Ising Machines for Model Predictive Path Integral-Based Optimal Control

Lorin Werthen-Brabants
imec - Ghent University
9052 Ghent, Belgium
`lorin.werthenbrabants@ugent.be`

Pieter Simoens
imec - Ghent University
9052 Ghent, Belgium
`pieter.simoens@ugent.be`

Abstract

We present a sampling-based Model Predictive Control (MPC) method that implements Model Predictive Path Integral (MPPI) as an *Ising machine*, suitable for novel forms of probabilistic computing. By expressing the control problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem, we map MPC onto an energy landscape suitable for Gibbs sampling from an Ising model. This formulation enables efficient exploration of (near-)optimal control trajectories. We demonstrate that the approach achieves accurate trajectory tracking compared to a reference MPPI implementation, highlighting the potential of Ising-based MPPI for real-time control in robotics and autonomous systems.

1 Introduction

Model Predictive Control (MPC) (Camacho and Bordons, 2007) is a widely used optimal control strategy, used to predict the next optimal action to be taken in robotics and automated driving, in which the optimal control input sequence $\{u_n\}$ is computed by solving a finite-horizon optimization problem at each time step. The optimization relies on a model of the system dynamics, $x_{n+1} = f(x_n, u_n)$, to predict future states and determine an optimal trajectory. Although MPC often yields good performance and robustness, the need to repeatedly solve an optimization problem in real time can make it computationally demanding, particularly for high-dimensional systems or those operating under strict energy or latency constraints.

Model Predictive Path Integral (MPPI) control (Williams et al., 2016) offers a probabilistic perspective on MPC by recasting the finite-horizon optimization as a stochastic sampling problem, rather than a direct deterministic optimizer, as is done when making use of Quadratic Programming (QP) optimizers such as SNOPT (Gill et al., 2005) or Clarabel (Goulart and Chen, 2024). Instead of solving a deterministic optimization problem directly, MPPI samples control trajectories from a distribution — typically Gaussian — and weights them according to the exponential of their negative cost, analogous to a Boltzmann distribution in statistical mechanics.

Recent advances in alternative computing paradigms provide new opportunities to accelerate computation. In particular, probabilistic bits (p-bits) (Çamsarı et al., 2017) have emerged as a promising technology for probabilistic computing. These p-bit based computers, or p-computers, are able to sample states from Ising machines and a Boltzmann distribution faster and more energy efficient than their CPU counterparts. FPGA simulations have been shown to provide approximately a $10\times$ improvement in sampling throughput compared to highly optimized TPUs and GPUs (Chowdhury et al., 2023). Unlike deterministic bits, p-bits stochastically fluctuate between binary states, allowing them to efficiently explore energy landscapes described by Boltzmann distributions. This makes p-bits well suited for implementing MPPI in binary or finite action spaces, where control selection can be naturally mapped to energy-based models such as the Ising model.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Frontiers in Probabilistic Inference: Sampling Meets Learning.

Prior work has mapped MPC with finite inputs directly to QUBO and solved it via annealing-style optimizers (Inoue and Yoshida, 2020). Our contribution differs by recasting *sampling-based* MPPI on a Boltzmann energy landscape and use Gibbs sampling to realize MPPI updates in binary spaces (Ising-MPPI), natively targeting probabilistic hardware (p-bits) (Si et al., 2024; Kim et al., 2024), rather than annealing based approaches such as with quantum annealing (Morita and Nishimori, 2008; Inoue and Yoshida, 2020).

2 Ising-based Model Predictive Path Integral Control

Model Predictive Path Integral (MPPI) control (Williams et al., 2016) can be understood as sampling trajectories according to a Boltzmann distribution over their costs. In the standard *continuous* setting, control sequences are perturbed by Gaussian noise vector $\epsilon \sim \mathcal{N}(\mu, \sigma)$, and each noisy trajectory $\mathbf{u}+\epsilon$ is assigned a weight, where $J(\mathbf{u})$ is the trajectory cost, $w(\epsilon) = {\exp\left(-\frac{1}{\lambda}J(\mathbf{u}+\epsilon)\right)}/{Z}$, where Z is the sum of all sampled $w(\epsilon)$, so that low-cost perturbations are exponentially more likely. This corresponds to sampling from a Boltzmann distribution, where the trajectory cost $J(\mathbf{u})$ plays the role of energy and the temperature parameter λ regulates exploration.

However, an Ising machine does not work with continuous-valued inputs, but rather spin valued $(-1/+1)$ or binary $(0/1)$ inputs. In the binary setting the same principle applies as in the continuous one, but the action sequence is represented as a binary vector $\mathbf{a} \in \{0,1\}^d$, transformed to continuous quantities by calculating $E\mathbf{a}$, where E is a binary expansion matrix (Inoue and Yoshida, 2020) as described in Equation A-9 . The Boltzmann distribution is then defined by a quadratic energy function, with $d = NLm$, N being the horizon length of the control problem, L the number of bits for expansion matrix E and m the number of control inputs:

$$p(\mathbf{a}) \propto \exp\left(-\tfrac{1}{\lambda}H(\mathbf{a})\right), \quad \text{where } H(\mathbf{a}) = \mathbf{a}^\top \mathbf{J}\mathbf{a} + \mathbf{h}^\top \mathbf{a}, \quad \mathbf{a} \in \{0,1\}^d. \tag{1}$$

For the proposed Ising-MPPI technique, the construction of $\mathbf{J}$ and $\mathbf{h}$ is achieved by transforming a linear (or linearized) MPC problem into a QUBO formulation, as outlined in Appendix A, specifically Equations A-11 and A-12.

Instead of Gaussian perturbations, The proposed Ising-MPPI makes use of Gibbs sampling (Koller and Friedman, 2009; Chowdhury et al., 2023) to explore the binary action space, producing samples $\{\mathbf{a}^{(s)}\}$ distributed according to the distribution in Equation 1. In probabilistic computers, this process is implemented efficiently and asynchronously (Chowdhury et al., 2023). For a binary vector $\mathbf{a} \in \{0,1\}^d$, the conditional probability for bit i is computed from the local field $p(a_i = 1|a_{\setminus i}) = \sigma(-1/\lambda(h_i + 2\sum_{j\neq i} J_{ij}a_j))$ with σ the logistic function $\sigma(z_i) = 1/1+e^{-z_i}$. For Ising-MPPI the empirical mean of these samples is computed over S sweeps, and the final action is chosen by rounding each component $\hat{\mathbf{a}} = \mathbf{1}(1/S \sum_{s=1}^{S} \mathbf{a}^{(s)} > 0.5)$, which can then be transformed to the actual action sequence $\hat{\mathbf{u}} = \mathbf{E}\hat{\mathbf{a}}$. This is done to anticipate hardware implementation where a long time constant RC circuit could calculate the average voltage (Çamsarı et al., 2019), and could then be rounded to the nearest binary value. The entire procedure is outlined in Algorithm 1.

At each control step — MPPI outer iteration — we re-linearize around the current state x_t and the latest nominal control $\bar{\mathbf{u}}$, then rebuild $\mathbf{A}, \mathbf{B}, \mathbf{c}$ and $(\mathbf{J}, \mathbf{h})$ accordingly. Just as MPPI converges to the optimal control by sampling and reweighting noisy perturbations according to a Boltzmann distribution over costs, the binary formulation converges to the optimal discrete control by Gibbs sampling from the corresponding Boltzmann distribution and rounding.

3 Results

The Ising-MPPI approach is tested on a non-linear kinematic bicycle model, described in Appendix C. Three different MPC setups are tested: 1) the **Ising-MPPI** setting, following the approach outlined in Algorithm 1, 2) a **Non-Ising Linear MPPI** setting, which makes use of the same $\mathbf{J}$ and $\mathbf{h}$ as the Ising-MPPI (Equations A-11 and A-12), but with the $\mathbf{E}$ terms removed from their formulation, as explained in Section A.5. The MPPI error is defined by the energy function H in Equations 1 and A-10, but applied on the action vector $\mathbf{u}$ rather than the binary version $\mathbf{a}$. In this setup, we can assess the effect of linearization outside of the binary-valued Ising context. Lastly, 3) a **reference**

Algorithm 1 Ising-based Model Predictive Path Integral (Ising-MPPI)

```
1: Input: Reference $\mathbf{x}^{\text{ref}}$, horizon $N$, temperature $\lambda$, Gibbs samples $S$, MPPI Iterations $M$
2: $\bar{\mathbf{u}} \leftarrow \mathbf{0}$                                          ▷ Initialize nominal control sequence
3: for $1$ to $M$ do
4:     $\mathbf{J}, \mathbf{h} \leftarrow$ Equations A-11 and A-12, using $\mathbf{x}^{\text{ref}}$ and horizon $N$      ▷ Construct $\mathbf{J}, \mathbf{h}$ matrices
5:     $\{\mathbf{a}^{(s)}\} \leftarrow \text{Gibbs}(\mathbf{J}, \mathbf{h}; S, \lambda)$          ▷ Generate $S$ binary samples via Gibbs sampling
6:     $\hat{\mathbf{a}} \leftarrow \mathbf{1}\left(1/S \sum_{s=1}^{S} \mathbf{a}^{(s)} > 0.5\right)$               ▷ Round to nearest binary value
7:     $\bar{\mathbf{u}} \leftarrow \bar{\mathbf{u}} + \mathbf{E}\hat{\mathbf{a}}$                                ▷ Map to continuous actions
8: end for
9: Apply first control input $\bar{\mathbf{u}}_0$
```

Table 1: Tracking error for the different setups. The tracking error is calculated as the mean squared error (MSE) between the predicted trajectory and the control trajectory. The standard deviation is also reported.

Type	**Tracking Error (MSE)**
Ising-MPPI	0.0271 ± 0.1085
Non-Ising Linear MPPI	0.0383 ± 0.0656
Reference	0.0015 ± 0.0019

implementation making use directly of the nonlinear bicycle model, as introduced by Williams et al. (2016). The default hyperparameters for the experiments are listed in Table A-1.

All experiments were conducted on a workstation running Ubuntu 22.04.3 LTS, equipped with dual Intel Xeon Silver 4210 CPUs (40 threads total, 2.20 GHz) and 256 GB of RAM. All computations were performed on the CPU without GPU acceleration.

3.1 Overall Performance

We test the mean error over many different trajectories generated according to Appendix B, with $M = 4$, $S = 200$ for Ising-MPPI, $M = 4$, $S = 1000$ for Non-Ising Linear MPPI and $M = 4$, $S = 1000$ for the reference MPPI. These values were found empirically to work well. We generate 50 different trajectories and run the MPPI algorithms in the three different setups with 10 different sampling seeds for a total of 500 trials.

The resulting tracking errors are shown in Table 1. As expected, the reference setup achieves the lowest error with very small variance, serving as an ideal baseline. Ising-MPPI obtains a lower mean error than Non-Ising Linear MPPI, but its variance is much larger, indicating less consistent performance. In contrast, Non-Ising Linear MPPI shows slightly higher average error but with reduced variability across trials.

3.2 Impact of Number of MPPI Samples on Trajectory Quality

Figure 1 shows the effect of the number of MPPI samples on trajectory quality for a single, specific trajectory (Trajectory 1 from Figure A-1), measured by the mean squared error (MSE). It can be seen that Ising-MPPI provides consistently lower MSE compared to the Non-Ising Linear MPPI formulation. The Non-Ising Linear MPPI converges at higher sample sizes ($S = 10^3$) in a predictable way, with more iterations M leading to lower errors with fewer samples per iteration S. It is important to highlight that the Non-Ising Linear MPPI cannot be solved on specialized Ising hardware and only serves to show the impact of linearization for MPPI. On the other hand the Ising-MPPI sampling approach converges to a low trajectory error faster. A surprising behaviour to be noted here is that the number of iterations for the discrete case does not simply reduce the error, but affects the error more unpredictably (e.g. $M = 3$ having a lower MSE than $M = 4$), unlike the Non-Ising Linear MPPI in which there is a clear inverse correlation between iterations M and the tracking error. This is likely due to over-smoothing, occurring when averaging many binary control samples leads to values near $1/2$, so rounding them flattens the control signal and reduces its responsiveness.

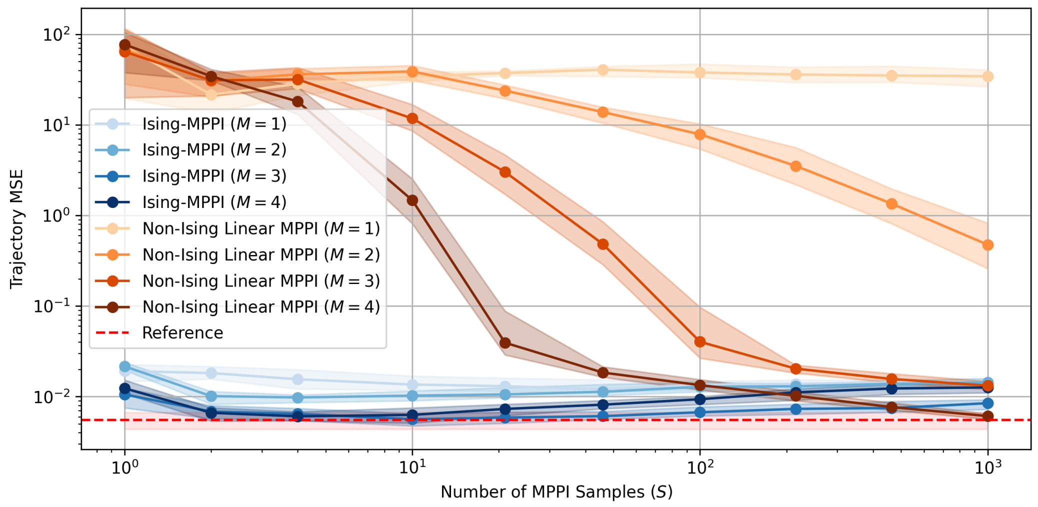

Figure 1: Comparison of Ising-MPPI vs Non-Ising Linear MPPI samples across different iteration limits. The x-axis represents the number of samples S (log scale), the y-axis shows the MSE (log scale). Different shades of blue and orange denote a differing amount of iterations M for the proposed finite-set Ising-MPPI and Non-Ising Linear MPPI respectively. The shaded areas represent the standard deviation. The red dashed line denotes the reference MPPI implementation.

4 Limitations

First, the QUBO mapping relies on linearization around a nominal trajectory; in strongly nonlinear regimes or when the system departs significantly from the nominal path, the local linear approximation can become inaccurate, and iterative relinearization only partially mitigates this. Second, discretizing control inputs introduces quantization error: achieving very fine control resolution requires many binary variables, which increases the sampling and encoding cost. Third, our empirical validation here is limited to a kinematic bicycle model and CPU-based sampling: the hardware advantages we discuss need hardware-in-the-loop experiments with p-bit or FPGA Ising machines for full validation. Finally, the approach assumes reasonable model knowledge (dynamics and Jacobians); significant model error will degrade linearization quality and performance.

5 Conclusion

In this work, we introduced a novel formulation of MPC with Ising Machine-based MPPI control. We show that MPC can be reinterpreted as sampling from a Boltzmann distribution over binary control sequences. This viewpoint bridges stochastic control and energy-based models, and allows utilization of specialized hardware for efficient trajectory sampling. The preliminary evaluation on a kinematic bicycle model shows similar results to the reference MPPI method, suggesting that binary sampling-based MPC can offer significant computational advantages when making use of p-computers.

From the previously discussed limitations in Section 4 we can derive several directions for future work. First, the current formulation assumes linearized dynamics around a nominal trajectory; extending the framework to handle more complex and less easily differentiable models and constraints would broaden its applicability, such as by encoding the model in a (deep) Boltzmann machine (Niazi et al., 2024; Salakhutdinov and Hinton, 2009). Second, hardware-in-the-loop experiments with FPGA- or actual p-bit-based Ising machines are necessary to validate the expected gains in sampling throughput and energy efficiency. Third, the discretization of control inputs introduces an approximation trade-off; exploring adaptive or multi-level quantization schemes could improve accuracy without sacrificing hardware compatibility.

References

E. F. Camacho and C. Bordons. *Introduction to Model Predictive Control*, pages 1–11. Springer London, London, 2007. ISBN 978-0-85729-398-5. doi: 10.1007/978-0-85729-398-5_1.

Shuvro Chowdhury, Andrea Grimaldi, Navid Anjum Aadit, Shaila Niazi, Masoud Mohseni, Shun Kanai, Hideo Ohno, Shunsuke Fukami, Luke Theogarajan, Giovanni Finocchio, et al. A full-stack view of probabilistic computing with p-bits: Devices, architectures, and algorithms. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 9(1):1–11, 2023. doi: 10.1109/JXCDC.2023.3256981.

Alessandro De Luca, Giuseppe Oriolo, and Claude Samson. Feedback control of a nonholonomic car-like robot. In *Robot motion planning and control*, pages 171–253. Springer, 2005. doi: 10.1007/BFb0036073.

Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005. doi: 10.1137/S0036144504446096.

Paul J Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*, 2024. doi: 10.48550/arXiv.2405.12762.

Daisuke Inoue and Hiroaki Yoshida. Model predictive control for finite input systems using the d-wave quantum annealer. *Scientific Reports*, 10(1):1591, 2020. doi: 10.1038/s41598-020-58081-9.

Jaehyun Kim, Joon-Kyu Han, Ho-Young Maeng, Janguk Han, Jeong Woo Jeon, Yoon Ho Jang, Kyung Seok Woo, Yang-Kyu Choi, and Cheol Seong Hwang. Fully cmos-based p-bits with a bistable resistor for probabilistic computing. *Advanced Functional Materials*, 34(22):2307935, 2024. doi: 10.1002/adfm.202307935.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. ISBN 978-0-262-01319-2.

Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12), 2008. doi: 10.1063/1.2995837.

Shaila Niazi, Shuvro Chowdhury, Navid Anjum Aadit, Masoud Mohseni, Yao Qin, and Kerem Yunus Çamsarı. Training deep boltzmann networks with sparse ising machines. *Nature Electronics*, 7(7): 610–619, 2024. doi: 10.1038/s41928-024-01182-4.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009. URL `https://proceedings.mlr.press/v5/salakhutdinov09a.html`.

Jia Si, Shuhan Yang, Yunuo Cen, Jiaer Chen, Yingna Huang, Zhaoyang Yao, Dong-Jun Kim, Kaiming Cai, Jerald Yoo, Xuanyao Fong, et al. Energy-efficient superparamagnetic ising machine and its application to traveling salesman problems. *Nature Communications*, 15(1):3457, 2024. doi: 10.1038/s41467-024-47818-z.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1433–1440. IEEE, 2016. doi: 10.1109/ICRA.2016.7487277.

Kerem Yunus Çamsarı, Rafatul Faria, Brian M Sutton, and Supriyo Datta. Stochastic p-bits for invertible logic. *Physical Review X*, 7(3):031014, 2017. doi: 10.1103/PhysRevX.7.031014.

Kerem Yunus Çamsarı, Brian M Sutton, and Supriyo Datta. P-bits for probabilistic spin logic. *Applied Physics Reviews*, 6(1), 2019. doi: 10.1063/1.5055860.

A MPC Formulation

A.1 Linear MPC Formulation

Consider a discrete-time system with linear dynamics (Camacho and Bordons, 2007):

$$x_{n+1} = A_n x_n + B_n u_n, \tag{A-1}$$

where $x_n \in \mathbb{R}^s$ is the system state at time step n, $u_n \in \mathbb{R}^m$ is the control input, and A_n, B_n are time-varying system matrices, with s and m the state and control dimensions.

In the Model Predictive Control (MPC) framework, the objective is to compute a sequence of future control actions $\{u_n\}_{n=0}^{N-1}$ over a finite horizon N that minimizes a cost function measuring the deviation of the system from a desired reference trajectory $\{x_n^{\text{ref}}\}$:

$$\min_{\{u_n\}} \sum_{n=0}^{N-1} \mathcal{L}(x_n, x_n^{\text{ref}}) + \mathcal{R}(u_n), \tag{A-2}$$

where $\mathcal{L}(x_n, x_n^{\text{ref}})$ quantifies state tracking error (e.g., quadratic error), and $\mathcal{R}(u_n)$ is a control regularization term to penalize excessive actuation. In a QUBO formulation, this can also be expressed as

$$J_n = x_n^\top Q x_n + u_n^\top R u_n, \tag{A-3}$$

where $Q \in \mathbb{R}^s \succeq 0$ and $R \in \mathbb{R}^m \succ 0$ are positive semidefinite and positive definite penalty matrices respectively.

At each time step, MPC solves this finite-horizon optimization using the current system state as the initial condition, applies the first control input u_0, and then repeats the process at the next time step. This receding-horizon strategy provides a balance between optimality and feedback robustness, enabling the controller to react dynamically to disturbances and model uncertainties.

A.2 Finite set MPC as a QUBO formulation

To reformulate Model Predictive Control (MPC) as a Quadratic Unconstrained Binary Optimization (QUBO) problem (Inoue and Yoshida, 2020), it is necessary to express the system dynamics over a finite prediction horizon in closed form. Consider a nonlinear system linearized around a sequence of nominal trajectories $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$. For each time step $n \in \{0, 1, \ldots, N-1\}$, the system dynamics are approximated by:

$$x_{n+1} = (I + A_n \Delta t) x_n + B_n u_n \Delta t + (f(\bar{x}_n, \bar{u}_n) - A_n \bar{x}_n - B_n \bar{u}_n)\Delta t, \tag{A-4}$$

where A_n and B_n are the Jacobians of the system dynamics with respect to the state and control inputs, respectively, evaluated at the nominal point $(\bar{x}_n, \bar{u}_n)$.

For sake of brevity, we will denote the time-discretized matrices $A_n \Delta t = A_n^\Delta$ and $B_n \Delta t = B_n^\Delta$. The block matrices $\mathbf{A}$ and $\mathbf{B}$, along with the constant vector $\mathbf{c}$, are constructed using the state transition matrices:

$$\Phi(i,j) = \begin{cases} \prod_{k=i}^{j-1}(I + A_k^\Delta), & \text{if } i < j, \\ I, & \text{if } i = j, \end{cases} \tag{A-5}$$

which describe the forward evolution of the linearized dynamics from time i to j.

Using these definitions:

$$\mathbf{A} = \begin{bmatrix} \Phi(0,1) \\ \Phi(0,2) \\ \vdots \\ \Phi(0,N) \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \Phi(1,2)B_1^{\Delta} & 0 & 0 & \cdots & 0 \\ \Phi(1,3)B_1^{\Delta} & \Phi(2,3)B_2^{\Delta} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi(1,N)B_1^{\Delta} & \Phi(2,N)B_2^{\Delta} & \Phi(3,N)B_3^{\Delta} & \cdots & 0 \end{bmatrix}, \tag{A-6}$$

$$c_i = \sum_{i=0}^{j-1} \Phi(i, j+1) \left[f(\bar{x}_i, \bar{u}_i) - A_i \bar{x}_i - B_i \bar{u}_i \right] \Delta t. \tag{A-7}$$

The resulting block matrices $\mathbf{A}$, $\mathbf{B}$, and vector $\mathbf{c}$ provide a compact, closed-form description of the system evolution over the prediction horizon:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\mathbf{A}x_0}_{\text{initial state contribution}} + \underbrace{\mathbf{B}\mathbf{u}}_{\text{control contribution}} + \underbrace{\mathbf{c}}_{\text{nonlinear offset}}, \tag{A-8}$$

where, with s the size of the state dimension, m the size of the action dimension and N the number of timesteps in the horizon: $\mathbf{A} \in \mathbb{R}^{Ns \times s}$ captures the effect of the initial state, $\mathbf{B} \in \mathbb{R}^{Ns \times Nm}$ maps the stacked control inputs to the state trajectory, $\mathbf{c} \in \mathbb{R}^{Ns}$ includes constant terms arising from deviations from the linearization point.

A.3 Binary Expansion Matrix

To discretize continuous controls into binary form, we use an expansion matrix $E \in \mathbb{R}^{m \times L}$, where m is the number of control inputs and L the number of binary variables per input. The matrix is constructed with scaled powers of two:

$$E_{j,i} = K \frac{2^{i-1}}{2^{L-1}} \quad \text{for } j \in \{1, \dots, m\}, i \in \{1, \dots, L\}, \tag{A-9}$$

with the sign of the most significant bit $E_{j,L} = -K$ to allow negative values. Each row of E is then repeated for all inputs to become a block matrix $\mathbf{E} \in \mathbb{R}^{Nm \times NL}$ where N is the horizon length and $\mathbf{E}\mathbf{a} = \mathbf{u}, \mathbf{a} \in \{0,1\}^{NLm}, \mathbf{u} \in \mathbb{R}^{Nm}$. The hyperparameters used for experiments are listed in Table A-1.

A.4 Interaction terms and bias

Substituting the closed-form expression of the state trajectory into the cost function, and expressing the control input vector $\delta\mathbf{u} = E\mathbf{a}$, where $\mathbf{a}$ is a binary vector encoding the control actions, we obtain a quadratic expression of the following form, analogous to the approach by Inoue and Yoshida (2020):

$$H(a) = \mathbf{a}^\top \mathbf{J} \mathbf{a} + \mathbf{h}^\top \mathbf{a}, \tag{A-10}$$

where, with x_0 the current state and $\bar{\mathbf{u}}$ a nominal action vector

$$\mathbf{J} = \mathbf{E}^\top \left(\mathbf{B}^\top \mathbf{Q} \mathbf{B} + \mathbf{R} \right) \mathbf{E}, \tag{A-11}$$

$$\mathbf{h}^\top = 2 \left(\mathbf{A}x_0 + \mathbf{B}\bar{\mathbf{u}} + \mathbf{c} - \mathbf{x}^{\text{ref}} \right)^\top \mathbf{QBE}. \tag{A-12}$$

Here, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{c}$ are derived from the closed-form representation discussed previously. $\mathbf{Q} \in \mathbb{R}^{Ns}$ and $\mathbf{R} \in \mathbb{R}^{Nm}$ are stacked penalty matrices Q and R respectively. $\mathbf{x}^{\text{ref}} \in \mathbb{R}^{Ns}$ is the reference trajectory that needs to be followed. The binary vector $\mathbf{a}$ parametrizes the control sequence via a linear mapping $\mathbf{u} = \mathbf{E}\mathbf{a}$, where $\mathbf{E}$ is the binary expansion block matrix. Note that in Equation A-12 the effect of the nominal action sequence $\mathbf{B}\bar{\mathbf{u}}$ is present. As a result, the optimal vectors $\delta\mathbf{u} = \mathbf{E}\mathbf{a}$ are only deviations from the nominal action sequence $\bar{\mathbf{u}}$, in order to make it compatible with the MPPI formulation.

The matrix $\mathbf{J}$ defines the quadratic coefficients of the QUBO problem, while the vector $\mathbf{h}$ defines the linear coefficients. Together, these specify the optimization objective in standard QUBO form:

$$\min_{\mathbf{a}\in\{0,1\}^d} H(\mathbf{a}) \tag{A-13}$$

where d is the total number of binary decision variables, in our case $d = NLm$.

A.4.1 Symmetrization

For a standard Ising formulation, $\mathbf{J}$ needs to be symmetric and have a zero-valued diagonal. In order to ensure this, we can apply the following method. Given a quadratic coefficient matrix $\mathbf{J}$ and linear vector $\mathbf{h}$, we apply the following preprocessing routine:

$$\begin{aligned} \mathbf{J} &\leftarrow \tfrac{1}{2}(\mathbf{J} + \mathbf{J}^\top) && \text{(symmetrization)}, && \text{(A-14)} \\ \mathbf{h} &\leftarrow \mathbf{h} + \operatorname{diag}(\mathbf{J}) && \text{(absorb diagonal into biases)}, && \text{(A-15)} \\ \operatorname{diag}(\mathbf{J}) &\leftarrow 0 && \text{(remove self-interactions)}. && \text{(A-16)} \end{aligned}$$

A.5 Non-Ising Linear MPPI formulation

In order to analyze the impact of discretization and direct sampling of the Ising Machine, a **Non-Ising Linear MPPI** formulation is made, where the regular MPPI algorithm is applied with the Ising energy objective with the binary expansion block matrix $\mathbf{E}$ removed:

$$\mathbf{J} = \mathbf{B}^\top \mathbf{Q}\mathbf{B} + \mathbf{R}, \tag{A-17}$$

$$\mathbf{h}^\top = 2\left(\mathbf{A}x_0 + \mathbf{B}\bar{\mathbf{u}} + \mathbf{c} - \mathbf{x}^{\text{ref}}\right)^\top \mathbf{Q}\mathbf{B}, \tag{A-18}$$

with Equation A-10 ($H(a) = \mathbf{a}^\top \mathbf{J}\mathbf{a} + \mathbf{h}^\top \mathbf{a}$) used as the trajectory cost for MPPI.

B Random Trajectory Generation

To test the proposed binary MPPI controller, we require a variety of reference trajectories that reflect different driving conditions. For this purpose, we generate randomized spline-based trajectories using the procedure below.

The generator begins from a fixed start position at the origin, $(0, 0)$, and iteratively samples a sequence of control points. At each step, a random heading angle and a random step distance (between 4 and 5 units) are drawn, and the new control point is placed relative to the previous one. The first heading angle is uniformly sampled in $[0, 2\pi)$, while subsequent headings are restricted to lie within a $\pm\pi/2$ cone around the previous segment direction. This ensures smooth trajectories without sharp reversals. The procedure is repeated until a total of eight control points are obtained.

Examples of the proposed MPPI technique applied to different, randomly generated trajectories can be seen in Figure A-1.

C Bicycle Model Benchmark Environment

To evaluate the proposed binary MPPI framework, we consider a simple kinematic bicycle model with steering dynamics. The system state and control inputs are defined as

$$x = [p_x,\, p_y,\, \theta,\, v,\, \delta]^\top, \quad u = [a,\, \omega_\delta]^\top, \tag{A-19}$$

where (p_x, p_y) denote the vehicle position, θ is the heading angle, v is the forward velocity, and δ is the steering angle. The control vector consists of the longitudinal acceleration a and the steering rate ω_δ. The continuous-time dynamics (De Luca et al., 2005) are given by

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v}{L}\tan\delta \\ a \\ \omega_\delta \end{bmatrix}. \tag{A-20}$$

For our tests, we set $L = 1$. The system dynamics $f(x, u)$ are linearized around the current state x and control input u. This is done by computing the Jacobian matrices of the dynamics with respect to the state and input:

$$\mathbf{f}_x = \frac{\partial f}{\partial x}, \qquad \mathbf{f}_u = \frac{\partial f}{\partial u}, \tag{A-21}$$

where $\mathbf{f}_x$ represents the sensitivity of the system dynamics to the current state, and $\mathbf{f}_u$ represents the sensitivity to the control input.

Evaluating these Jacobians at a specific state x and control u gives the linearized system matrices:

$$A = \mathbf{f}_x\big|_{x,u}, \qquad B = \mathbf{f}_u\big|_{x,u}. \tag{A-22}$$

The matrices A and B provide a local linear approximation of the nonlinear dynamics around the chosen operating point, which is essential for control design and analysis techniques that rely on linear models.

D Hyperparameters

Table A-1: Hyperparameters used in experiments.

Category	Symbol / Name	Value
MPC/Dynamics	Horizon length	$N = 8$
	Time step	$\Delta t = 0.1$
	Vehicle model	Kinematic bicycle
Cost function	State cost Q	$\mathrm{diag}([1000, 1000, 1, 0, 0])$
	Control cost R	$\mathrm{diag}([1, 1])$
Binary encoding	Bits per control input	$L = 5$
	Magnitude of speed control	$K = 15$
	Magnitude of steering control	$K = 2.2$
Sampling (Gibbs / Ising)	Temperature	$\lambda = 0.1$
	Outer iterations	$M = 4$
	Gibbs sweeps per sample	$S = 1000$

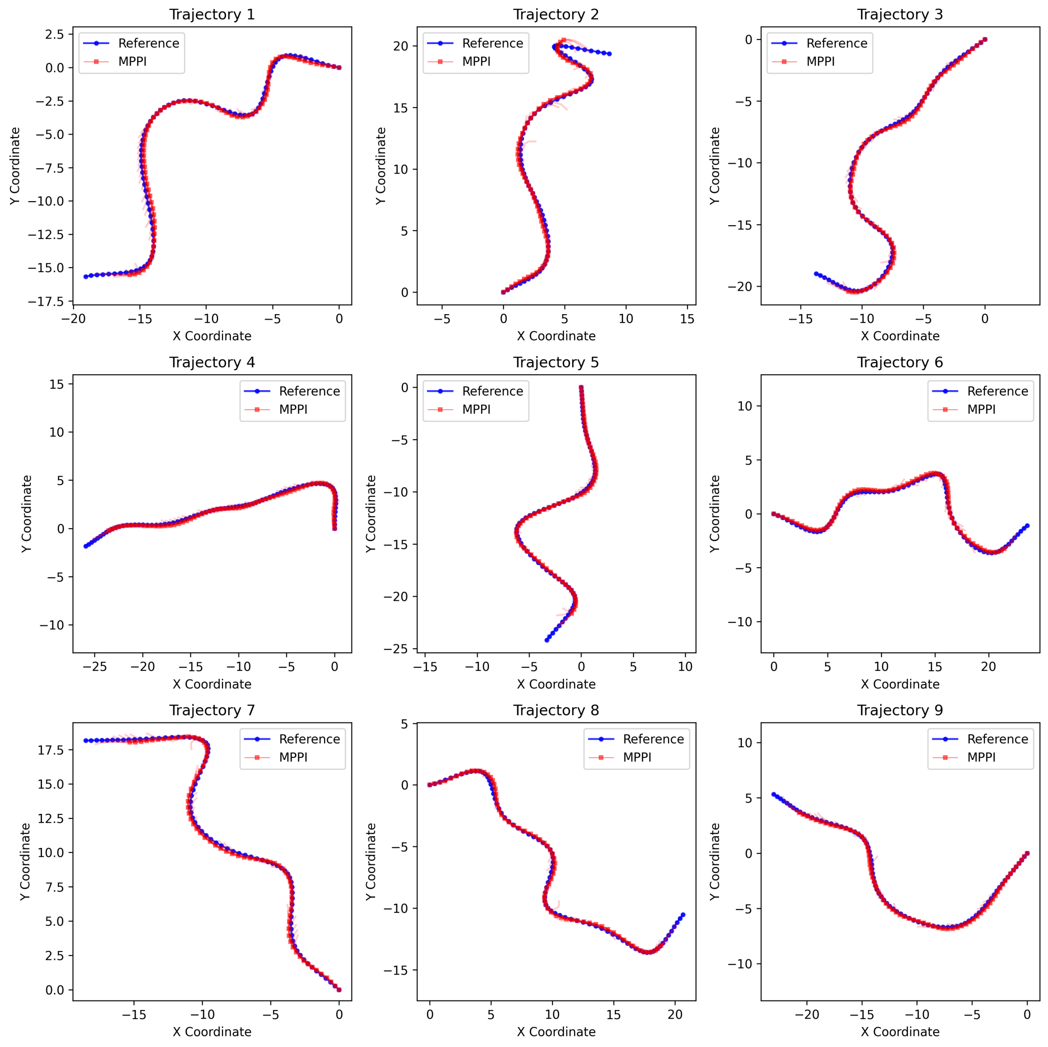

Figure A-1: Examples of trajectories and their Ising-MPPI solutions. Note that the $N = 8$ last points in the reference trajectory are not solved due to no additional reference points being available. Each trajectory starts at $(0, 0)$. The light red streaks visible are the predicted trajectories at each point.