

# Control of Humanoid Robots with Parallel Mechanisms using Differential Actuation Models

Victor Lutz<sup>1,\*</sup>, Ludovic De Matteis<sup>1</sup>, Virgile Batto<sup>1,2</sup> Nicolas Mansard<sup>1,3</sup>

**Abstract**—Several recently released humanoid robots, inspired by the mechanical design of Cassie, employ actuator configurations in which the motors are displaced from the joints to reduce leg inertia. While studies accounting for the full kinematic complexity have demonstrated the benefits of these designs, the associated loop-closure constraints greatly increase computational cost and limit their use in control and learning. As a result, the non-linear transmission is often approximated by a constant reduction ratio, preventing exploitation of the mechanism’s full capabilities. This paper introduces a compact analytical formulation for the two standard knee and ankle mechanisms that captures the exact non-linear transmission while remaining computationally efficient. The model is fully differentiable up to second order with a minimal formulation, enabling low-cost evaluation of dynamic derivatives for trajectory optimization and of the apparent transmission impedance for reinforcement learning. We integrate this formulation into trajectory optimization and locomotion policy learning, and compare it against simplified constant-ratio approaches. Hardware experiments demonstrate improved accuracy and robustness, showing that the proposed method provides a practical means to incorporate parallel actuation into modern control algorithms.

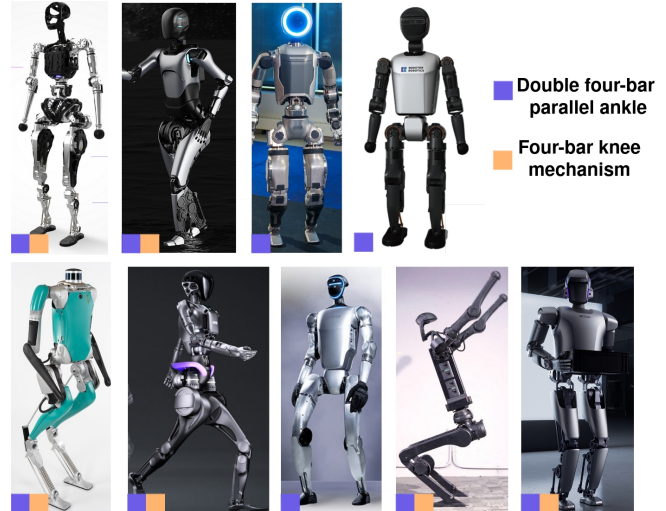


Fig. 1: Recent humanoids using parallel architectures. From top-left to bottom-right: Adam, A2, Atlas, T1, Digit V2, GR1, G1, H1, Walker S1 (Non exhaustive list)

## I. INTRODUCTION

Recent advances in biped locomotion have largely been driven by new hardware designs. Many companies and research groups, designing the most recent biped robots, shifted toward serial-parallel architecture (see Fig. 1), especially in the leg design. These architectures reduce limb reflected inertia and improve impact absorption [1], [2], thereby enabling more dynamic movements, albeit at the cost of more complex modeling and control.

In most control frameworks - whether based on Whole-Body Model Predictive Control (WB-MPC) or Reinforcement Learning (RL) - motions are usually computed in the joint space using a serial dynamical model. Several approaches have been described in the literature. The most direct one exactly models the parallel branching of the kinematics, accounting for the closed-loop constraint and the additional internal degrees of freedom (DoF) [3]. Although described early in the literature [4], the additional algorithmic cost has long been a bottleneck to their deployment until

recent advances [5]–[8]. Yet, the cost of solving the resulting motion problem remains.

On the other hand, dedicated analytical models have been formulated with reduced cost. A systematic analytical model of the transmission is derived in [9], from which we take inspiration in this work. Dedicated studies have been proposed in the literature for some designs [9]–[11]. It has also been proposed to rely on heuristic models with equivalent dynamic property but reduced computational cost [12]. On commercial robots, the handling of the parallel mechanism is often done in a closed-source algorithmic layer hidden to the user and sparsely documented. To mitigate the need for this conversion layer, some robot design rely on unitary transmissions [13], [14]. Yet, we notice that most robots rely on kinematic mechanisms displaying a near-constant transmission ratio, which possibly allows to ignore the transmission non-linearity. We will also show in the result section that such a hypothesis is quite limiting.

In RL, accounting for the full robot dynamics requires a simulator that supports closed-loop mechanisms. Until recently, such simulators were limited to CPU, making it challenging for locomotion policy learning [15]. Recent works used new generation GPU simulators such as Isaac Lab and Mujoco MJX to train locomotion policies with the full robot dynamics [13], [16], [17]. Despite some significant overhead, the computation time is only increased offline and

This work is supported by ROBOTEX 2.0 (ROBOTEX ANR-10-EQPX-0044 and TIRREX ANR-21-ESRE-0015), ANITI (ANR-19-P3IA-0004), by the French government (INEXACT ANR-22-CE33-0007 and "Investissements d'avenir" ANR-19-P3IA-0001) (PRAIRIE 3IA Institute), and by the Louis Vuitton ENS Chair on Artificial Intelligence

<sup>1</sup> Gepetto, LAAS-CNRS, Université de Toulouse, France

<sup>2</sup> Auctus, Inria, centre de l'université de Bordeaux, Talence, France

<sup>3</sup> Artificial and Natural Intelligence Toulouse Institute, France

\* Corresponding author: [nmansard@laas.fr](mailto:nmansard@laas.fr)

does not impact policy inference. However, these simulators yet rely on soft or approximate contacts models [18] for constraining the closed chain, eventually increasing the sim-to-real gap and limiting deployments on real hardware. This requires using specific techniques such as adversarial training or contact solver tuning to mitigate the drawbacks of the simulator [13], [16], [17]. A recent study has shown that policy training with a serial model and actuator-space output is possible [19]. Yet, these methods still come at extra costs and burden, are not yet mature and fail to impact recent developments, which all rely on a limiting serial model.

In this paper, we propose a complete solution to accurately account for serial-parallel designs, both in WB-MPC and in RL, with a minimal additional implementation complexity and algorithmic cost. Building on existing analytical models of the transmission [9], we propose a dedicated formulation of the two most classical transmission (see Fig. 1) which leads to efficient derivatives. These derivatives are key to the efficiency of WB-MPC solvers. Extending some early work [20], we show that these models modify the impedance of the actuator and derive an analytical model of it, also relying on the model derivatives. Finally, we empirically evaluate the contributions and show the added value in the control compared to simplified serial model.

In the following, we first derive a **differential analytical model** of the geometric and kinematic mappings of the closed-loop transmissions composing modern robot architectures (Sec. II), obtaining cheap derivatives of the dynamics and the transmission reflected impedance. We then propose two formulations to generate movements with this model, by trajectory optimization using the model derivatives and reinforcement learning using the derived transmission impedance (Sec. III). The implementation of our contributions are empirically compared (Sec. IV), demonstrating the benefits of such approach in Trajectory Optimization (TO) and its capabilities through an RL framework with impedance transfer. We finally validate the method on a real robot with parallel knees and ankles.

## II. METHOD

In this section, we first provide the equations for a 1-DoF parallel transmission before extending it to a 2-DoF transmission. These transmissions are typically used in the knee and ankle design of many robots. For clarity, we illustrate the designs with the robot Bipetto (Fig. 2), where the linkages clearly appear as it has no covers. However, all the developments directly apply on the other design shown in Fig 1. We then derive the expression to obtain the Actuation Jacobian (first order), relating the motor torque to the actuated serial joint torque, and the transmission dynamics (second order). Finally, we present a method to transfer impedance gains from the serial-space to the actuator-space, allowing impedance control and maintaining stabilization properties.

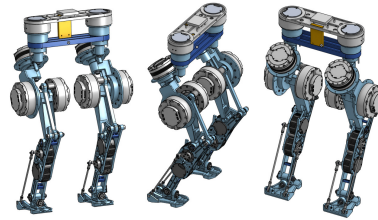


Fig. 2: The Bipetto robot includes serial-parallel architecture, with parallel actuation for the knees and ankles.

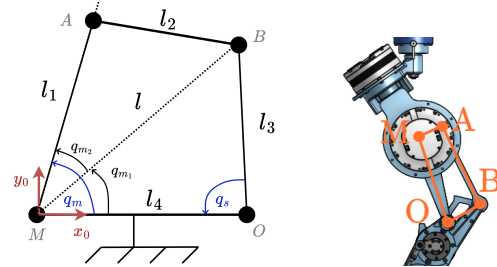


Fig. 3: Planar 4-bar mechanism, with the serial link rotating around O, of angle  $q_s$ , motor rotating around M of angle  $q_m$ , B the attachment of the linkage on the lower limb and A the joint of the closed-loop linkage. A concrete example is given with the knee of the Bipetto robot.

### A. Direct Geometry

1) *Four-bar linkages*: We first consider a simple planar four-bars linkage transmission, present in many modern robots such as in the knee of H1 [21] or in the ankle of Talos [22]. The knee actuation of our robot Bipetto is shown in Fig. 2 and Fig. 3. In this mechanism we denote  $q_m$  the motor joint configuration and  $q_s$  the configuration of the actuated serial joint.

The transmission is represented by the mapping  $f$  for which we seek an efficient formulation.

$$q_m \triangleq f(q_s) \quad (1)$$

We denote by  $b = (x_B, y_B)^T \in \mathbb{R}^2$  the coordinate vector of point B in the motor frame,  $l(b) = \|b\|$  its norm,  $q_{m_1} = \widehat{OMB}$ ,  $q_{m_2} = \widehat{BMA}$  and  $r = \cos(q_{m_2})$ . Applying Al-Kashi theorem (law of cosines) yields:

$$r(b) = \frac{l(b)^2 + l_1^2 - l_2^2}{2l(b)l_1} \quad (2)$$

The motor configuration is given by  $q_m = q_{m_1} + q_{m_2}$ , with:

$$\begin{cases} q_{m_1}(b) = \text{atan2}(y_B, x_B) \\ q_{m_2}(b) = \text{acos}(r(b)) \end{cases} \quad (3)$$

We restrict our derivations to  $m_2 \in [0, \pi]$ , giving a bijection between  $r$  and  $m_2$ . We get that the motor configuration  $q_m(b) = q_{m_1}(b) + q_{m_2}(b)$  is a function of the position of the point B. Moreover, we can express the coordinate of the point B in the motor frame as a function of the serial joint configuration:  $b(q_s) = [l_4 + l_3 \cos(q_s), l_3 \sin(q_s)]$ , or

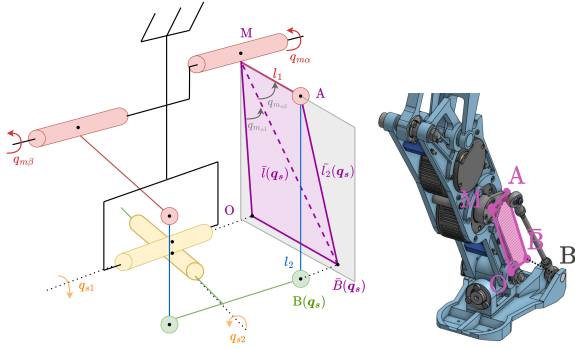


Fig. 4: Sub projected planar four-bar from the ankle in purple. A concrete example is given on the right (ankle of the Bipetto robot)

directly express it as the forward kinematics of the serial chain denoted by  $FK$ :

$$b(q_s) = FK(q_s) \quad (4)$$

Combining (4) with (3) gives the complete expression of  $q_m = f(q_s)$ .

2) *Intricate four-bar linkages*: A more complex mechanism, consisting in two intricate four-bar linkages, yielding a coupled control of two serial DoF using two motors, can be observed in the ankles of H1, G1, GR1, Digit, T1. A schematic representation of this mechanism is presented in Fig. 4. While this mechanism is not planar, each side of it (denoted  $\alpha$  and  $\beta$ ) can be projected into a plane, orthogonal to the motor joint and going through the linkage point A, resulting in a virtual planar four-bars linkage.

Focusing on one side of the mechanism, we denote as  $\bar{B}$  the projection of  $B$  on the plane, so that  $b \in \mathbb{R}^3$  and  $\bar{b} = (x_B, y_B)^T \in \mathbb{R}^2$ . Let us note,  $l_1 = \|\vec{AM}\|$ ,  $l_2 = \|\vec{AB}\|$ ,  $\bar{l}_2^2(z_B) = l_2^2 - z_B^2 = \|\vec{A\bar{B}}\|$ , and  $\bar{l}(\bar{b}) = \|\vec{M\bar{B}}\|$ . We can now proceed as before, with  $r(b)$  that becomes:

$$r(b) = \frac{\bar{l}(\bar{b})^2 + l_1^2 - \bar{l}_2(z_B)^2}{2\bar{l}(\bar{b})l_1} \quad (5)$$

By doing so, we get a virtual planar four-bar mechanism that gives a relation between one motor configuration  $q_{m_\alpha}$  and the ankle configuration  $q_s = (q_{s_1} \quad q_{s_2})^T$ . Applying this method on both sides of the mechanism yields the relation:

$$q_m = \begin{pmatrix} q_{m_\alpha} \\ q_{m_\beta} \end{pmatrix} = \begin{pmatrix} f_\alpha(q_s) \\ f_\beta(q_s) \end{pmatrix} = f(q_s) \quad (6)$$

This relation generalizes the four-bar linkage (as it is its own projection) and we will use this notation for both mechanisms.

### B. Actuation Jacobian

To transfer motor controls to their actions on the serial joints, we need a relation between the motor torques  $\tau_m$  and the serial joints torques  $\tau_s$ . To obtain such a relation, we compute the so called **Actuation Jacobian**  $J_A$ , function of

$q_s$  that satisfies the relations:

$$\begin{cases} \dot{q}_m = J_A(q_s)\dot{q}_s \\ \tau_s = J_A(q_s)^T \tau_m \end{cases} \quad (7)$$

In this section, we will focus on the intricate four-bar mechanism. We first consider the effect of a single motor on the serial joints, before summing the contributions of the two motors (denoted  $M_\alpha$  and  $M_\beta$ ). Without loss of generality, we derive the equations for  $\tau_{m_\alpha}$  and denote  $\tau_{s[\tau_{m_\alpha}]}$  the serial torques it induces and  $J_{A,\alpha}$  the corresponding Actuation Jacobian. The derivations for  $\tau_{s[\tau_{m_\beta}]}$  do not add any additional complexity. Computing the derivative of the mapping  $f$  with respect to  $q_s$  gives

$$\frac{dq_{m_\alpha}}{dq_s} = \frac{df_\alpha(q_s)}{dq_s} \triangleq J_{A,\alpha}(q_s) \quad (8)$$

Since we focus on one side of the mechanism, we drop the subscript  $\alpha$  from the four-bar notations, using for instance  $q_{m_1}$  instead of  $q_{m_{\alpha_1}}$ . To apply the chain rule, we write  $q_{m_\alpha} = f_\alpha(q_s) = q_{m_1}(b) + q_{m_2}(r(\bar{l}(\bar{b}), \bar{l}_2(z_B)))$  and  $b = \begin{pmatrix} \bar{b} \\ z_B \end{pmatrix} = FK(q_s)$ , yielding:

$$\frac{dq_{m_\alpha}}{dq_s} = \left( \frac{dq_{m_1}}{db} + \frac{dq_{m_2}}{db} \right) \frac{db}{dq_s} \quad (9)$$

We will note  $\frac{db}{dq_s} = B_s$  and compute it using the analytical derivatives of the forward kinematics, implemented in Pinocchio [23]. The other terms are computed from Eq. (3):

$$\frac{dq_{m_1}}{db} = \frac{1}{\bar{l}^2} \begin{pmatrix} -y_B & x_B & 0 \end{pmatrix} = b^T \begin{bmatrix} 0 & 1/\bar{l}^2 & 0 \\ -1/\bar{l}^2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (10)$$

Introducing  $\hat{r} = \sin(q_{m_2})$ , we also get:

$$\frac{dq_{m_2}}{db} = \begin{cases} \frac{dq_{m_2}}{db} = \frac{\partial q_{m_2}}{\partial r} \frac{\partial r}{\partial \bar{l}} \frac{d\bar{l}}{d\bar{b}} = \frac{l_1 r - \bar{l}}{\hat{r} \bar{l}^2} \bar{b}^T \\ \frac{dq_{m_2}}{dz_B} = \frac{\partial q_{m_2}}{\partial r} \frac{\partial r}{\partial \bar{l}_2} \frac{d\bar{l}_2}{dz_B} = \frac{-z_B}{\hat{r} \bar{l}_1} \end{cases} \quad (11)$$

This can be condensed as

$$\frac{dq_{m_\alpha}}{db} = \frac{dq_{m_1}}{db} + \frac{dq_{m_2}}{db} = b^T \underbrace{\begin{bmatrix} \mu & \nu & 0 \\ -\nu & \mu & 0 \\ 0 & 0 & \xi \end{bmatrix}}_K \quad (12)$$

where  $\mu = \frac{r l_1 - \bar{l}}{\hat{r} \bar{l}^2 l_1}$ ,  $\nu = \frac{1}{\bar{l}^2}$  and  $\xi = \frac{-1}{\hat{r} \bar{l}_1}$ . It follows:

$$J_{A,\alpha}(q_s) \triangleq \frac{dq_{m_\alpha}}{dq_s} = b(q_s)^T K(r(q_s), l(q_s)) B_s(q_s) \quad (13)$$

It is worth noting that for the intricate four-bar (i.e. the ankle transmission),  $B_s \in \mathbb{R}^{3 \times 2}$ , giving that  $J_{A,\alpha}$  (same holds for  $J_{A,\beta}$ ) is an element of  $\mathbb{R}^{1 \times 2}$  while it is a scalar for the four-bar mechanism (and equals  $J_A$ ). For the full ankle, we obtain

$$J_A = \begin{bmatrix} J_{A,\alpha} \\ J_{A,\beta} \end{bmatrix} \in \mathbb{R}^{2 \times 2} \quad (14)$$

This matrix can evidently be obtained from any other analytical model such as [9]. However, the resulting formulation

is very compact, although trying to compare the algorithmic cost is likely useless. We need it for the next step, since any automatic differentiation we tried failed to produce a short and efficient formulation of the second-order terms.

### C. Actuation derivatives

We now extend the derivatives to  $\frac{\partial \tau_s}{\partial u}$  and  $\frac{\partial \tau_s}{\partial x_s}$ , which we need to compute the derivatives of the dynamic [23]. Our approach allows controlling directly the motor torques, giving  $u = \tau_m$ , through eq. (7). The derivative of the serial torques with respect to  $u$  is then the Actuation Jacobian. The state of the robot consists of the serial joints configuration and velocity  $x_s = [q_s, \dot{q}_s]$ . The derivative of the serial torques with respect to the articular velocity directly yields:

$$\frac{d\tau_{s[\tau_{m\alpha}]}}{dq_s} = 0 \quad (15)$$

Differentiating  $\tau_s$  with respect to the configuration  $q_s$  is more complex, since the product rule naturally involve tensorial elements. We will split again the Actuation Jacobian into two parts  $J_{A,\alpha}$  and  $J_{A,\beta}$ , and focus on the first term. Let us differentiate this product by rewriting it differently:

$$\tau_{s[\tau_{m\alpha}]} = J_{A,\alpha}^T \tau_{m\alpha} = B_s^T \lambda \quad (16)$$

where  $\lambda = K^T b \tau_{m\alpha}$  corresponds to the force applied by the motor  $\alpha$  on the point B. Deriving this relation gives

$$\frac{d\tau_{s[\tau_{m\alpha}]}}{dq_s} = \frac{dB_s^T}{dq_s} \lambda + B_s^T \frac{d\lambda}{db} B_s \quad (17)$$

In the first term, we denote  $\frac{dB_s^T}{dq_s}$  as  $B_{ss}$  and directly compute the term  $B_{ss}^T \lambda$  using the spatial algebra implemented in Pinocchio [24] to avoid manipulating tensors.

The second term is given by

$$\frac{d\lambda}{db} = \frac{dK^T}{db} b \tau_{m\alpha} + K^T \tau_{m\alpha} \quad (18)$$

where the derivative of  $K(\bar{l}(b), r(q_{m_2}(b)), \hat{r}(q_{m_2}(b)))$  with respect to  $b$ , is given by:

$$\frac{dK}{db} = \frac{\partial K}{\partial \bar{l}} \frac{d\bar{l}}{db} + \left( \frac{\partial K}{\partial r} \frac{dr}{dq_{m_2}} + \frac{\partial K}{\partial \hat{r}} \frac{d\hat{r}}{dq_{m_2}} \right) \frac{dq_{m_2}}{db} \quad (19)$$

The matrix elements  $\frac{\partial K}{\partial \bar{l}}$ ,  $\frac{\partial K}{\partial r}$  and  $\frac{\partial K}{\partial \hat{r}}$  are computed by differentiating  $\mu(\bar{l}, r, \hat{r})$ ,  $\nu(\bar{l})$  and  $\xi(\bar{l}, \hat{r})$  separately. The resulting non-zero elements are:

$$\begin{aligned} \mu_{\bar{l}} &= \frac{\partial \mu}{\partial \bar{l}} = \frac{\bar{l} - 2rl_1}{\hat{r}l^3 l_1} & \mu_r &= \frac{\partial \mu}{\partial r} = \frac{1}{\hat{r}l^2} \\ \mu_{\hat{r}} &= \frac{\partial \mu}{\partial \hat{r}} = \frac{\bar{l} - rl_1}{\hat{r}^2 l^2 l_1} & \nu_{\bar{l}} &= \frac{\partial \nu}{\partial \bar{l}} = \frac{-2}{\bar{l}^3} \\ \xi_{\bar{l}} &= \frac{\partial \xi}{\partial \bar{l}} = \frac{1}{\hat{r}l^2 l_1} & \xi_{\hat{r}} &= \frac{\partial \xi}{\partial \hat{r}} = \frac{1}{\hat{r}^2 \bar{l} l_1} \end{aligned} \quad (20)$$

Finally, the covector  $\frac{d\bar{l}}{db}$  and the scalars  $\frac{dr}{dq_{m_2}}$  and  $\frac{d\hat{r}}{dq_{m_2}}$  are computed to be:

$$\frac{d\bar{l}}{db} = \frac{\bar{b}}{\bar{l}} \quad \frac{dr}{dq_{m_2}} = -\hat{r} \quad \frac{d\hat{r}}{dq_{m_2}} = r \quad (21)$$

Using (11) for  $\frac{dq_{m_2}}{db}$  concludes the derivation of  $\frac{d\lambda}{db}$  and gives

$$\frac{d\tau_{s[\tau_{m\alpha}]}}{dq_s} = B_{ss}^T \lambda + B_s^T \left( \frac{dK^T}{db} b + K^T \right) B_s \tau_{m\alpha} \quad (22)$$

In the end, we can recover the dependency from each motor by summing their influence to the final derivative:

$$\frac{d\tau_s}{dq_s} = \sum_i \frac{d\tau_{s[\tau_{m_i}]}}{dq_s} \quad (23)$$

with  $i \in \{\alpha\}$  for the four-bar and  $i \in \{\alpha, \beta\}$  for the ankle transmission.

### D. Serial impedance transfer

The classical approach for controlling robots (in particular nearly systematic in RL or for sample based MPC [25]) is to impose an active impedance to the serial joint through a PD controller whose reference  $q^*$  is the output of the high level policy. This approach provides a compliant control law implemented at high frequency on the robot motors that stabilizes the system between two control steps  $t_k$  and  $t_{k+1}$ :

$$\tau(t) = K_P(q^*(t_k) - q(t)) - K_D \dot{q}(t) \quad t \in [t_k, t_{k+1}] \quad (24)$$

While having constant stiffness and damping at joint level is desirable for policy training, transferring this impedance control to the actuators of a robot with parallel mechanism is not straightforward and requires a transfer of the gains from the joint-space into the actuator-space. At time  $t_k$ , a policy trained in serial space gives a reference  $q_s^*(t_k)$  that generates a torque  $\tau_s^*(t_k)$  using (24). In the case of a PD in serial joint space, we denote  $K_{Ps}$  and  $K_{Ds}$  the impedance gains of the serial joint. Using the Actuation Jacobian, we get the corresponding desired motor torque  $\tau_m^*(t_k) = J_A(q_s(t_k))^{-T} \tau_s^*(t_k)$ .

To ensure the low-level controller acts in a stabilizing way, the derivatives of the applied torque with respect to the robot state must correspond to those of the desired torque ((5)). Using (24), this condition at control time  $t_k$  becomes [20]:

$$\begin{aligned} \frac{d\tau_m}{dq_m}(t_k) &= -K_{Pm}(t_k) = \frac{d\tau_m^*}{dq_m}(t_k) \\ \frac{d\tau_m}{dq_m}(t_k) &= -K_{Dm}(t_k) = \frac{d\tau_m^*}{dq_m}(t_k) \end{aligned} \quad (25)$$

where  $K_{Pm}(t_k)$  and  $K_{Dm}(t_k)$  denote the impedance gains in the motor-space for  $t \in [t_k, t_{k+1}]$ . Developing (25) gives:

$$\begin{aligned} K_{Pm}^*(t_k) &= - \underbrace{\left( \frac{dJ_A(q_s)^{-T}}{dq_m} \right) \tau_s}_{A_{Pm}} + \underbrace{J_A(q_s)^{-T} K_{Ps}}_{B_{Pm}} \frac{dq_s}{dq_m} \\ &\quad + \underbrace{J_A(q_s)^{-T} K_{Ds}}_{C_{Pm}} \frac{dq_s}{dq_m} \end{aligned} \quad (26)$$

Previous studies [20] adopted a similar approach, while neglecting the  $A_{pm}$  and  $C_{pm}$  terms by assuming that  $(q_s^* - q_s)$  remains small. However, this assumption is not valid in

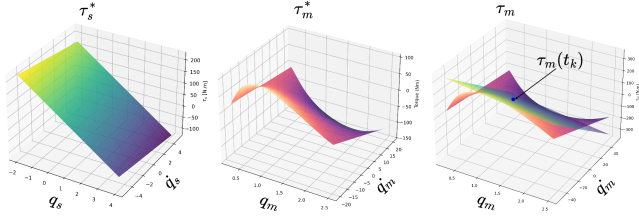


Fig. 5: The constant serial gains generate an affine (with slopes  $K_{P_s}$  and  $K_{D_s}$ ) torque control in the joint-space (left). Computing the corresponding motor torques using the Actuation Model gives a non-linear control law (middle) that is approximated with a tangent plane (i.e. with correct  $K_{P_m}$  and  $K_{D_m}$ ) at the desired point  $\tau_m^* = J_A^{-1}\tau_s^*$ . Using the reference torque  $\tau_m^*$  without feedback gains would lead to a horizontal plane which is a very wrong approximation of the curved manifold (in the middle).

the context of reinforcement learning. The term  $B_{P_m}$  can directly be written as:

$$B_{P_m} = J_A(q_s)^{-T} K_{P_s} J_A(q_s)^{-1} \quad (27)$$

Let us derive the  $A_{P_m}$  term:

$$A_{P_m} = J_A^{-T} \left( \frac{dJ_A(q_s)}{dq_m} \right)^T J_A^{-T} \tau_s = J_A^{-T} \left( \frac{dJ_A(q_s)}{dq_s} \frac{dq_s}{dq_m} \right)^T \tau_m \quad (28)$$

Which is computed in the same way as the derivative of (16)

$$A_{P_m} = J_A^{-T} J_A^{-T} \frac{dJ_A(q_s)^T \tau_m}{dq_s} \Big|_{\tau_m = \text{const.}} \quad (29)$$

Finally, we can derive the  $C_{p_m}$  term with similar operations:

$$C_{P_m} = -J_A(q_s)^{-T} K_{D_s} J_A(q_s)^{-1} \frac{dJ_A(q_s)u}{dq_s} \Big|_{u=J_A(q_s)^{-2}q_m} \quad (30)$$

For the damping gain, we denote the analog terms in the derivative  $A_{D_m}$ ,  $B_{D_m}$  and  $C_{D_m}$ . Among those, only  $C_{D_m}$  is non-zero:

$$C_{D_m} = J_A(q_s)^{-T} K_{D_s} J_A(q_s)^{-1} \quad (31)$$

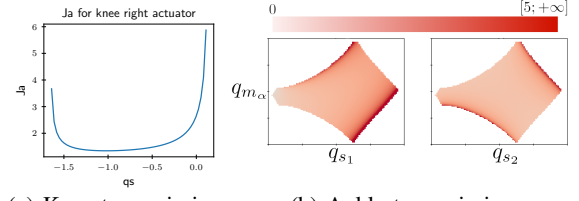
Finally, to make sure the control law is consistent, we enforce  $\tau_m(t_k) = \tau_m^*(t_k)$  by finding  $q_m^*$  with:

$$q_m^*(t_k) = (K_{P_m}^*(t_k))^{-1} [\tau_m^*(t_k) + K_{D_m}^*(t_k) \dot{q}_m(t_k) + K_{P_m}^*(t_k) q_m(t_k)] \quad (32)$$

### E. State estimation

In a MPC or RL setting, the actual  $q_s$  angles from the robot are needed. However, on the real system, the motor encoders provide only  $q_m$ , requiring the inverse mapping  $f^{-1}$  to recover  $q_s$ . While an analytical form can be found for the knee, we cannot invert the ankle geometry model. In both cases, we can estimate  $q_s$  via numerical optimization:

$$q_s = f^{-1}(q_m) = \min_{\hat{q}_s} \mathcal{L}(\hat{q}_s, q_m) = \min_{\hat{q}_s} \|f(\hat{q}_s) - q_m\|^2 \quad (33)$$



(a) Knee transmission (b) Ankle transmission

Fig. 6: Variable transmission ratio  $J_A(q_s)$  for parallel transmissions of our robot (four-bar and intricate-four bar). For the ankle, the contribution of motor  $\alpha$  alone on the two DoF of the ankle -  $q_{s1}$  and  $q_{s2}$  - is shown ( $\frac{dq_s}{dq_{m\alpha}}$ ). The diamond shape on each plot correspond to the feasible space of the mechanism and the color represents the absolute value of the scalar reduction ratios  $\frac{dq_{s1}}{dq_{m\alpha}}$  and  $\frac{dq_{s2}}{dq_{m\alpha}}$ .

Gradient of the state estimation cost function is given by

$$\nabla \mathcal{L}_{\hat{q}_s} = 2J_A^T(\hat{q}_s)(f(\hat{q}_s) - q_m) \quad (34)$$

State continuity allows using accurate warm starts, enabling rapid convergence of the minimization problem. Serial velocities  $\dot{q}_s$  can be simply recovered using (7).

## III. IMPLEMENTATION

### A. Robot model

We implemented our strategy on the Bipetto robot, showcased in Fig. 2. The robot is composed of a four-bar transmission for the knee joint and of an intricate four-bars transmission for the ankle, following the inspiration of Digit, H1 and others. We define the *Minimal Serial* model of the robot by freezing the joints in the parallel transmission and adding fictive actuation in the serial joints, resulting in a model with 6 DoF per leg.

To account for the parallel actuation of the robot, represented in Fig. 3 and Fig. 4, we compute the corresponding actuation jacobians and their derivatives. This extends the *Minimal Serial* model to create our *Actuated Serial* model, with same dimension. A visual representation of the Actuation Jacobians for our robot architecture are shown in Fig. 6.

Note that the variable reduction ratio for each mechanism increases (in absolute value) when the parallel mechanism gets closer to singularities. For the knee actuation, this corresponds to fully bent and fully stretched legs.

### B. Trajectory Optimization

We formulate our control problem as a multiple shooting Optimal Control Problem (OCP), solving for controls  $u[k]$  and states  $x[k]$  at each time step  $k$  [26]. This classically transcribes as a Non-Linear Program (NLP)

$$\begin{aligned} \min_{U, X} \quad & \sum_{k=0}^{N-1} l_k(x[k], u[k]) + l_N(x[N]) \\ \text{s.t.} \quad & \forall k \in \llbracket 0, N-1 \rrbracket \quad x[k+1] = f_k(x[k], u[k]) \end{aligned} \quad (35)$$

where  $l_k$  defines the running costs,  $l_N$  the terminal cost, and  $f_k$  is the dynamics of the system. We opt for a simplified

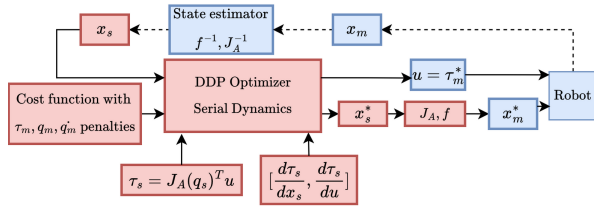


Fig. 7: Whole Body Trajectory Optimization strategy. The OCP is solved with a serial dynamics controlled directly with motor torques, thanks to our derivatives of the Actuation Model. The problem can include cost and constraint on either serial states  $x_s$  or motor states  $x_m$ . It outputs motor torques  $\tau_m$  directly sent to the robot. The dotted line shows the extension needed to turn the TO into MPC.

model of the robot dynamics by using a *Minimal Serial* model with joint actuation defined as  $\tau_s[k] = J_A^T u[k]$ , so that the controls correspond to motor torques  $u[k] = \tau_m[k]$  (see Fig. 7 for a global view of the TO strategy). This formulation accounts for the parallel actuation nonlinearities through the Actuation Jacobian while keeping the model complexity as low as possible.

The problem (35) is solved using the FDDP algorithm implemented within the Crocoddyl library [27], and relies on Pinocchio [24] for the serial dynamics computation. Our formulation of the dynamics also allows setting constraints (such as limits) on motor torques directly, which would otherwise not be possible for the coupled motors of the ankle. FDDP being a derivative based algorithm, it requires the derivatives of the dynamics with respect to the states and controls. While the algorithms implemented in Pinocchio can be used to compute the derivatives of the serial dynamics  $f_k(x[k], \tau_s[k])$  with respect to  $x[k]$  and  $\tau_s[k]$ , it is not sufficient for our implementation since the dependencies  $\tau_s(x[k], u[k])$  are not accounted for. Through our actuation model, we get  $\tau_s[k] = \tau_s(x[k], u[k])$  and add the corresponding additional derivatives, that we derived in section II-C.

Using the actuation model also allows using costs and constraints directly in the motor-space even though the OCP uses serial states. For instance, we set in problem (35) some lower and upper bounds ( $\underline{q}_m$  and  $\overline{q}_m$  respectively) on the ankle motor joints through the  $f$  mapping:

$$\underline{q}_m \leq f(q_s) \leq \overline{q}_m \quad (36)$$

We implemented our approach for a walk on a flat and on a rough terrain, and for a stair climbing problem. These tasks were chosen to push the robot close to his joint limits to demonstrate the impact of the non-linear transmission. The cost functions of (35) are standard for this kind of problem and we refer the reader to [28] for more details.

We compare the results with a *Minimal Serial* model, where  $u = \tau_s$  is turned a posteriori - when the  $q_s$  is feasible - into motor torques using the relation  $\tau_m = J_A^{-T} \tau_s$ .

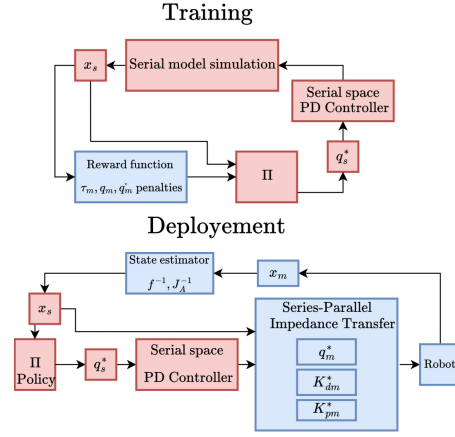


Fig. 8: RL training and deployment. During deployment, serial references pass through a conversion step that outputs  $q_m^*$ ,  $K_{Pm}$  and  $K_{Dm}$  for the actuator controller.

### C. Reinforcement Learning

We implemented bipedal locomotion in RL using the Isaac Lab framework. We used the constrained PPO formulation CaT [29] to avoid extensive tuning. Reward design is outside the scope of this paper, however, for interested readers, our rewards and constraints formulation is very close to [30]. Fig. 8 presents an overview of the training and deployment method, that we detail in the following sections. Our policy was trained on a Nvidia RTX 4090 GPU with 4096 parallel environments.

1) *Observation space*: To use a serial model in the simplest way, our observations contains the serial joints positions  $q_s$  and velocities  $\dot{q}_s$ , obtained through the state estimator described in II-E. Our observation vector also contains the projected gravity vector, as well as the robot base angular velocity, and velocity tracking commands. All observations are stored during a history of 6 steps and then concatenated in the final observation vector.

2) *Action space*: The policy is trained to output a serial position offset  $\delta_{q_s}$  that is added to a reference standing position  $q_{s0}$ , to create the reference position  $q_s^* = q_{s0} + \delta_{q_s}$ . This position offset then goes into a low-level simulated serial joint PD controller that generates torque  $\tau_s$  for the joint at a higher rate:  $\tau_s = K_{P_s}(q_s^* - q_s) - K_{D_s}\dot{q}_s$ . The impedance gains  $K_{P_s}$  and  $K_{D_s}$  are chosen to be constant for simplicity.

3) *Series-parallel deployment*: Once trained, our policy is deployed on an embedded CPU and uses the impedance transfer described in section II-D to transfer the impedance gains from the serial-space to the actuator-space. The policy inference is performed at 30Hz, the reference  $q_m^*$  and the gains  $K_{Pm}$  and  $K_{Dm}$  are updated at 100Hz to better fit the curved manifold of Fig. 5 and the motors low-level impedance controller runs at about 1kHz.

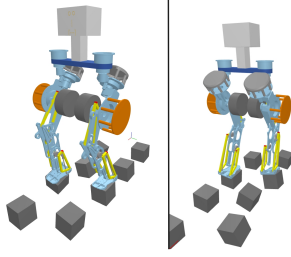


Fig. 9: Illustration of the task of walking on rough terrain. Ground orientation is randomly generated for each step.

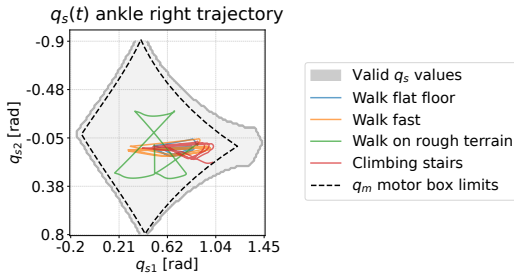


Fig. 10: Ankle  $q_s(t)$  for walking on flat floor (slowly and fast), walking on rough terrain, climbing stairs. The dotted black lines limit the feasible configurations on the real robot. No serial box constraint can cover all motions while staying in the motor constraints.

## IV. RESULTS

### A. Trajectory Optimization

We first perform various tasks to assess the applicability of our *Actuated Serial Model* and demonstrate the limits of the *Minimal Serial Model*. We perform a walk on a flat terrain at a constant target Center of Mass velocity of 0.7m/s (equivalent to 8km/h walk for a human-sized robot). On this movement, the *Actuated Serial Model*, that accounts for the closed-loop transmission in the optimization, and the *Minimal Serial* model, that optimizes on serial joints torques then converted into motor controls, yield similar results. However, the two models start to diverge when advancing toward more demanding movements such as walking on rough terrain (see Fig. 9) and climbing stairs. Fig. 10 presents the trajectories of the left and right ankles serial joints, with 2 DoF per ankle and opposes it to the limits of the actuation model, discussed in Sec. III-B. We observe that the *Actuated Serial Model* reaches ankles configurations that could not be attained with box constraints in the joint-space (that would appear as a straight square limit in Fig. 10). Indeed, clamping the serial joints range to allow walking on flat floor would prevent configurations yet necessary to walk on rough terrains or to climb high stairs. On the opposite, setting bounds that allow all the motions demonstrated here, would also allow unfeasible configurations. Our approach can successfully exploit this range of motion by stating the limits in the actuator space, resulting in more permissive, yet feasible, serial limits. The different movements can be seen in the companion video.

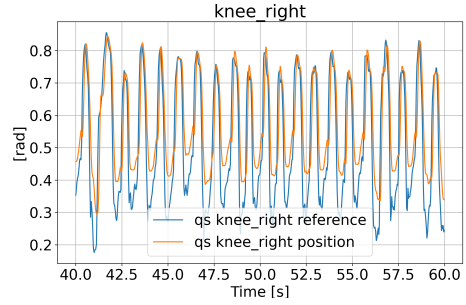


Fig. 11:  $q_s^*$  and  $q_s$  of right knee mechanism during walking policy deployment. Even with  $q_s^*$  outside of the limits, which is frequent in RL, the computed  $q_m^*$ ,  $K_{Pm}$  and  $K_{Dm}$  allow to replicate the behavior of the serial joints expected by the policy.

### B. Reinforcement Learning

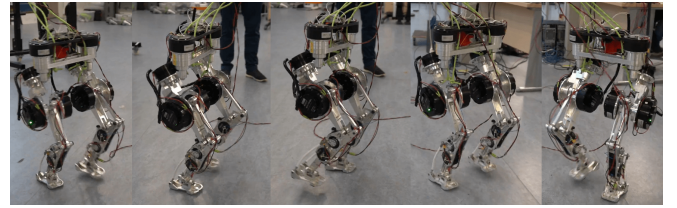


Fig. 12: RL with series-parallel impedance transfer allows successful omnidirectional walking policy on the real robot.

Using the RL implementation presented in Sec. III, we were able to deploy a walking policy that was trained on a serial model of the robot. We validated the learned policy in MuJoCo [31], to ensure that the policy has not been overfit to Isaac Sim. Our simulation uses a model with the closed-loop mechanisms to validate the complete control strategy. Our impedance transfer shown in II-D and the state estimator shown in II-E allow successful deployment on the system, with a computation under 10 ms for both, with pure Python code, allowing updates up to 100Hz on a Raspberry PI 5 onboard computer. We leverage the use of the low level control loop of the actuators to stabilize the system, that runs at a way higher rate (typically thousands of Hz). This allows efficient tracking of the serial space reference as showcased in Fig. 11. To validate our method, we also tested to send the reference torques  $\tau_m^*$  directly to the actuators, which led to dramatic failure, causing the robot to fall.

The walking movements are reported in Fig. 12 and the companion video. This result highlight the potential of the method in handling a wide range of operational scenarios, even beyond restrictive limits on serial joints. Our method allows seamless transfer using the impedance gain conversion, avoiding the drawbacks and difficulties of training with simulated chain closures.

We demonstrated that ankle motor limits play a significant role in more complex motions, such as locomotion on uneven terrain. Nevertheless, these limits were not incorporated into our RL experiments, as our evaluation focuses solely

on flat-ground walking. However, our code (which will be released as open source) provides GPU implementations of the models  $f$  and  $J_A$  using CusAdi [32], thereby facilitating the integration of such limits into RL frameworks.

## V. CONCLUSION

We presented a framework that extends both MPC and RL locomotion methods to robots equipped with closed-kinematic actuators, while introducing minimal computational overhead. Our approach builds on a differential geometric description of two common transmission mechanisms (knee and ankle), providing analytical expressions of the actuation model and its derivatives. This enables efficient derivative-based trajectory optimization, as well as the transfer of impedance gains from joint space to actuator space, allowing RL policies trained in serial space to be deployed directly on hardware. We validated the method in trajectory optimization on challenging locomotion tasks and demonstrated successful policy transfer to a real robot using the variable impedance gains, which has not been possible without. We have focused our empirical analysis on explaining and showcasing the importance of modeling the transmission, compared to only modeling the serial kinematics. Yet the model also offers evident advantages in computational complexity, involving negligible additional computation in addition to the serial model, compared to the extra complexity of modeling the entire constrained kinematics in exhaustive approaches. By delivering practical implementations in both TO and RL, our work highlights how compact transmission models can enhance control accuracy and robustness in modern legged robots, and opens opportunities for future designs with richer non-linear actuation mechanisms.

## REFERENCES

- [1] V. Batto, T. Flayols, N. Mansard, and M. Vulliez, "Comparative metrics of advanced serial/parallel biped design and characterization of the main contemporary architectures," *IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023.
- [2] D. Mronga, S. Kumar, and F. Kirchner, "Whole-body control of series-parallel hybrid robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.
- [3] M. Boukheddimi, R. Kumar, S. P. Kumar, J. Carpentier, and F. Kirchner, "Investigations into exploiting the Full Capabilities of a Series-Parallel Hybrid Humanoid Using Whole Body Trajectory Optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.
- [4] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.
- [5] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and sparse resolution of constrained dynamic equations," in *Robotics: Science and Systems XVII*, 2021.
- [6] S. Kumar, J. Martensen, A. Mueller, and F. Kirchner, "Model Simplification For Dynamic Control of Series-Parallel Hybrid Robots - A Representative Study on the Effects of Neglected Dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [7] L. De Matteis, V. Batto, J. Carpentier, and N. Mansard, "Optimal Control of Walkers with Parallel Actuation," in *International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- [8] A. S. Sathya and J. Carpentier, "Constrained articulated body dynamics algorithms," *IEEE Transactions on Robotics*, 2025.
- [9] S. Kumar and A. Mueller, "An analytical and modular software workbench for solving kinematics and dynamics of series-parallel hybrid robots," *Journal of Mechanisms and Robotics*, 2019.
- [10] C. Zhou and N. Tsagarakis, "On the comprehensive Kinematics Analysis of a Humanoid Parallel Ankle Mechanism," *Journal of Mechanisms and Robotics*, 2018.
- [11] E. M. Hoffman *et al.*, "Modeling and numerical analysis of kangaroo lower body based on constrained dynamics of hybrid serial-parallel floating-base systems," *Robotics and Autonomous Systems*, 2024.
- [12] Y. Liang *et al.*, "Reduced-Dimensional Whole-Body Control Based on Model Simplification for Bipedal Robots With Parallel Mechanisms," *IEEE Robotics and Automation Letters*, 2025.
- [13] E. Maslennikov *et al.*, "Robust RL Control for Bipedal Locomotion with Closed Kinematic Chains," 2025.
- [14] Q. Liao, B. Zhang, X. Huang, X. Huang, Z. Li, and K. Sreenath, "Berkeley humanoid: A research platform for learning-based control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [15] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *The International Journal of Robotics Research*, 2025.
- [16] Y. Tanaka, A. Zhu, Q. Wang, and D. Hong, "Mechanical Intelligence-Aware Curriculum Reinforcement Learning for Humanoids with Parallel Actuation," 2025.
- [17] F. Amadio, H. Li, L. Uttini, S. Ivaldi, V. Modugno, and E. Mingo Hoffman, "Learning to Walk with Hybrid Serial-Parallel Linkages: a Case Study on the Kangaroo Robot," in *13th International Conference on Robot Intelligence Technology and Applications*, 2025.
- [18] Q. L. Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact models in robotics: a comparative analysis," *IEEE Transactions on Robotics*, 2024.
- [19] Q. Zhang *et al.*, "LiPS: Large-Scale Humanoid Robot Reinforcement Learning with Parallel-Series Structures," 2025.
- [20] F. Ruscelli, A. Laurenzi, E. Mingo Hoffman, and N. G. Tsagarakis, "A fail-safe semi-centralized impedance controller: Validation on a parallel kinematics ankle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [21] Unitree - h1. Unitree. <https://www.unitree.com/h1>, Accessed on 2025-02-28. [Online]. Available: <https://www.unitree.com/h1>
- [22] O. Stasse *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017.
- [23] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems (RSS)*, 2018.
- [24] J. Carpentier *et al.*, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *SII - International Symposium on System Integrations*, 2019.
- [25] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-Order Sampling-Based MPC for Torque-Level Locomotion Control via Diffusion-Style Annealing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [26] 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: optimization and feedback control*. Springer, 2006.
- [27] C. Mastalli *et al.*, "Crocodyl: An efficient and versatile framework for multi-contact optimal control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [28] E. Dantec *et al.*, "Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022.
- [29] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.
- [30] C. Roux *et al.*, "Constrained reinforcement learning for unstable point-foot bipedal locomotion applied to the bolt robot," in *IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, 2025.
- [31] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
- [32] S. H. Jeon, S. Hong, H. J. Lee, C. Khazoom, and S. Kim, "CusADi: A GPU Parallelization Framework for Symbolic Expressions and Optimal Control," *IEEE Robotics and Automation Letters*, 2024.