POLITECNICO DI TORINO
Repository ISTITUZIONALE

Stabilizing Humanoid Robot Trajectory Generation via Physics-Informed Learning and Control-Informed Steering

Original
Stabilizing Humanoid Robot Trajectory Generation via Physics-Informed Learning and Control-Informed Steering / D'Elia, E., Viceconte, P.M., Rapetti, L., Ferigo, D., Romualdi, G., L'Erario, G., Camoriano, R., Pucci, D.. - (2025), pp. 4781-4788. (IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Hangzhou (CHN) 19-25 October 2025) [10.1109/IROS60139.2025.11247546].

Availability:
This version is available at: 11583/3017201 since: 2026-10-29T12:21:33Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/IROS60139.2025.11247546

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

10 November 2026

arXiv:2509.24697v1 [cs.RO] 29 Sep 2025

Stabilizing Humanoid Robot Trajectory Generation via Physics-Informed Learning and Control-Informed Steering

Evelyn D'Elia[1,2], Paolo Maria Viceconte[1], Lorenzo Rapetti[1], Diego Ferigo[1], Giulio Romualdi[1], Giuseppe L'Erario[1,2], Raffaello Camoriano[3,4], and Daniele Pucci[1,2]

***Abstract*— Recent trends in humanoid robot control have successfully employed imitation learning to enable the learned generation of smooth, human-like trajectories from human data. While these approaches make more realistic motions possible, they are limited by the amount of available motion data, and do not incorporate prior knowledge about the physical laws governing the system and its interactions with the environment. Thus they may violate such laws, leading to divergent trajectories and sliding contacts which limit real-world stability. We address such limitations via a two-pronged learning strategy which leverages the known physics of the system and fundamental control principles. First, we encode physics priors during supervised imitation learning to promote trajectory feasibility. Second, we minimize drift at inference time by applying a proportional-integral controller directly to the generated output state. We validate our method on various locomotion behaviors for the ergoCub humanoid robot, where a physics-informed loss encourages zero contact foot velocity. Our experiments demonstrate that the proposed approach is compatible with multiple controllers on a real robot and significantly improves the accuracy and physical constraint conformity of generated trajectories.**

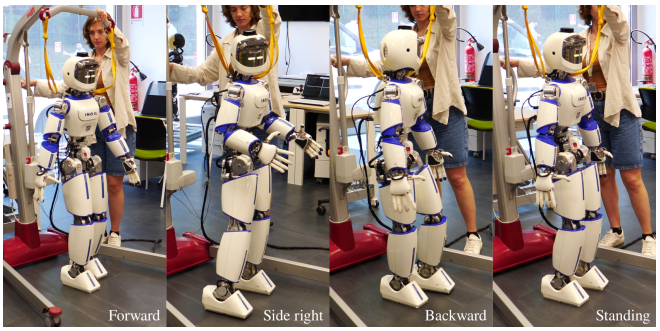


Fig. 1: Our method used to execute various walking directions in a single trajectory on the ergoCub robot.

I. Introduction

When working in collaborative, human-centric environments, humanoid robots need to perform in a smooth, safe fashion. Moreover, in collaborative scenarios it is important for the robot's motions to be *human-like*, such that the human can trust, understand, and predict its behavior [1], [2], as well as to improve the robot's performance in tasks designed for humans. However, human-likeness is not a clearly defined property, and enforcing it in trajectory generation by explicit modeling is impractical due to high computational cost thanks to the highly nonlinear dynamics of a humanoid robot.

Human-like motion cannot be achieved without a control architecture that is able to stabilize the desired robot behavior. This is challenging due to the large number of degrees of freedom and complex stability constraints. Many state-of-the-art approaches exploit a hierarchical structure to address the problem: *trajectory optimization*, which generates desired trajectories, *trajectory adjustment*, which stabilizes the trajectory, and *trajectory control*, which solves a quadratic programming (QP) problem to translate the trajectory into motor references [3].

Humanoid trajectory optimization is categorized as either *model-based* or *model-free*. *Model-based* techniques, which employ an explicit model of the system, may describe the full dynamics or a simplified/reduced version. Full dynamics approaches achieve high-fidelity motions, but require meticulous design and are computationally expensive. Simplified model-based approaches, such as representing locomotion with an inverted pendulum model [4], or centroidal dynamics models [5], [6], adopt a low-dimensional parameterization of the complex motion of a humanoid to improve optimization efficiency. However, the use of simplified models for trajectory generation hinders their human-likeness.

Alternatively, *model-free* approaches use optimization methods to directly approximate training data without the use of an explicit model, allowing them to achieve better computational efficiency than model-based methods. One class of trajectory generation solutions which reduce model dependence is *imitation learning (IL)*. One special case of IL, which relies entirely on mimicking the dataset, is called *behavior cloning (BC)* and has been applied to achieve legged locomotion trajectory planning in simulation [7], [8], [9] and more recently on real robots [10], [11] by harnessing task-specific training data. IL has also been employed to supplement the reward for learning human-like policies with reinforcement learning (RL) [12], [13], [14], [15], [16]. In

*This work was supported by the ergoCub project.
RC acknowledges the following: this study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

[1]Artificial and Mechanical Intelligence, Italian Institute of Technology, 16163 Genoa, Italy evelyn.delia@iit.it
[2]Machine Learning and Optimisation, University of Manchester, M13 9PL Manchester, U.K.
[3]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Turin, Italy
[4]Rehab Technologies Lab, Italian Institute of Technology, 16163 Genoa, Italy

recent years, the use of IL in an RL context has led to robust real-world humanoid control [17], [18], [19].

The advantages of IL over RL include a much better sample efficiency, a simpler supervised learning problem, and reduced computational requirements. However, IL is limited by the quantity and quality of available demonstration data [20]. As a result, trajectories generated via IL can break physical laws and exhibit asymmetry or bias toward certain modes. To overcome this weakness, in recent years *physics-informed machine learning (PIML)* has been proposed [21]. PIML incorporates priors about the physical system dynamics into the learning process, thus combining the strengths of model-based and learning-based methods. A key strength of PIML lies in its ability to encode *differentiable* priors during learning, which can speed up convergence and reduce the amount of data needed to infer the underlying physical laws. This strategy is rendered computationally feasible by automatic differentiation (AD), which is nowadays available in most machine learning libraries.

PIML proved effective at accurately learning to emulate partial differential equations (PDEs) to model complex physical systems [22], [23]. In the context of learning for humanoid control, incorporating physics priors as part of the reward function has been shown in simulation to counteract learned artifacts related to ground contacts in humanoid trajectory generation [24]. However, the physical reward components in [24] are not differentiable and thus cannot influence backpropagation. Additionally, smooth humanoid locomotion has been achieved on real-world robots with PIML by formulating a Lipschitz constraint as a reward function [25]. However, this approach does not use IL and the resulting policies are less human-like.

In summary, BC methods have achieved notable success in trajectory generation and control for humanoids in recent years. In parallel, PIML has been shown to improve physical fidelity when training examples are scarce or unevenly distributed. Therefore, the incorporation of PIML into BC-based humanoid locomotion offers significant potential to improve the stability of generated trajectories while preserving the human-likeness learned from the data.

In this work, we propose a modular PIML and control theory-inspired framework to improve physical fidelity and mitigate learned bias of BC for a humanoid robot. First, we introduce a supervised learning approach to predict the full next state of the robot given the previous state and a user-provided desired direction input. We then propose improvements which enhance the symmetry and physical fidelity of the final trajectory. We emphasize that our contributions are modular. Our contributions are summarized as follows:

- We present a framework to learn the full kinematic state of a floating-base robot while considering kinematically feasible contacts via a *physics-informed* loss, in addition to a standard data-fitting loss term. Specifically, the network learns to keep the contact foot still while the swing foot moves.
- We harness fundamental control concepts to impose a corrective term on the output of our network. This *control-informed* blending of user input with network predictions reduces the drift of generated trajectories.
- We perform ablation studies and real-world validation experiments on the ergoCub robot using two control architectures, highlighting the efficacy and modularity of our approach.

This paper is structured as follows: Section II introduces the relevant background information, Section III describes the details of our approach, Section IV contains an analysis and discussion of experimental results, and Section V summarizes our contribution and proposes future work.

II. BACKGROUND

A. Notation

- $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{B}}$ denote the nonlinear differential and boundary condition operators for an arbitrary differential equation.
- $\boldsymbol{u}(\boldsymbol{z})$ indicates the differential equation solution as a function of the state and time vector $\boldsymbol{z} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d; t]$.
- $\boldsymbol{f}(\boldsymbol{z})$ and $\boldsymbol{g}(\boldsymbol{z})$ are, respectively, the functions defining the data dynamics and boundary conditions.
- $\mathcal{L}(\boldsymbol{z}; \theta)$ is a loss function computed with respect to all unknown parameters θ.
- $\mathcal{B}$ and $\mathcal{I}$ are the robot base and inertial frames, where $\mathcal{B}$ is located at the center of the waist with the x axis pointing forward and the z axis pointing upwards.
- ${}^{\mathcal{A}}\boldsymbol{R}_{\mathcal{C}}$ is the matrix which describes the rotation from frame $\mathcal{A}$ to frame $\mathcal{C}$.
- $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega}) \in \mathbb{R}^6$ is a 6D velocity vector, where $\boldsymbol{v}$ is the linear velocity, $\boldsymbol{\omega}$ is the angular velocity.
- $\boldsymbol{s}, \dot{\boldsymbol{s}} \in \mathbb{R}^n$ are the joint positions and velocities, respectively, where n is the number of considered joints.
- $\boldsymbol{q} = (\boldsymbol{q}_{\mathcal{B}}, \boldsymbol{s}) \in \mathbb{R}^{6+n}$ is the combined position vector of the floating base model, where $\boldsymbol{q}_{\mathcal{B}} = (\boldsymbol{p}_{\mathcal{B}}, \boldsymbol{\psi}_{\mathcal{B}})$ contains the floating base position $\boldsymbol{p}_{\mathcal{B}}$ and extrinsic Tait-Bryan angles $\boldsymbol{\psi}_{\mathcal{B}}$.
- $\boldsymbol{\nu} = (\mathbf{v}_{\mathcal{B}}, \dot{\boldsymbol{s}}) \in \mathbb{R}^{6+n}$ is the combined velocity vector of the floating base model.
- $\mathbf{J}$ is the Jacobian matrix, which is a function of the robot state, and describes the relationship between $\boldsymbol{\nu}$ and $\mathbf{v}$ at each robot link.
- The operator $(\cdot)^\top$ takes the transpose of a matrix.
- A variable with a hat $\hat{(\cdot)}$ denotes a predicted value.

B. Physics-Informed Machine Learning

PIML is based on injecting prior knowledge into the learning process to promote solutions which comply with the physical laws governing the system. This can be particularly useful for learning high-dimensional systems, especially when the amount of data is limited, since PIML can compensate for the curse of dimensionality by relying on known system properties [20]. As a result, PIML is promising for humanoid control, where the interactions with the environment are high-dimensional, non-linear, and difficult to fully capture in a finite dataset. Using physics-informed (PI) methods, the gradient of quantities involved in humanoid dynamics, such as the Jacobian, are computed easily with

AD, thus allowing them to directly influence learning. This is in contrast to many traditional RL methods, which may integrate physical laws via reward terms, but do not leverage the gradient. In this paper, we implement PIML via the loss function, exploiting AD to compute derivatives of the PI loss during backpropagation. We define a differential equation and its initial conditions based on [26, Sec. II]:

$$\begin{aligned} \boldsymbol{\mathcal{F}}(\boldsymbol{u}(\boldsymbol{z});\gamma) &= \boldsymbol{f}(\boldsymbol{z}) \quad \boldsymbol{z} \in \Omega, \\ \boldsymbol{\mathcal{B}}(\boldsymbol{u}(\boldsymbol{z})) &= \boldsymbol{g}(\boldsymbol{z}) \quad \boldsymbol{z} \in \partial\Omega, \end{aligned} \tag{1}$$

where γ is the set of physical parameters of the nonlinear differential equation, Ω is a manifold, and $\partial\Omega$ is the manifold's boundary. A physics-informed neural network (PINN) $\hat{\boldsymbol{u}}_\theta(\boldsymbol{z})$ aims to approximate the solution such that

$$\hat{\boldsymbol{u}}_\theta(\boldsymbol{z}) \approx \boldsymbol{u}(\boldsymbol{z}), \tag{2}$$

by minimizing a physics-informed loss taking the following form with respect to NN weights θ:

$$\mathcal{L}(\boldsymbol{z};\theta) = w_{\boldsymbol{\mathcal{F}}}\mathcal{L}_{\boldsymbol{\mathcal{F}}}(\boldsymbol{z};\theta) + w_{\boldsymbol{\mathcal{B}}}\mathcal{L}_{\boldsymbol{\mathcal{B}}}(\boldsymbol{z};\theta) + w_{\boldsymbol{\mathcal{D}}}\mathcal{L}_{\boldsymbol{\mathcal{D}}}(\boldsymbol{z};\theta), \tag{3}$$

where the $\boldsymbol{\mathcal{F}}$ component depends on the differential equation, the $\boldsymbol{\mathcal{B}}$ component on the boundary conditions, and the $\boldsymbol{\mathcal{D}}$ component is the empirical loss on the training data. Specifically, the individual loss components are formulated as follows:

$$\mathcal{L}_{\boldsymbol{\mathcal{F}}}(\boldsymbol{z};\theta) = \frac{1}{N_F}\sum_{i=1}^{N_F} \|\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{u}}_\theta(\boldsymbol{z}_i);\gamma) - \boldsymbol{f}(\boldsymbol{z}_i)\|^2, \tag{4}$$

$$\mathcal{L}_{\boldsymbol{\mathcal{B}}}(\boldsymbol{z};\theta) = \frac{1}{N_B}\sum_{i=1}^{N_B} \|\boldsymbol{\mathcal{B}}(\hat{\boldsymbol{u}}_\theta(\boldsymbol{z}_i)) - \boldsymbol{g}(\boldsymbol{z}_i)\|^2, \tag{5}$$

$$\mathcal{L}_{\boldsymbol{\mathcal{D}}}(\boldsymbol{z};\theta) = \frac{1}{N_d}\sum_{i=1}^{N_d} \|\hat{\boldsymbol{u}}_\theta(\boldsymbol{z}_i^*) - \boldsymbol{u}_i^*\|^2, \tag{6}$$

where N_F, N_B, and N_d are the numbers of differential equations, boundary conditions, and training set points, respectively, and $(\cdot)^*$ denotes ground-truth values. When physics-informed loss components $\mathcal{L}_{\boldsymbol{\mathcal{F}}}$ and $\mathcal{L}_{\boldsymbol{\mathcal{B}}}$ are not included, Eq. (3) simplifies to $\mathcal{L}(\theta) = w_{\boldsymbol{\mathcal{D}}}\mathcal{L}_{\boldsymbol{\mathcal{D}}}(\theta)$, which corresponds to the mean squared error (MSE) loss often used in supervised regression.

C. Mode-Adaptive Neural Networks

In this work, we adopt Mode-Adaptive Neural Networks (MANN) [9] as a supervised learning architecture for trajectory generation. MANN is well-suited to learning a variety of locomotion modes for legged models. The MANN input, $\mathbf{x}_i$, is composed of selected model state information as well as user inputs. From $\mathbf{x}_i$, MANN predicts the desired legged model state at the next time step $\mathbf{x}_{i+1}$. The MANN architecture consists of 2 neural network (NN) components: a gating network and a motion prediction network. The gating network employs the Mixture of Experts (MoE) paradigm [27] and computes a set of K blending coefficients $\boldsymbol{\theta}_i = \{\theta_{i1}, \ldots, \theta_{iK}\}$ in the feature space, given a subset $\hat{\mathbf{x}}_i$ of the full NN input $\mathbf{x}_i$. Then, the motion prediction network weights $\hat{\alpha}_i = \sum_{j=1}^{K} \theta_{ij}\boldsymbol{\alpha}_j$ are computed using the predicted blending coefficients, with the K expert weights $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K\}$. With these weights and the model state and a user input, the motion prediction network predicts the state at the next time step, $\mathbf{y}_i$. In the original MANN implementation, the training loss function is simply the mean squared error (MSE) between the predictions and the labels in the dataset, i.e., $\mathcal{L}_{\boldsymbol{\mathcal{D}}}(\boldsymbol{z};\theta)$. We remark that prior work employing MANN for locomotion trajectory generation only considered this data-fitting loss term [28], [10].

D. Trajectory Controllers

To execute the generated trajectories on the robot, multiple types of controllers can be adopted. In humanoid robotics, a nested control architecture is typically employed consisting of 3 layers: *trajectory generation*, *trajectory adjustment*, and *trajectory control* [3]. The *trajectory generation* layer parametrizes the high-level path for the robot to follow in terms of contact, joint, and centroidal dynamics [29]. In our work, the trajectory is generated via our physics-informed BC approach.

We then stabilize generated trajectories with both an offline and online controller. This section describes the two control strategies.

1) Offline control: The offline controller makes use of the full trajectory generated by the *trajectory generation* layer. The offline *trajectory adjustment* layer stabilizes the zero moment point (ZMP) position, center of mass (CoM) position, and CoM velocity over the duration of the trajectory. The final *whole-body quadratic programming (QP) control* layer solves for the optimal robot velocity over the full trajectory given a set of tasks. Details of this control scheme can be found in [30].

2) Online control: Unlike the offline controller, the online controller performs the *trajectory generation* step online during deployment. the *trajectory adjustment* step is formulated as a model-predictive control (MPC) optimization problem, which stabilizes a trajectory that has been generated at that iteration for the future horizon. The MPC problem adjusts the centroidal momentum and contacts based on the future horizon prediction from the *trajectory generation* layer as well as constraints placed on the momentum and contacts for stability. Finally, at each iteration the *trajectory control* layer uses robot feedback and outputs of the previous two layers to track the desired ZMP and CoM, generate swing foot trajectories, and finally compute the joint values with a QP optimization problem. Details are present in [29].

III. METHODS

This paper presents two main methodological contributions. First, we propose a *physics-informed* architecture for learning the kinematic state of a floating base system subject to constraints. Second, we propose a *control-informed* correction scheme which mitigates the drift of the predictions and steers them towards desired values, without the need for retraining or modifying the learning architecture itself. Figure 2 summarizes how the two proposed approaches fit into the overall trajectory generation and control scheme which we adopt to validate their efficacy.

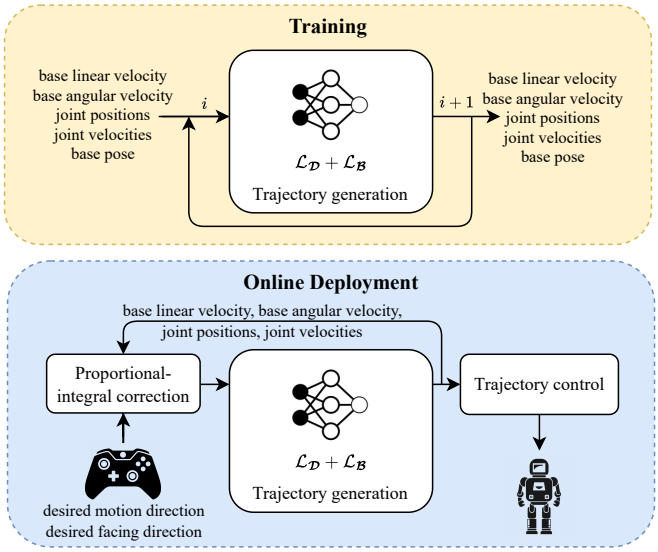


Fig. 2: Overall humanoid locomotion architecture integrating the proposed PI loss and proportional-integral correction block.

A. Training features

We define the input and output features such that the full kinematic state $(\boldsymbol{q}, \boldsymbol{\nu})$ is learned. Thus, the input feature vector $\mathbf{x}_i$ is structured as:

$$\mathbf{x}_i = \{{}^{\mathcal{B}}\mathbf{V}_{\mathcal{I},\mathcal{B},i-1}, {}^{\mathcal{B}}\boldsymbol{\Omega}_{\mathcal{I},\mathcal{B},i-1}, \boldsymbol{s}_{i-1}, \dot{\boldsymbol{s}}_{i-1}, {}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B},i-1}, {}^{\mathcal{I}}\boldsymbol{\psi}_{\mathcal{B},i-1}\} \in \mathbb{R}^{2n+78}, \tag{7}$$

where ${}^{\mathcal{B}}\mathbf{V}_{\mathcal{I},\mathcal{B},i-1}, {}^{\mathcal{B}}\boldsymbol{\Omega}_{\mathcal{I},\mathcal{B},i-1} \in \mathbb{R}^{36}$ are the vectorizations of the 12 base linear and angular velocities, expressed in the body-fixed frame, sampled at 6 Hz over a 2 s window centered at the current timestep i, and ${}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B},i-1}, {}^{\mathcal{I}}\boldsymbol{\psi}_{\mathcal{B},i-1} \in \mathbb{R}^3$ are the base position and orientation measured in the inertial frame at $i-1$. Similarly, the output vector $\mathbf{y}_i$ is expressed as:

$$\mathbf{y}_i = \{{}^{\mathcal{B}}\mathbf{V}_{\mathcal{I},\mathcal{B},i}, {}^{\mathcal{B}}\boldsymbol{\Omega}_{\mathcal{I},\mathcal{B},i}, \boldsymbol{s}_i, \dot{\boldsymbol{s}}_i, {}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B},i}, {}^{\mathcal{I}}\boldsymbol{\psi}_{\mathcal{B},i}\} \in \mathbb{R}^{2n+48}, \tag{8}$$

where ${}^{\mathcal{B}}\mathbf{V}_{\mathcal{I},\mathcal{B},i}, {}^{\mathcal{B}}\boldsymbol{\Omega}_{\mathcal{I},\mathcal{B},i} \in \mathbb{R}^{21}$ are the flattened 7 future body-fixed base velocities over a 1 s window of future information, starting at the current timestep, and ${}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B},i}, {}^{\mathcal{I}}\boldsymbol{\psi}_{\mathcal{B},i} \in \mathbb{R}^3$ contain the base pose information at timestep i. We provide the full, 3D kinematic state to the MANN model, as opposed to previous work which considers only ground projections [9], [28], [10].

B. Physics-informed loss component

We propose a PI loss component to reduce contact foot sliding. This is added to the MSE loss (see Eq. (3)), which penalizes deviations between predicted and actual robot state.

We design the PI loss to encourage the support foot of the humanoid to remain fixed when in contact with the ground:

$$\mathbf{v}_{SF} = \mathbf{J}_{SF}\boldsymbol{\nu} = \begin{bmatrix}\mathbf{J}^{\mathcal{B}}_{SF} & \mathbf{J}^{\dot{\boldsymbol{s}}}_{SF}\end{bmatrix}\begin{bmatrix}\mathbf{v}_{\mathcal{B}} \\ \dot{\boldsymbol{s}}\end{bmatrix} = 0, \tag{9}$$

where SF stands for the support foot, while $\mathbf{J}^{\mathcal{B}}_{SF} \in \mathbb{R}^{6\times 6}$ and $\mathbf{J}^{\dot{\boldsymbol{s}}}_{SF} \in \mathbb{R}^{6\times n}$ indicate the components of the Jacobian that map to the SF velocities from the 6D base velocity and joint velocities, respectively. Eq. (9) can be rearranged to act as a constraint on the velocity of the floating base:

$$\mathbf{v}_{\mathcal{B}} = -(\mathbf{J}^{\mathcal{B}}_{SF})^{-1}\mathbf{J}^{\dot{\boldsymbol{s}}}_{SF}\dot{\boldsymbol{s}}, \tag{10}$$

where $\mathbf{J}^{\mathcal{B}}_{SF}$ is an upper triangular block matrix whose diagonal blocks are rotation matrices, and is thus always invertible.

We formulate the PI loss component as:

$$\mathcal{L}_{\mathcal{B}}(\mathbf{x};\theta) = \left\| \begin{bmatrix}{}^{\mathcal{B}}\hat{\boldsymbol{v}}_{\mathcal{I},\mathcal{B}} \\ {}^{\mathcal{B}}\hat{\boldsymbol{\omega}}_{\mathcal{I},\mathcal{B}}\end{bmatrix}(\mathbf{x};\theta) + \big[\alpha(\mathbf{J}^{\mathcal{B}}_{LF})^{-1}\mathbf{J}^{\dot{\mathbf{s}}}_{LF} + (1-\alpha)(\mathbf{J}^{\mathcal{B}}_{RF})^{-1}\mathbf{J}^{\dot{\mathbf{s}}}_{RF}\big]\hat{\dot{\mathbf{s}}}(\mathbf{x};\theta)\right\|^2, \tag{11}$$

where $\mathbf{x}$ is the input feature vector, and α is a binary variable that is set to 0 when the right foot (RF) is the support foot, or 1 when the left foot (LF) is the support foot.

The PI loss term is included in the overall loss function $\mathcal{L}(\mathbf{x};\theta)$, together with the data-fitting loss $\mathcal{L}_{\mathcal{D}}$:

$$\mathcal{L}(\mathbf{x};\theta) = \mathcal{L}_{\mathcal{D}}(\mathbf{x};\theta) + w\mathcal{L}_{\mathcal{B}}(\mathbf{x};\theta), \tag{12}$$

where $w \equiv w_{\mathcal{B}}$ is the positive scalar weight given to the PI loss function component during training of the network. We set $w_{\mathcal{D}} = 1$ to simplify parameter tuning.

C. Control-informed steering for network drift correction

One drawback of IL methods is that due to biases in training data, a trained model may produce asymmetric outputs and thus cause the trajectory to drift from the desired direction. Assuming an arbitrary learning model which predicts the kinematic state $(\boldsymbol{q}, \boldsymbol{\nu})$, such as the one depicted in the top of Figure 2, given the previous state, we propose closing the loop on $\boldsymbol{\nu}$ using the desired user input $\boldsymbol{q}_{\mathcal{B},d}$. This solution requires no changes to the training or inference of the learned model, and allows the user specifications to directly steer the trajectory. This correction block removes drift by counteracting learned biases and enforces additional desired dynamics into the system to naturally slow down and stop the walking at the goal position and orientation.

Our drift correction is composed of a linear part and an angular part. The linear velocity controller is formulated as:

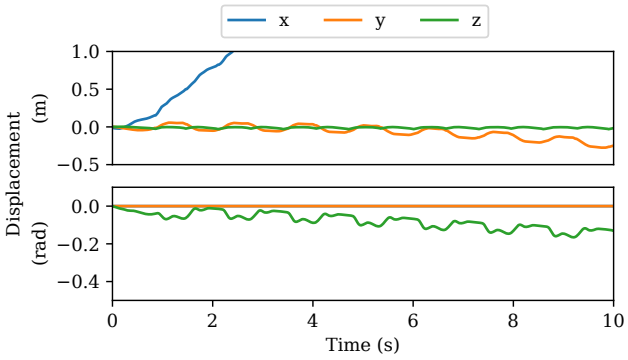
$${}^{\mathcal{B}}\boldsymbol{v}^{c}_{\mathcal{I},\mathcal{B}} = {}^{\mathcal{B}}\hat{\boldsymbol{v}}_{\mathcal{I},\mathcal{B}} - k_0\left({}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}\right)^{\top}\left({}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B}} - {}^{\mathcal{I}}\boldsymbol{p}^{d}_{\mathcal{B}}\right), \tag{13}$$

where ${}^{\mathcal{B}}\boldsymbol{v}^{c}_{\mathcal{I},\mathcal{B}}$ denotes the linear base velocity control output for the next network input, ${}^{\mathcal{B}}\hat{\boldsymbol{v}}_{\mathcal{I},\mathcal{B}}$ is the network-predicted linear base velocity, k_0 is the positional error gain, ${}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}$ is the current base orientation, and ${}^{\mathcal{I}}\boldsymbol{p}_{\mathcal{B}}$ and ${}^{\mathcal{I}}\boldsymbol{p}^{d}_{\mathcal{B}}$ are the current and desired base positions.

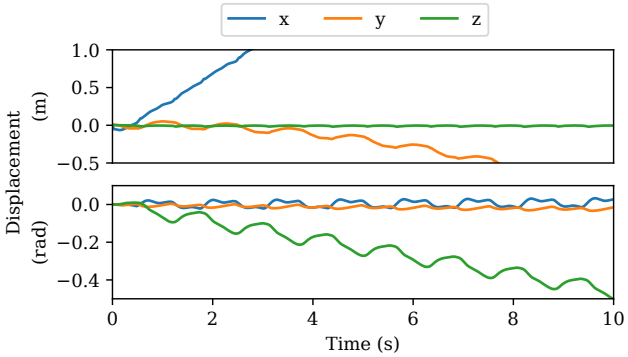
Instead, the rotational velocity controller is defined based on [31, Sec. III.E] and [32, Sec. 5.11.6], such that the generated state approaches the desired base rotation ${}^{\mathcal{I}}\boldsymbol{R}^{d}_{\mathcal{B}}$:

$${}^{\mathcal{B}}\boldsymbol{\omega}^{c}_{\mathcal{I},\mathcal{B}} = {}^{\mathcal{B}}\hat{\boldsymbol{\omega}}_{\mathcal{I},\mathcal{B}} - k_1\mathrm{skew}\left({}^{\mathcal{I}}\boldsymbol{R}^{d\top}_{\mathcal{B}}{}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}\right)^{\vee}, \tag{14}$$

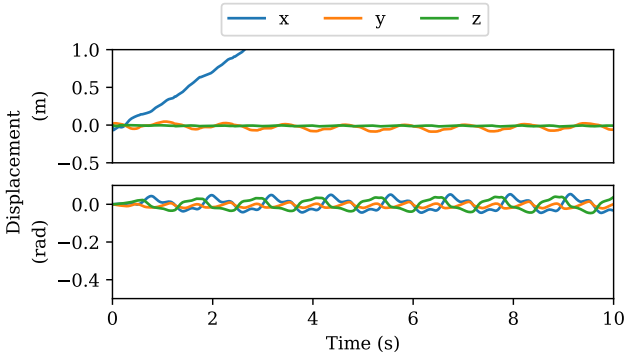
where ${}^{\mathcal{B}}\boldsymbol{\omega}^{c}_{\mathcal{I},\mathcal{B}}$ becomes the angular base velocity for the MANN input at the next time step, ${}^{\mathcal{B}}\hat{\boldsymbol{\omega}}_{\mathcal{I},\mathcal{B}}$ is the predicted angular velocity, k_1 is the positional error gain, and $\mathrm{skew}(\cdot)^{\vee}$ is the vectorized skew-symmetric part of a square matrix.



(a) Base displacement for a baseline model [10]. In that formulation, the x and y base rotations were constrained to 0.

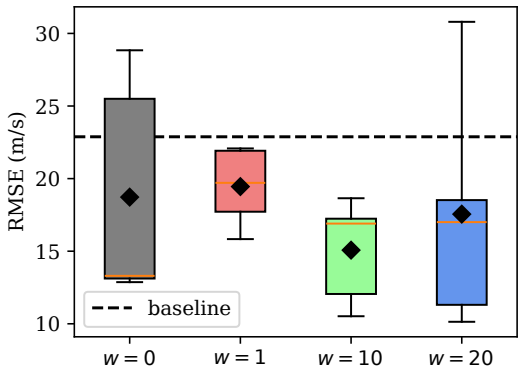


(b) Base displacement when correction block is disabled.

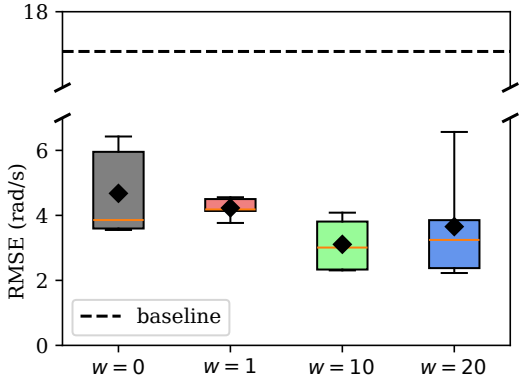


(c) Base displacement when correction block is enabled.

Fig. 3: Comparison of the drift in base position (top) and orientation (bottom) for (a) baseline trajectory generation, (b) our approach without drift correction block, and (c) our approach with drift correction block for forward walking. All values are measured w.r.t. the inertial frame, and the robot starts at the inertial origin, facing the positive x direction.



(a) Average foot linear velocity RMSE.



(b) Average foot angular velocity RMSE.

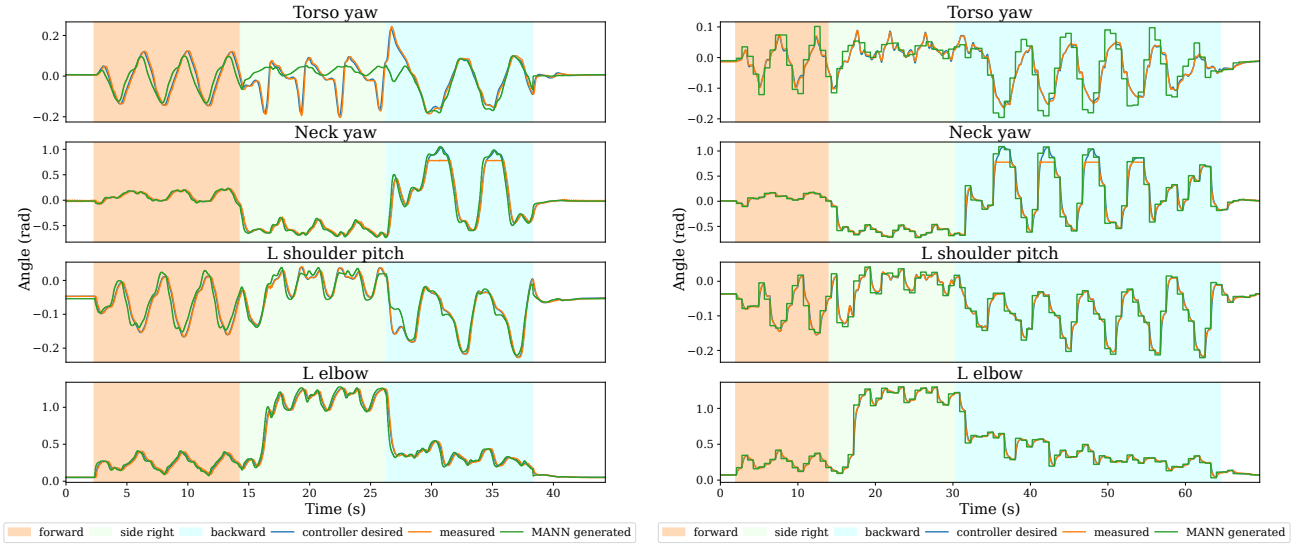
Fig. 4: Comparison of the average total RMSE of the velocity of the support foot during a 10 s forward walking trajectory, for $w \in \{0, 1, 10, 20\}$ ($w = 0$ means $\mathcal{L}_{\mathcal{D}}$ only). Medians are represented by orange lines and means as black diamonds. Baseline approach is from [10].

IV. RESULTS AND DISCUSSION

For validation of the proposed approach, we train the proposed model on 30 minutes of multi-directional human locomotion data using the XSens [33] sensor suit. The dataset size is doubled by mirroring the base pose across the world xz plane and joint positions across the sagittal plane. In total, this corresponds to roughly $200,000$ data points sampled at 50 Hz, which are then retargeted onto the 160 cm, 56 kg ergoCub humanoid robot, considering $n = 26$ joints. We use the same dataset and retargeting method as [10].

MANN model training is carried out using PyTorch with 4 experts over the course of 150 epochs, with the data downsampled to 50 Hz and making use of vectorized batch computations for higher efficiency. The single hidden layer of the gating network contains 32 neurons while that of the prediction network has 512. We use the AdamW optimizer to train the model [34], requiring approximately 24 hours on a single NVIDIA GeForce RTX 3050 Ti GPU. Experimental code and datasets are available at: `https://github.com/ami-iit/paper_delia_2025_iros_physics-informed_trajectory_generation`.

In order to assess the capabilities of our proposed approach, we perform three experiments: i) a comparison of the generated trajectories with and without our physics-informed loss and correction block, ii) an empirical analysis of our approach on the ergoCub robot combined with both an offline and an online controller, and iii) an ablation study isolating the effects of the proposed correction block and the PI loss component on the generated trajectories.



(a) Offline DCM controller results at 0.33x speed and 0.4x step size.

(b) Online MPC controller results at 0.25x speed and 0.3x step size.

Fig. 5: Side-by-side of selected joint tracking for two different controllers when deployed on the ergoCub robot. Data in each case contains a mixed walking trajectory with smooth transitions: standing, forward, side, backward, and standing again.

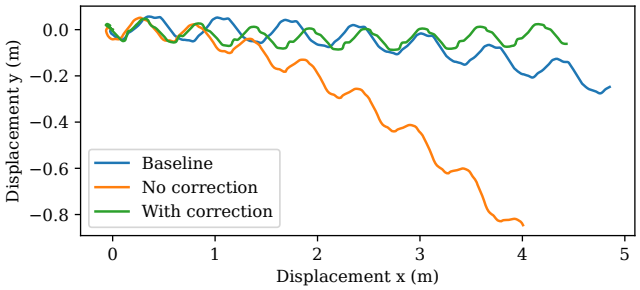


Fig. 6: Ground-projected base displacements for the baseline model, our model without the correction, and our model with the correction, for a 10 s forward walking trajectory.

A. *Performance of our approach*

We compare models trained using our approach to one trained using the MANN implementation presented in [10] as a baseline for comparison. We observe that our approach produces measurable improvements in the physical feasibility and how effectively the trajectories adhere to user input. In particular, as shown in Figure 3, the deviation from the desired walking direction is significantly reduced between the baseline (a) and our approach with the correction block (c). In particular, we observe the y linear and z angular drift. The x displacement is expected due to the walking direction.

Notably, the drift for our model without a correction block (b) is larger than that of the baseline model (a). This is because, unlike the baseline, the new model allows for variations in the base roll and pitch, making the model more versatile. Nonetheless, the drift is successfully counteracted when our proposed correction block is activated.

In terms of foot sliding, models trained with our proposed approach display clear advantages. In fact, the baseline model results in average support foot RMSE values of 22.88 m/s for the linear velocities and 17.74 rad/s for the angular velocities. As shown in Figure 4, such errors are significantly larger than those obtained by the proposed approach across the considered PI loss weights $w \in \{0, 1, 10, 20\}$, especially for the angular velocities. Accordingly, reduced support foot sliding translates to better stability during locomotion.

B. *Real-world experiments on the ergoCub robot*

To demonstrate the modularity and successful real-world application of our proposed approach, we deploy our trajectory generator with two different tracking controllers on the ergoCub humanoid robot. The two controllers, described in Section II-D, perform slightly differently, yet ultimately track the generated trajectories successfully as shown in the accompanying video.

Figure 5 compares the tracking of generated upper-body joint trajectories for multiple walking directions and transitions between the two controllers. When testing on the real robot, it is sometimes necessary to scale the footstep size or the trajectory speed to improve dynamic stability. As reported in the captions of Figure 5, the offline controller achieves slightly faster walking and slightly larger step sizes. Furthermore, the joint trajectory tracking of the offline controller is noticeably smoother than that of the online controller. This may be because the online controller's *trajectory control* block runs at a much lower frequency, which reduces its reaction speed to fast motions, resulting in slightly smaller and slower steps. An interesting emergent human-like behavior is MANN's generation of backward-facing neck configurations when following a backward walking trajectory. With both

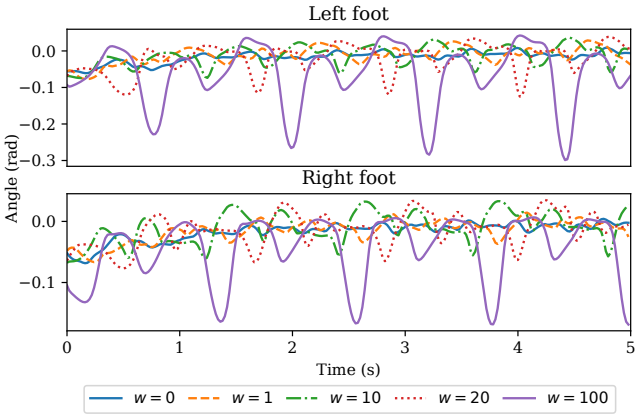


Fig. 7: Left and right foot pitch values over time for a forward walking trajectory, based on the PI loss weight used.

control schemes, the controller modifies the MANN output which exceeds the neck yaw limit for backward walking and generates very small torso yaw displacements for side walking.

C. Ablation studies

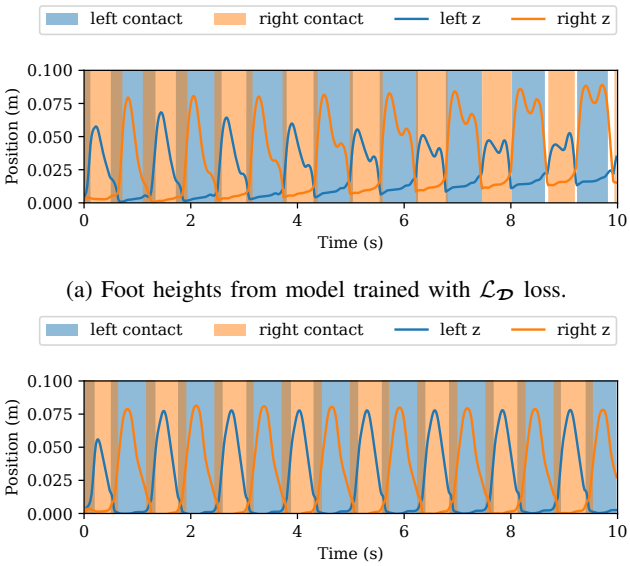
With the aim of quantifying the effects of our proposed techniques on the quality of the generated trajectories, we perform experimental comparisons by selectively deactivating either or both the drift correction block and the PI loss component during training.

1) Drift correction block: We compare the base pose trajectory with and without the correction block to test its efficacy. For a short forward walking trajectory, it is already clear from Figure 3 that the correction block mitigates the drift from the desired forward walking trajectory over 10 s. In particular, the correction eliminates the significant lateral (y) and yaw drift from the raw generated trajectory. Figure 6 also shows that the ground-projected base displacement for the same forward trajectory follows the desired walking direction significantly better with the correction block than without.

2) PI loss component: The efficacy of the PI loss component is evaluated with the drift correction block deactivated, to exclude the confounding factor of gain tuning.

First, we perform a statistical analysis to determine which w results in the highest model performance. Five different models are trained with each of the following weights: $w = \{0, 1, 10, 20, 100\}$. For each model, the epoch with the lowest overall test loss is employed to evaluate the amount of sliding of the support foot during a 10-second forward walking trajectory. Then, the RMSE of the support foot velocity at any given time is summed over the duration of the trajectory. Figure 4 shows that increasing w up to 10 results in a decrease in support foot sliding.

Interestingly, without the correction block, the models trained using the largest PI weight, $w = 100$, are not able to walk at all. Instead, with the correction enabled, walking is achieved, which may indicate that the guidance the correction provides helps overcome undesirable learned artifacts.



(a) Foot heights from model trained with $\mathcal{L}_{\mathcal{D}}$ loss.

(b) Foot heights from model trained with $\mathcal{L}_{\mathcal{D}} + 10\mathcal{L}_{\mathcal{B}}$ loss.

Fig. 8: Comparison of the variation in foot height over 10 s of forward locomotion. Foot height should remain constant when in contact with the ground.

In fact, we observe that when w is too large, the PI loss component overpowers the $\mathcal{L}_{\mathcal{D}}$ component, unexpectedly causing excessive foot rotation to arise, which detracts from the smoothness and human-likeness of the trajectories. Figure 7 displays the generated foot pitch over time for models trained with simple data loss and PI weights of 1, 10, 20, and 100. We note that the pitch deviations become more substantial as the PI weight increases. This may be a result of the PI loss only encouraging the foot to have zero velocity when in contact with the ground, and not explicitly encouraging it to be parallel. If this is the case, the result is that the learned model "cheats" by keeping only the heel on the ground during the terminal stance phase, probably because this helps to keep the foot stationary during the phase where it is most likely to slip. Due to the aforementioned tradeoffs, it was determined that training with $w = 10$ produces models with the best compromise of minimizing contact foot sliding while also retaining stable walking behavior.

Given the above results, we show a comparison of the foot height during forward locomotion between a model trained with $\mathcal{L}_{\mathcal{D}}$ (no PI component) and a model trained with the addition of PI loss with $w = 10$. We can clearly see in Figure 8 that the PI loss significantly decreases support foot displacement and yields more consistent step heights.

V. CONCLUSION AND FUTURE WORK

This paper presents a method which improves the physical fidelity and input-following of IL humanoid trajectory generation. There are two main contributions, a *physics-informed* architecture and a *control-informed* correction block. We have shown that these generalizable improvements can

measurably improve a deep learning approach's adherence to real-world physics laws and desired behavior. Physics-informed learning and corrections to learned predictions result in safer locomotion and more accurate input following for a humanoid robot trajectory generation task. Also, this approach has the potential to be used for many other robotics and machine learning applications due to its modularity.

The presented approach only validates one type of PI implementation. It also requires postprocessing of network outputs in order to sample the contacts needed for the other control layers. Future work will investigate use of PIML with more recent IL methods such as diffusion models [11], [16], and other physics priors, such as ground reaction forces or dynamic stability, in the loss function.

REFERENCES

[1] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, mar 2008.

[2] L. D. Riek, T.-C. Rabinowitch, B. Chakrabarti, and P. Robinson, "How anthropomorphism affects empathy toward robots," in *2009 4th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2009, pp. 245–246.

[3] G. Romualdi, S. Dafarra, Y. Hu, and D. Pucci, "A benchmarking of dcm based architectures for position and velocity controlled walking of humanoid robots," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, Nov. 2018.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, ser. IROS-01. IEEE.

[5] J. Englsberger, C. Ott, and A. Albu-Schaffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, Apr. 2015.

[6] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, Nov. 2014.

[7] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–13, jul 2017.

[8] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, apr 2020.

[9] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, aug 2018.

[10] P. M. Viceconte, R. Camoriano, G. Romualdi, D. Ferigo, S. Dafarra, S. Traversaro, G. Oriolo, L. Rosasco, and D. Pucci, "ADHERENT: Learning human-like trajectory generators for whole-body control of humanoid robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2779–2786, apr 2022.

[11] X. Huang, Y. Chi, R. Wang, Z. Li, X. B. Peng, S. Shao, B. Nikolic, and K. Sreenath, "Diffuseloco: Real-time legged locomotion control with diffusion from offline datasets," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 06–09 Nov 2025, pp. 1567–1589.

[12] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–20, jul 2021.

[13] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–17, jul 2022.

[14] M. Hassan, Y. Guo, T. Wang, M. Black, S. Fidler, and X. B. Peng, "Synthesizing physical character-scene interactions," Feb. 2023.

[15] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba, "Humanmimic: Learning natural locomotion and transitions for humanoid robot via wasserstein adversarial imitation," Sep. 2023.

[16] A. Serifi, R. Grandia, E. Knoop, M. Gross, and M. Bächer, "Robot motion diffusion model: Motion generation for robotic characters," in *SIGGRAPH Asia 2024 Conference Papers*, ser. SA '24. ACM, Dec. 2024, pp. 1–9.

[17] J. Dao, H. Duan, and A. Fern, "Sim-to-real learning for humanoid box loco-manipulation," 2023.

[18] X. Gu, Y.-J. Wang, and J. Chen, "Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer," 2024.

[19] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, Apr. 2024.

[20] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, may 2021.

[21] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, feb 2019.

[22] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, nov 2018.

[23] C. Gin, B. Lusch, S. L. Brunton, and J. N. Kutz, "Deep learning models for global coordinate transformations that linearise PDEs," *European Journal of Applied Mathematics*, vol. 32, no. 3, pp. 515–539, sep 2020.

[24] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz, "Physdiff: Physics-guided human motion diffusion model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[25] Z. Chen, X. He, Y.-J. Wang, Q. Liao, Y. Ze, Z. Li, S. S. Sastry, J. Wu, K. Sreenath, S. Gupta, and X. B. Peng, "Learning smooth humanoid locomotion through lipschitz-constrained policies," *arxiv preprint arXiv:2410.11825*, 2024.

[26] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, Jul. 2022.

[27] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, feb 1991.

[28] C. Yang, K. Yuan, S. Heng, T. Komura, and Z. Li, "Learning natural locomotion behaviors for humanoid robots using human bias," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2610–2617, apr 2020.

[29] G. Romualdi, P. M. Viceconte, L. Moretti, I. Sorrentino, S. Dafarra, S. Traversaro, and D. Pucci, "Online dnn-driven nonlinear mpc for stylistic humanoid robot walking with step adjustment," *Submitted to IEEE-RAS Humanoids.*, 2024.

[30] G. Romualdi, S. Dafarra, Y. Hu, P. Ramadoss, F. J. A. Chavez, S. Traversaro, and D. Pucci, "A benchmarking of dcm-based architectures for position, velocity and torque-controlled humanoid robots," *International Journal of Humanoid Robotics*, vol. 17, no. 01, p. 1950034, Feb. 2020.

[31] D. Pucci, S. Traversaro, and F. Nori, "Momentum control of an underactuated flying humanoid robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 195–202, Jan. 2018.

[32] R. Olfati-Saber and A. Megretski, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, USA, 2001, aAI0803036.

[33] D. Roetenberg, H. Luinge, and P. J. Slycke, "Full 6dof human motion tracking using miniature inertial sensors," *Xsens Motion Technol. BV Tech. Rep.*, vol. 3, Jan. 2009.

[34] I. Loshchilov, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019.