the robot's CoM and actuated joint q_3 during the flight. The left-side scale applies to c_x and c_y and the right-side scale applies to q_3.

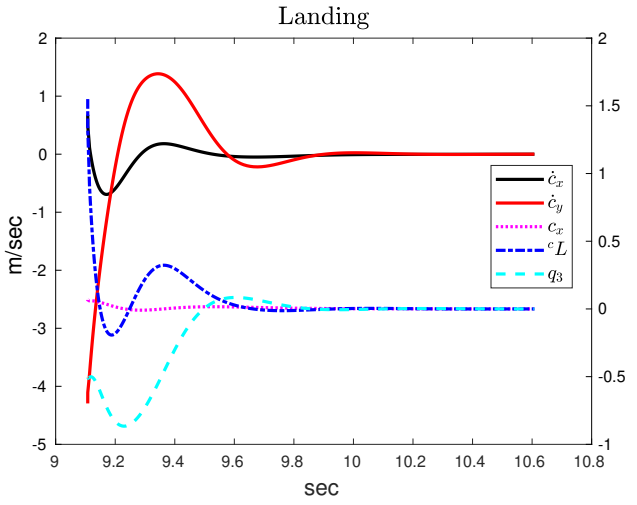

Fig. 6. Evolution of the robot's CoM's velocity $\dot{c}$, centroidal angular momentum ^cL, position of the CoM along the x axis c_x and position of the actuated joint q_3. The left-side scale applies to $\dot{c}_x$ and $\dot{c}_y$ and the right-side scale applies to c_x, ^cL and q_3.

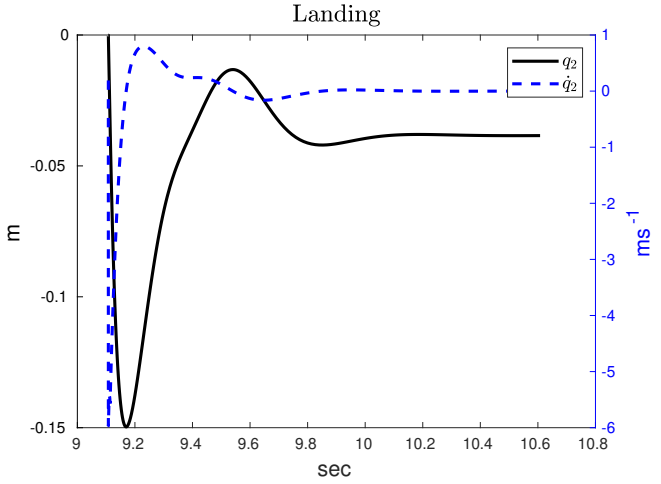

Fig. 7. Evolution of the spring-loaded joint position q_2 and velocity $\dot{q}_2$ after the touchdown. The left-side scale applies to q_2 and the right-side scale applies to $\dot{q}_2$.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3287769

a maximum of 5.816 J, and the balance controller successfully released most of this energy while keeping the robot's balance.

VI. Conclusion

This work presented a scalable optimization framework for a spring-loaded monopod robot performing a double backflip. The event-based optimization approach is the most suitable for tasks with unknown motion duration, like hopping.

The ability of the robot to store and use elastic energy has demonstrated a substantial increase in performance for achieving a faster and more complex takeoff and reducing the impact force at landing. This strategy is still valid for a continuous hop task for recycling energy from one hop to another. Moreover, the optimization framework is easily adaptable to different robots by modifying the robot model. Extra equations depending on the system's dynamics can also be implemented using the algebraic representation obtained.

The proposed strategy succeeded in performing a double backflip with the spring-loaded monopod and significantly reduced the tracking errors obtained in [5] for producing a takeoff motion, which is crucial for performing complex movements.

On the other hand, the controller also demonstrated the ability to accurately drive the CoM's velocity and centroidal angular momentum to achieve the necessary conditions to takeoff successfully and perform the double backflip. The spring parameters (according to [31]) also played an essential role in reducing the impact force at the landing to achieve a smooth motion while recovering the fixed balance position.

References

[1] M. T. Pope, S. Christensen, D. Christensen, A. Simeonov, G. Imahara, and G. Niemeyer, "Stickman: Towards a human scale acrobatic robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2134–2140, 2018.

[2] A. Kollarčík, *Modeling and Control of Two-Legged Wheeled Robot*. PhD thesis, Czech Technical University in Prague. Faculty of Electrical Engineering, Department of Control Engineering, 2021.

[3] X. Xiong and A. Ames, "Sequential motion planning for bipedal somersault via flywheel slip and momentum transmission with task space control," 2020.

[4] D. Tian, J. Gao, C. Liu, and X. Shi, "Simulation of upward jump control for one-legged robot based on qp optimization," *Sensors*, vol. 21, no. 5, 2021.

[5] M. Azad and R. Featherstone, "Balancing and Hopping Motion of a Planar Hopper with One Actuator," in *2013 IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), pp. 2027–2032, May 6–10 2013.

[6] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2008.

[7] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[8] R. Featherstone, "A Simple Model of Balancing in the Plane and a Simple Preview Balance Controller," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1489–1507, 2017.

[9] J. J. M. Driessen, A. E. Gkikakis, R. Featherstone, and B. R. P. Singh, "Experimental Demonstration of High-Performance Robotic Balancing," in *2019 International Conference on Robotics and Automation*, (Montreal, Canada), pp. 9459–9465, May 20-24 2019.

[10] C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D. G. Caldwell, and C. Semini, "Line walking and balancing for legged robots with point feet," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3649–3656, 2020.

[11] J. K. Yim, B. R. P. Singh, E. K. Wang, R. Featherstone, and R. S. Fearing, "Precision Robotic Leaping and Landing Using Stance-Phase Balance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3422–3429, 2020.

[12] M. W. Spong, "The Swing up Control Problem for the Acrobot," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 49–55, 1995.

[13] J. W. Grizzle, C. H. Moog, and C. Chevallereau, "Nonlinear Control of Mechanical Systems with an Unactuated Cyclic Variable," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 559–576, 2005.

[14] M. D. Berkemeier and R. S. Fearing, "Tracking Fast Inverted Trajectories of the Underactuated Acrobot," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 740–750, 1999.

[15] J. D. Gamba and R. Featherstone, "Balancing on a Springy Leg," in *2021 IEEE International Conference on Robotics and Automation*, (Xi'an, China), June 01-03 2021.

[16] J. D. Gamba, A. C. Leite, and R. Featherstone, "Robust balancing control of a spring-legged robot based on a high-order sliding mode observer," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pp. 384–391, 2021.

[17] M. H. Raibert, H. B. Brown, M. Chepponis, J. Koechling, J. K. Hodgins, D. Dustman, W. K. Brennan, D. S. Barrett, C. M. Thompson, J. D. Hebert, W. Lee, and L. Borvansky, "Dynamically stable legged locomotion," Tech. Rep. 1179, MIT, 1989.

[18] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Frison, and M. Diehl, "NMPC for racing using a singularity-free path-parametric model with obstacle avoidance," in *Proceedings of the IFAC World Congress*, 2020.

[19] T. Dinev, S. Xin, W. Merkt, V. Ivan, and S. Vijayakumar, "Modeling and control of a hybrid wheeled jumping robot," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2020.

[20] M. Fevre, P. M. Wensing, and J. P. Schmiedeler, "Rapid bipedal gait optimization in casadi," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pp. 3672–3678, IEEE, 2020.

[21] A. Witkin and M. Kass, "Spacetime constraints," *SIGGRAPH Comput. Graph.*, vol. 22, p. 159–168, jun 1988.

[22] O. von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Ann. Oper. Res.*, vol. 37, p. 357–373, Jan. 1992.

[23] A. Cervantes and L. T. Biegler, "Optimization strategies for dynamic systems," in *Encyclopedia of Optimization, Second Edition* (C. A. Floudas and P. M. Pardalos, eds.), pp. 2847–2858, Springer, 2009.

[24] L. T. Biegler, *Nonlinear Programming*. Society for Industrial and Applied Mathematics, 2010.

[25] L. Biegler, A. Cervantes, and A. Wachter, "Advances in simultaneous strategies for dynamic process optimization," *Chemical Engineering Science*, vol. 57, pp. 575–593, 02 2002.

[26] N. Hale and A. Townsend, "Fast and accurate computation of gauss–legendre and gauss–jacobi quadrature nodes and weights," *SIAM Journal on Scientific Computing*, vol. 35, no. 2, pp. A652–A674, 2013.

[27] L. N. Trefethen, *Approximation Theory and Approximation Practice (Other Titles in Applied Mathematics)*. USA: Society for Industrial and Applied Mathematics, 2012.

[28] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, p. 25–57, Mar. 2006.

[29] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington, "Brief paper: A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, vol. 46, p. 1843–1851, Nov. 2010.

[30] HSL, "A collection of Fortran codes for large scale scientific computation." //https://www.hsl.rl.ac.uk/, 2021.05.05. Last access: Sept. 13th 2022.

[31] D. A. Peters, "Optimum spring-damper design for mass impact," *SIAM Review*, vol. 39, no. 1, pp. 118–122, 1997.

[32] R. Featherstone, "Quantitative Measures of a Robot's Physical Ability to Balance," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1681–1696, 2016.

[33] J. D. Gamba, *Hopping, Landing, and Balancing with Springs*. PhD thesis, Italian Institute of Technology, and University of Genova, 2022.

[34] M. Azad, *Balancing and hopping motion control algorithms for an under-actuated robot*. PhD thesis, Australian National University. Research School of Engineering, 2014.

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.