2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Real-Time Model Predictive Control of Nonlinear Coupled Joints Using MPPI: Application to Humanoid Ankle Joints

Gunoo Park, Jaewan Bak, Yunsoo Seo, Euncheol Im, Hoseok Lee, Jongbok Lee, Nicolas Mansard, Jongwon Lee, and Yisoo Lee

Abstract— **Modern robotic systems increasingly employ nonlinear coupled joints, which present significant challenges in control. Unlike traditional serial chain configurations, where simplicity was the primary concern, parallel mechanisms such as those found in humanoid ankle joints add another layer of complexity. In this work, we propose an actuation controller for nonlinear coupled joints based on Model Predictive Path Integral (MPPI) control framework: a sampling-based model predictive control framework that incorporates nonlinearity and coupling effect simultaneously. Highly nonlinear Actuator-Joint mapping, expressed through lightweight neural network, enables intuitive controller design by exposing the actuator space control to the joint space command. Also, our method enables posing joint limit constraints, enabling safe operation on a real-robot platform. To experimentally validate our method, joint position control of a humanoid ankle joint with 2-DOF has been conducted, where accurate, real-time control and constraint-respecting behavior has been demonstrated.**

I. Introduction

It has been a recent trend to place actuators proximally to reduce link inertia, enabling robots to perform more dynamic tasks [1]. In particular, recent humanoid robots [2], [3], [4], [5], [6] have adopted parallel mechanisms to efficiently package actuators within constrained volumes and to reduce end-effector inertia, thereby enhancing overall agility. A common example is the ankle joint, where parallel structures enable lightweight leg designs and improved dynamic performance.

Despite their advantages, the practical use of parallel mechanisms remains limited because their structural complexity makes control challenging. The inherent nonlinear and coupled dynamics of such mechanisms hinder the ability to achieve consistent joint-level control, which is a critical requirement for both traditional high-level controllers, such as task-space control schemes [7], and recent reinforcement learning-based approaches [8], [9]. In particular, end-to-end reinforcement learning methods, which have gained increasing attention, face similar challenges: the complexity of parallel mechanisms is difficult to fully represent in simulation, making it necessary to train joint-level policies and then map the joint commands to actuator inputs. Consequently, both conventional control frameworks and simulation-based reinforcement learning approaches rely on accurate and consistent joint-level control, which remains difficult to realize in robots employing parallel mechanisms.

To control the coupled passive joints inherent in parallel mechanisms, an explicit mapping between the actuators and joints is required. This mapping is further complicated by the fact that the relationship between actuators and joints is typically nonlinear. Jacobian matrix-based *reactive control* approaches have been proposed in the literature to address this issue. In [2] and [10], the inverse kinematic relationship and the associated Jacobian were derived analytically, while the forward relationship was computed numerically using Newton's method. In [3], the velocity command of a hydraulic actuator at the ankle joint was obtained based on the Jacobian matrix. In [11], polynomial approximation was used for the forward relationship of coupled differential hip joints of the robot leg, and the corresponding Jacobian was subsequently derived by differentiation.

Traditional reactive control approaches often struggle to guarantee stable performance, as they do not explicitly account for system constraints. To ensure safe operation, however, actuator and joint limits must be systematically incorporated into the control framework. In this regard, *Model Predictive Control* (MPC) [12] has emerged as a promising solution, since it naturally accommodates state and input constraints within its optimization-based formulation. Furthermore, MPC can optimize and smooth discontinuous control references generated by high-level controllers, which helps to suppress overshoot and vibrations while improving overall stability and robustness.

However, it is not straightforward to incorporate nonlinear dynamics or nonlinear cost functions into the standard MPC formulation. Conventional MPC is typically solved through quadratic programming, which is inherently limited to handling linear dynamics and quadratic costs. When applied to nonlinear systems, linear MPC based on local linearization often leads to performance degradation. Nonlinear MPC (NMPC) can in principle address nonlinear dynamics and non-quadratic costs, but it requires differentiable formulations, posing challenges in constraint handling [13], or incurs significantly higher computational costs [14]. Therefore, both linear MPC and NMPC approaches are unsuitable for high-frequency control tasks, where fast online computation is crucial.

*This work was supported by the Korea Institute of Science and Technology (KIST) Institutional Programs under grant numbers 2E33591 and by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (RS-2024-00339632, RS-2025-25448259).

Gunoo Park and Nicolas Mansard are with the LAAS-CNRS, 7 Av. du Colonel Roche, 31400 Toulouse, France. Gunoo Park was with KIST, 02792, Seoul, South Korea, when this work was conducted.

Jaewan Bak, Euncheol Im, Hoseok Lee, Jongwon Lee and Yisoo Lee are with the KIST.

Jongbok Lee is with Virginia Tech, Blacksburg, VA 24061, USA

Yunsoo Seo is with University of Texas at Austin, Austin, Texas 78712, USA. Yunsoo Seo was with KIST when this work was conducted.

Corresponding author: Yisoo Lee, yisoo.lee@kist.re.kr

Recently, *sampling-based MPC* methods, such as the Model Predictive Path Integral (MPPI) framework [15], have gained increasing attention. By accommodating arbitrarily nonlinear system dynamics without requiring differentiability of the cost function, offering relatively simple implementation and enabling fast computation suitable for high-frequency control, these approaches are being actively applied across a wide range of robotic systems [16], [17], [18].

In this study, an MPPI-based actuation controller is proposed for nonlinear coupled joint. The main contributions are summarized as follows: (1) To the best of our knowledge, this work presents the first high-frequency real-time MPC-based controller for nonlinear coupled parallel joints. (2) The difficulty of explicitly formulating the nonlinear coupling between actuators and joints is addressed by training a neural network through deep learning to represent their relationship. (3) A novel MPC scheme is developed by integrating the neural network model into the MPPI framework, enabling effective control of nonlinear coupled joints. (4) The proposed approach is validated through both simulation and experimental studies on the ankle joint of a humanoid leg.

II. Review: Model Predictive Path Integral Control

MPPI [15], [19] is a sampling-based optimal control method. At each loop, it computes the next control input by minimizing a finite-horizon cost using importance-weighted evaluations of noisy rollouts.

In the following, we provide a brief overview to MPPI, starting with defining the system dynamics. The time-invariant discrete-time dynamic system is expressed as

$$\mathrm{d}x = f(x,t)\Delta t + G(x,t)\left(u(x,t) + \delta u\right)\Delta t, \tag{1}$$

The state x evolves according to the forward dynamics $f(x,t)$, and the control effectiveness matrix $G(x,t)$ maps the control signal to the state space. The control consists of a nominal input $u(x,t)$ and an injected Gaussian perturbation $\delta u \sim \mathcal{N}(0,\nu)$ that enables stochastic exploration.

The objective is to minimize a cumulative cost functional of the form

$$\tilde{S}(\mathbf{x}) = \phi(x_T) + \int_t^T l(x_t,t)\,dt, \tag{2}$$

where $\mathbf{x}$ is the state trajectory $x_t, ..., x_T$, $\phi(x_T)$ is the terminal cost and $l(x_t,t)$ denotes the running cost. In MPPI, the introduction of the likelihood ratio modifies the cost to go, resulting in an augmented form that naturally recovers the original quadratic control penalty from the optimal control formulation. Given these simplifications, the instantaneous cost reduces to

$$\begin{aligned}\tilde{l}(x,u,\delta u) = l(x,t) + \frac{(1-\nu^{-1})}{2}\delta u^T R \delta u \\ + u^T R \delta u + \tfrac{1}{2} u^T R u,\end{aligned} \tag{3}$$

where the additional terms arise from the stochastic control perturbations δu and the quadratic control effort penalty $u^T R u/2$. This formulation highlights that MPPI embeds the standard control cost structure within the sampling-based path integral framework.

Given the modified running cost, the cumulative cost-to-go of each sampled trajectory can be evaluated. The optimal control sequence is then updated by computing an importance-weighted over the sampled control perturbations:

$$u_k \leftarrow u_k + \frac{\sum_{n=1}^{N} \exp\!\left(-\frac{1}{\lambda}\left(\tilde{S}_n(\mathbf{x}) - \tilde{S}_{\min}\right)\right)\delta u_{k,n}}{\sum_{n=1}^{N} \exp\!\left(-\frac{1}{\lambda}\left(\tilde{S}_n(\mathbf{x}) - \tilde{S}_{\min}\right)\right)}, \tag{4}$$

where $\tilde{S}_n(\mathbf{x})$ denotes the cumulative cost of the n-th sampled trajectory, $\tilde{S}_{\min}$ is the minimum cost among all sampled rollouts introduced to prevent numerical overflow or underflow, and λ is the inverse temperature parameter that governs the selectiveness of weighting. Intuitively, trajectories with lower cumulative cost receive higher weights, which biases the control update toward control perturbations $\delta u_{k,n}$ associated with more favorable outcomes.

Through this sampling-based update process, MPPI inherits several favorable properties. Because the control update is obtained directly from weighted rollouts, the method does not rely on gradient information of the cost function, which allows it to handle non-differentiable objectives and discontinuous constraints in a straightforward manner. At the same time, since the dynamics are propagated through the rollouts without approximation, nonlinear system models or neural networks can be directly incorporated into the optimization. This makes MPPI particularly suitable for the nonlinear and coupled joint control problems considered in this paper. Moreover, due to the simplicity of the optimization process, the proposed method achieves real-time performance even with limited computational resources. In addition, the independence of sampled trajectories makes the method inherently parallelizable, enabling efficient implementation on modern hardware and supporting real-time scalability to complex robotic systems.

III. Proposed Method

A. Overview of Proposed Control Structure

The overall structure of the proposed MPPI-based actuation controller for coupled joints is illustrated in Fig.1. The controller receives a desired joint position command $\boldsymbol{q}^d$ and generates the corresponding actuator desired state, consisting of position $\boldsymbol{\psi}^d$ and velocity $\dot{\boldsymbol{\psi}}^d$, while also mapping the measured actuator state $(\boldsymbol{\psi}, \dot{\boldsymbol{\psi}})$ into the joint state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$. At the core of the actuation controller is the *Actuator–Joint mapping*, implemented using a neural network (NN), which is essential for converting between actuator and joint spaces.

To compute optimal control input trajectory $\{\dot{\boldsymbol{\psi}}^{d*}\}_n$, dynamics rollout is performed for all N samples to produce state trajectories $\{\boldsymbol{\psi}\}_{n,h}$ and input trajectories $\{\dot{\boldsymbol{\psi}}^d\}_{n,h}$. When evaluating those trajectories, NN based Actuator-Joint Mapping engages in the cost function, to calculate the cost-weighted sum of sampled trajectories. Finally, immediate control input $\dot{\boldsymbol{\psi}}^{d*}$ is integrated to compute $\dot{\boldsymbol{\psi}}^d$, and a

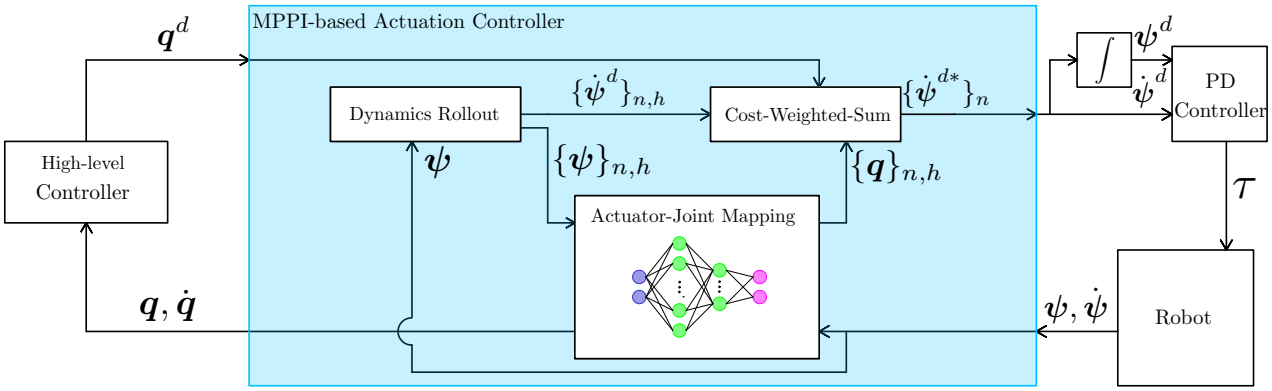


Fig. 1: Overview of the proposed MPPI-based actuation controller.

PD control law is applied to generate the actuator torque command. Details are described in the following subsections.

B. Mapping Between Actuator and Joint

In nonlinear coupled joint control, deriving the forward relationship between actuator and joint positions is generally challenging and often requires either simplifying assumptions for linearization or iterative numerical methods. In contrast, the inverse relationship is typically more straightforward to obtain analytically. For example, in [2], the inverse mapping $\psi = f^{-1}(q)$ is derived analytically, and the forward mapping $q = f(\psi)$ is computed numerically using the Newton–Raphson method. Furthermore, in most coupled mechanisms, only actuator angles are directly measurable, making the forward relationship essential for practical control.

To efficiently address the challenge of deriving the forward relationship for nonlinear coupled joints, we employ a Neural Network (NN) to directly learn the mapping between actuator and joint angles. Training data were collected in simulation by sweeping each actuator angle across its full range, ensuring dense coverage of the configuration space. This approach was chosen because, in most real robotic systems, joint angle sensors are not typically available. However, if both actuator and joint angle measurements are accessible on a real robot, the same dataset could alternatively be obtained through physical experiments.

For every sampled pair of actuator angles as input, the corresponding pair of joint angles as output was recorded, enabling the NN to effectively learn the full nonlinear coupling behavior. The network architecture consists of two hidden layers with 32 and 16 units, respectively, which provides sufficient nonlinearity to capture the complex mapping. While increasing the network size led to a slight improvement in prediction accuracy, the gain was marginal compared to the substantial increase in inference time. Since the proposed NN is integrated into the MPPI framework, maintaining high computational efficiency is critical for real-time system. Considering this trade-off, we selected this compact architecture as it provides sufficient accuracy while ensuring low computational cost.

Algorithm 1 MPPI-based actuation controller

Input: ψ_{init} : current actuator angle
Parameters: N : rollout, T : time horizon
while task not completed **do**
  $\psi_0 \leftarrow \psi_{init}$
  $\delta\dot{\psi}^d \leftarrow$ random noise generation$(N, T,$ mean, variance$)$
  for $n = 0, \ldots, N-1$ **do**
    for $h = 0, \ldots, T-1$ **do**
      $\psi_{n,h+1} \leftarrow$ state update$(\dot{\psi}_{n,h}, \delta\dot{\psi}^d_{n,h})$ $\Rightarrow$ Eq. (5)
      $q_{n,h+1} \leftarrow$ NN state mapping$(\psi_{n,h+1})$
      $\tilde{l}_{n,h+1} \leftarrow$ cost calculation$(q_{n,h+1})$ $\Rightarrow$ Eq. (3),(6)
      $\tilde{S}_{n,h+1} \leftarrow \tilde{S}_{n,h} + \tilde{l}_{n,h+1}$
    end for
  end for
  $\dot{\psi}^{d*}_h \leftarrow$ control input calculation$(\tilde{S}_n, \delta\dot{\psi}^d)$ $\Rightarrow$ Eq. (4)
end while

C. MPPI-Based Actuation Controller Design

To use MPPI as an actuation controller for a coupled joint, we define the state variable x of the system (1) as the actuator position $\psi = [\psi_1 \ \psi_2]^\top$. To obtain actuator space control input, we define the control input u to be the desired actuator velocity $\dot{\psi}^d = [\dot{\psi}^d_1 \ \dot{\psi}^d_2]^\top$, which can be used in tandem with the integrated control input $\int \dot{\psi} dt = \psi^d$ in a PD control scheme. At each time step, the actuator states are integrated with the update rule given by

$$\psi_{n,h+1} = \psi_{n,h} + \left(\dot{\psi}^d_{h-1} + \delta\dot{\psi}^d_h\right)\Delta t, \tag{5}$$

where ψ_h represents the actuator angle at time h, $\dot{\psi}^d_{h-1}$ denotes the desired angular velocity from the previous time step, and $\delta\dot{\psi}^d_h$ represents the perturbation. By adding the perturbation with a previously optimized control input, MPPI algorithm warm-starts itself, resulting in a faster convergence and more effective exploration.

After each trajectories are generated, they are evaluated through the cost function. It is worth mentioning that *the trajectories are evaluated in the joint space, not the actuator space*, to directly reason about the coupled states. The state-

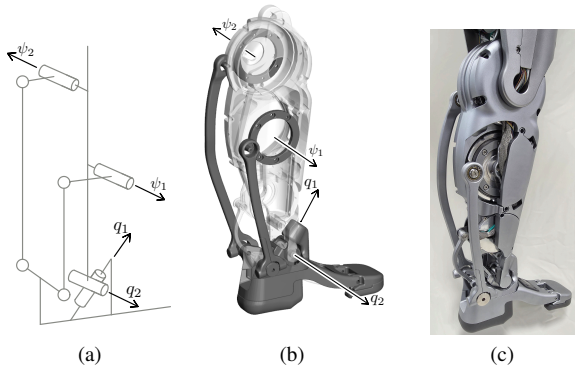


Fig. 2: Kinematic structure of the ankle joint. (a) Stick figure of the joints, (b) simulation model, and (c) real robot.

dependent cost function is designed to penalize both tracking error and constraint violations:

$$l(x,h) = 1000(\boldsymbol{q}^d - \boldsymbol{q}(\boldsymbol{\psi}))^\top(\boldsymbol{q}^d - \boldsymbol{q}(\boldsymbol{\psi})) + 10^6 I(\{\underline{\boldsymbol{q}} \le \boldsymbol{q}(\boldsymbol{\psi}) \le \overline{\boldsymbol{q}}\}), \quad (6)$$

where joint space angle $\boldsymbol{q}(\boldsymbol{\psi}) = [q_1\ q_2]$ is computed from the actuator angle $\boldsymbol{\psi}$ via the neural network, I is the indicator function, and $\underline{\boldsymbol{q}}$, $\overline{\boldsymbol{q}}$ is lower and upper bound of joint limits. This formulation leverages MPPI's flexibility to accommodate nonlinear cost structures, enabling the controller to effectively compensate for complex actuator-joint coupling effects. Once the optimal control input $\dot{\boldsymbol{\psi}}^{d*}$ is determined according to Eq. (4), it is integrated once to obtain desired actuator position $\boldsymbol{\psi}^d$ and the actuator torque is applied as $\boldsymbol{\tau} = K_p(\boldsymbol{\psi}^d - \boldsymbol{\psi}) + K_d(\dot{\boldsymbol{\psi}}^d - \dot{\boldsymbol{\psi}})$, ensuring stable execution and accurate tracking.

Algorithm 1 outlines the proposed control procedure. First, at each iteration, the controller initializes the current actuator angle $\boldsymbol{\psi}_0$ and samples N sequences of angular velocity perturbations $\delta\dot{\boldsymbol{\psi}}^d$ over the prediction horizon T. Next, dynamics rollout is performed using these perturbations as a control input. For each rollout, resulting actuator angles are mapped into the joint space through the neural network, enabling evaluation of the state-dependent cost in terms of the task-relevant joint variables. Lastly, after obtaining cumulative costs $\tilde{S}_h$ across all trajectories, the optimal control input $\dot{\boldsymbol{\psi}}_h^{d*}$ is obtained through the cost-weighted averaging.

IV. Experimental Verification

To validate the effectiveness of the proposed control method, both real-world experiments and physics-based simulations were conducted. The developed controller was implemented on the ankle joint of a humanoid robot. As shown in Fig.2(a), the ankle joint consists of two pitch motors connected via four-bar linkage mechanism, which simultaneously controls the ankle roll and pitch joints. This configuration leads to nonlinear coupling between the two DOFs, making the control problem challenging and representative of the focus of this study.

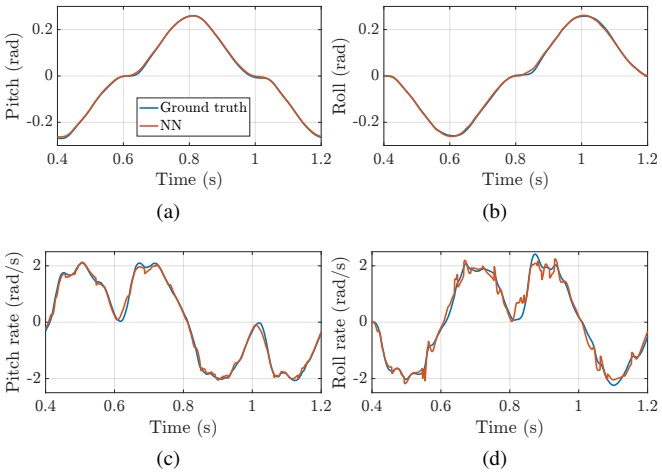


Fig. 3: Joint state estimation results using the NN compared with ground-truth data in simulation. (a), (b): pitch and roll joint position; (c), (d): pitch and roll joint velocity.

The controller computations were executed on a PC running Ubuntu 22.04 with a Preempt-RT real-time kernel patch to ensure low-latency performance and accurate time measurement. Communication between the PC and motors was established via an EtherCAT network at a frequency of 1 kHz. The motors were driven under a torque control scheme: the joint controller first computed the desired motor torques, which were then converted into the corresponding command currents by dividing by the motor torque constants.

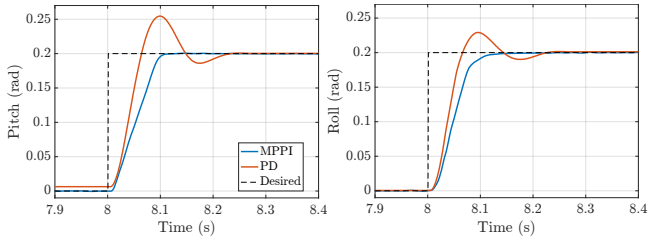
To analyze control performance aspects that are difficult to measure on the physical robot, we employed the MuJoCo [20] physics simulator. Within the simulation environment, a detailed model of the closed-loop four-bar linkage was implemented to closely replicate the real joint's kinematics and dynamics as shown in Fig 2(b).

A. Evaluation of Actuator-to-Joint Mapping

To evaluate the accuracy of the proposed Actuator-to-Joint mapping using the trained NN, we conducted simulations since the real robot does not provide direct measurements of joint angles. As the roll and pitch motions of the joints are kinematically coupled, both joints were commanded to move simultaneously. Reference trajectories were generated using cubic spline interpolation, and the joint angles estimated by the NN were compared with the ground-truth values obtained from the simulator.

The estimation accuracy of the joint angles is presented in Fig.3(a) and (b). The root mean squared estimation error was approximately 0.0036 rad, and the maximum error was about 0.0128 rad. Although small deviations were observed in some regions, the NN generally provided sufficiently accurate joint angle estimates.

To map the rate of change of actuator angles to joint angles, the Jacobian matrix $\mathbf{J} = \partial q/\partial\psi$ is used, as expressed by $\dot{q} = \mathbf{J}\dot{\psi}$. In our case, since the forward mapping $q = f(\psi)$ is represented by a trained NN, the input–output Jacobian of



(a) (b)

Fig. 4: Step input response of the proposed controller (blue) and PD controller (orange) without joint constraint. Black dashed line represents desired input.

an NN can be derived analytically using the chain rule [21]. Each layer of the network can be expressed as:

$$y = \mathrm{ReLU}(\mathbf{W}x + \mathbf{b}), \tag{7}$$

where x and y are the layer's input and output, and $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector. The derivative of the output with respect to the input is then:

$$\frac{\partial y}{\partial x} = \frac{\mathrm{dReLU}(z)}{\mathrm{d}z}\frac{\mathrm{d}z}{\mathrm{d}x}, \tag{8}$$

where $z = \mathbf{W}x + \mathbf{b}$. Since the derivative of ReLU is straightforward to compute and $\partial z/\partial x = \mathbf{W}$, the local Jacobian for each layer can be obtained easily. By recursively applying this process to all layers via the chain rule, the overall input–output Jacobian of the NN can be computed.

The joint velocities were computed using the Jacobian matrix $\mathbf{J}$, and the results are shown in Fig.3(c) and (d). The root mean squared error in joint velocity estimation was approximately 0.146 rad/s, and the maximum error was around 0.709 rad/s, indicating that the proposed approach yields reliable estimates for practical applications.

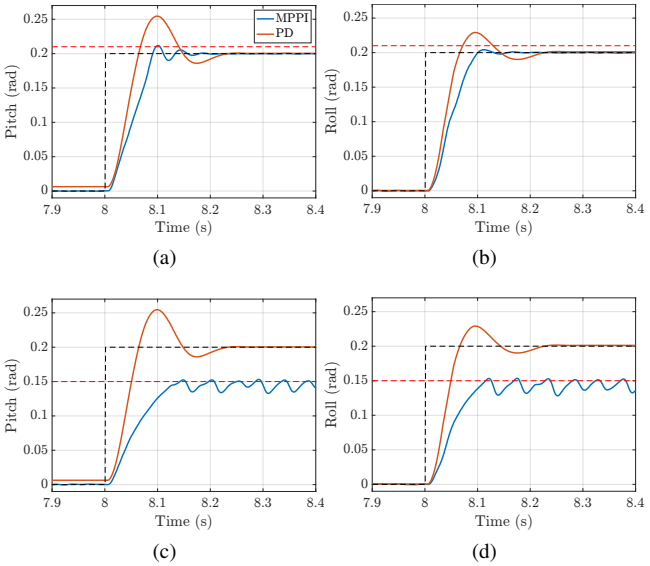
As discussed previously in Section III-B, the accuracy can be further improved by increasing the size of the NN, but at the cost of higher inference times. Considering the trade-off between accuracy and computational efficiency, the current NN architecture provides a reasonable balance suitable for real-time control, with each NN inference taking approximately 0.00186 ms on average.

B. Experimental Results

To verify the performance of the proposed controller, several tasks were performed on a real robot, with varying control inputs, joint limit constraints and parameters. For comparison, we designed a PD controller for the joint control where actuator torque is calculated based on the desired joint position and velocity using the Jacobian matrix transpose:

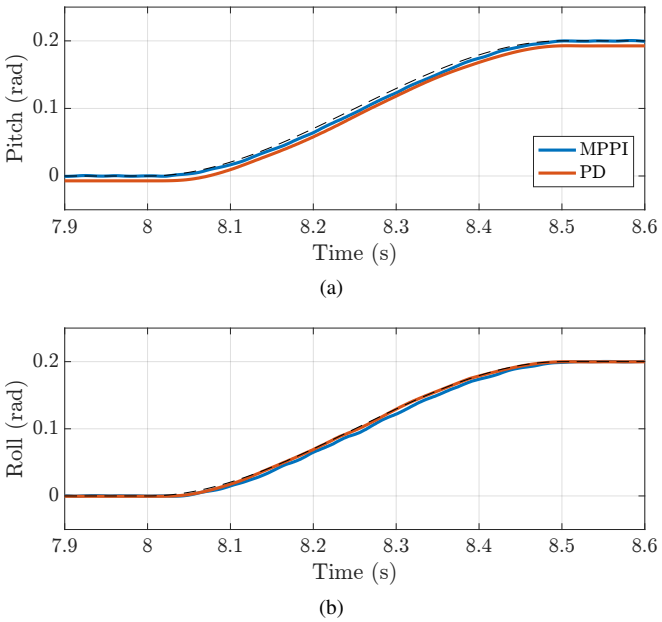
$$\boldsymbol{\tau} = \mathbf{J}^{\top}\left(\mathrm{K}_p(\boldsymbol{q}^d - \boldsymbol{q}) + \mathrm{K}_d(\dot{\boldsymbol{q}}^d - \dot{\boldsymbol{q}})\right), \tag{9}$$

where K_p and K_d are gain diagonal matrices. The gains of the PD controller were empirically tuned to minimize tracking errors during operation, ensuring stable and accurate control performance. Here, $\mathbf{J}$ is computed by the NN-based method (8).



(a) (b) (c) (d)

Fig. 5: Step input response of MPPI (blue) and PD (orange) controller for ankle joints, with constraints set (a), (b) above the desired position and (c), (d) below the desired position. Black dashed line represents for desired position and red dashed line represents for joint limit



(a) (b)

Fig. 6: Spline input response of MPPI controller (blue) and PD controller (orange). Black dashed line represents desired input.

1) Point-to-point control: When subject to discontinuous command, it is natural to think that controlling in an MPC manner would show better control performance in terms of overshoot or settling time, compared to a simpler reactive control. To observe this, we compared the step input response of the two controllers. 0.2 rad of step input was given in both pitch and roll joints. Rollout number $N = 200$ was used

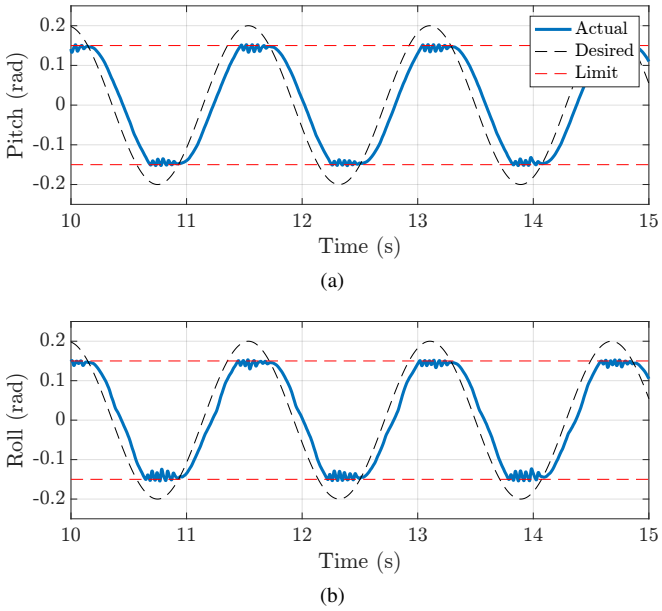


Fig. 7: Sine input response of MPPI controller for (a) pitch joint and (b) roll joint. Black dashed line represents for desired position and red dashed line represents for joint limit

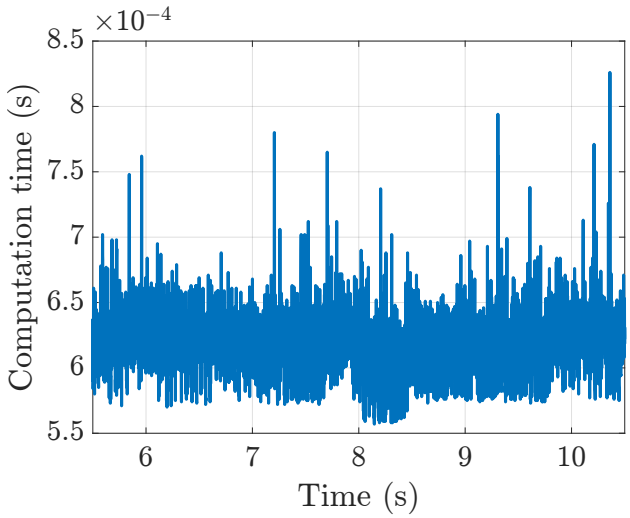


Fig. 8: Computation time of proposed MPPI controller

to ensure as many sample numbers as possible that satisfy the real-time requirement of 1 kHz. To ensure fast response, we set the parameters time interval $dt = 0.001$, predictive horizon $T = 3$, exploration variance $\nu = 9$, and temperature $\lambda = 0.1$. The lower the temperature, the more selective the weighted sum of rollouts will be, setting higher weights for low-cost trajectories.

As shown in Fig. 4, the proposed controller showed a shorter settling time (from 8 s to 8.1 s in the pitch joint) without much overshoot, while the PD controller settled slightly later (8.22 s), with an overshoot of 0.06 rad in the pitch joint. After settling, the PD controller has a steady-state error of 0.0012 rad in the roll joint, whereas the MPPI controller has an RMSE of 0.00029 rad in the roll joint.

To observe the behavior of the proposed controller when constraints are present, we set arbitrary joint constraints around the desired input. Fig.5(a) and (b) illustrate the result of the step input response of the PD controller and the proposed controller, with constraints set outside the commanded position. Both controllers converged to the command input without a large error. However, the PD controller showed a larger overshoot (approximately 0.05 rad in the pitch state), which inevitably violates the joint constraint, whereas the proposed method had a lower overshoot (approximately 0.01 in pitch state), respecting the joint limit constraint. Also, the PD controller settled at a slightly later time (approximately 8.22 s in both states), whereas the proposed method settled at a slightly earlier time (approximately 8.17 s in both states).

Fig.5(c), (d) shows the case when the constraint was set below the commanded position. In case of PD controller, it has failed to perform a safe control by exceeding the joint limit. On the other hand, the proposed controller satisfies the joint constraint, with vibrations present. Arguably, this vibration is coming from MPPI's repetitively re-establishing optimal trajectory over a very short predictive horizon. To avoid intense vibration, the temperature parameter had to be set as high as $\lambda = 1.0$.

2) Trajectory tracking: When subject to continuous input, the proposed method is also expected to perform as well as a simple reactive strategy. To observe this, we gave a sine and a cubic spline trajectory in both pitch and roll joints. Rollout number $N = 200$, time interval $dt = 0.001$, predictive horizon $T = 3$, and exploration variance $\nu = 9$ was used. In Fig. 6, the desired input smoothly changes from 8.0 s to 8.5 s, which both controllers tracks well with RMSE being 0.0012 for MPPI controller and 0.00029 for PD controller in the roll joint, respectively. In Fig. 7, it can be seen that the controller tracks the sine wave trajectory well while respecting the joint limit constraints, showing the effectiveness of the proposed method in the trajectory tracking task. In this case, to prevent excessive vibration, temperature $\lambda = 1.0$ was used.

3) Computation time: Fig.8 shows the computation time of the proposed controller, with a mean value of 0.617 ms. Since most of the computations are completed within 1 ms, these results verify that the proposed method can be executed successfully at a 1 kHz control frequency, thereby demonstrating the feasibility of real-time MPC. Although the maximum computation time occasionally exceeds 1ms, the overall stability of the computation confirms reliable high-frequency control. To further ensure this reliability, we set a timeout of 0.85ms; whenever the computation time exceeds this threshold, the MPPI algorithm immediately terminates the current rollout and proceeds to the next iteration. Under this setting, we observed stable performance throughout the experiments, confirming the feasibility of 1 kHz real-time control. Moreover, the computation time is expected to decrease further when GPU parallelization is employed, leveraging the inherently parallel structure of the MPPI algorithm.

V. Conclusion

In this work, we proposed a real-time MPC-based actuation controller for nonlinear coupled joints, built on the MPPI framework. We established a nonlinear mapping between actuators and joints through a NN and incorporated it into the MPPI cost function, allowing the controller to account for coupling effects. Owing to the flexible formulation of MPPI, joint limit constraints could also be imposed directly within the controller, ensuring safe operation of the robot. We experimentally validated the methodology by applying the controller to the humanoid ankle joints with a parallel mechanism, and the results demonstrated reliable real-time tracking of the desired inputs while strictly satisfying joint limit constraints.

Importantly, the proposed method shows that nonlinear joint control problems can be solved as real-time MPC using MPPI at a control rate of 1 kHz. By adjusting the parameter settings, the controller can flexibly address point-to-point regulation and trajectory tracking tasks while satisfying inequality constraints, highlighting significant advantages over conventional approaches. Although the current setup was sufficient to enable real-time control of 2-DOF joints, further extensions – such as increasing the number of sampled trajectories, lengthening the predictive horizon, or applying the method to higher-DOF systems – are readily achievable through computational parallelization, by employing GPU-parallelized MPPI computation [22].

References

[1] G. Ficht and S. Behnke, "Bipedal humanoid hardware design: A technology review," *Current Robotics Reports*, vol. 2, no. 2, pp. 201–210, 2021.

[2] C. Zhou and N. Tsagarakis, "On the comprehensive kinematics analysis of a humanoid parallel ankle mechanism," *Journal of Mechanisms and Robotics*, vol. 10, no. 5, 2018.

[3] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1028–1035.

[4] S. Sovukluk, J. Englsberger, and C. Ott, "Whole body control formulation for humanoid robots with closed/parallel kinematic chains: Kangaroo case study," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 390–10 396.

[5] S. Lohmeier, T. Buschmann, M. Schwienbacher, H. Ulbrich, and F. Pfeiffer, "Leg design for a humanoid walking robot," in *2006 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2006, pp. 536–541.

[6] Unitree, "Unitree G1 Humanoid Robot," 2025, [Online]. Available: https://www.unitree.com/g1 [Accessed: August 2025].

[7] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 22–27.

[8] 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*, vol. 44, no. 5, pp. 840–888, 2025.

[9] 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, p. eadi9579, 2024.

[10] H. Kaminaga, T. Ko, R. Masumura, M. Komagata, S. Sato, S. Yorita, and Y. Nakamura, "Mechanism and control of whole-body electro-hydrostatic actuator driven humanoid robot hydra," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 656–665.

[11] Y. Sim and J. Ramos, "Tello leg: The study of design principles and metrics for dynamic humanoid robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9318–9325, 2022.

[12] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[13] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[14] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[15] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[16] Y. Seo, D. Kim, J. Bak, Y. Oh, and Y. Lee, "Extremely fast computation of com trajectory generation for walking leveraging mppi algorithm," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–7.

[17] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 750–759.

[18] J. Alvarez-Padilla, J. Z. Zhang, S. Kwok, J. M. Dolan, and Z. Manchester, "Real-Time Whole-Body Control of Legged Robots with Model-Predictive Path Integral Control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. Atlanta, GA, USA: IEEE, May 2025, pp. 14 721–14 727. [Online]. Available: https://ieeexplore.ieee.org/document/11128271/

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[20] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[21] F. Cursi, V. Modugno, and P. Kormushev, "Model predictive control for a tendon-driven surgical robot with safety constraints in kinematics and dynamics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7653–7660.

[22] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "Mppi-generic: A cuda library for stochastic trajectory optimization," *arXiv preprint arXiv:2409.07563*, 2024.