machines

MDPI

Article

Soft-Constrained MPC Optimized by DBO: Anti-Disturbance Performance Study of Wheeled Bipedal Robots

Weihua Chen [1], Yehao Feng [2], Tie Zhang [2,*] and Canlin Peng [2]

[1] School of Mechanical Engineering, Guangzhou City University of Technology, Guangzhou 510800, China; adahuachen@foxmail.com
[2] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510640, China; fyh_dlut@163.com (Y.F.); 13327381577@163.com (C.P.)
* Correspondence: merobot@scut.edu.cn

Abstract

In disturbance scenarios, wheeled bipedal robots (WBRs) require effective control algorithms to restore balance. To address the trade-off between computational burden and control precision, and to enhance anti-disturbance capability, this paper proposes a soft-constrained Model Predictive Control (MPC) algorithm with optimized horizon parameters tailored to the hardware of the WBR. A cost function is designed, and the Dung Beetle Optimizer (DBO) is employed to optimize the MPC's prediction and control horizons. An experimental platform is built, and impact and load disturbance experiments are conducted. The experimental results show that, under impact disturbances, the pitch angle and displacement overshoot with optimized MPC are reduced by 58.57% and 42.20%, respectively, compared to unoptimized LQR. Under load disturbances, the pitch angle and displacement overshoot are reduced by 17.09% and 15.53%, respectively, with both disturbances converging to the equilibrium position.

Keywords: wheeled bipedal robot; anti-disturbance control; model predictive control; Dung Beetle Optimize

check for updates

Academic Editor: Giuseppe Carbone

Received: 31 August 2025
Revised: 28 September 2025
Accepted: 2 October 2025
Published: 4 October 2025

Citation: Chen, W.; Feng, Y.; Zhang, T.; Peng, C. Soft-Constrained MPC Optimized by DBO: Anti-Disturbance Performance Study of Wheeled Bipedal Robots. *Machines* **2025**, *13*, 916. https://doi.org/10.3390/machines13100916

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the field of mobile robotics, wheeled bipedal robots (WBRs) combine the advantages of humanoid and wheeled robots. They use leg structures to isolate torso vibrations caused by terrain irregularities, while employing wheels for balance and dynamic motion, making them well-suited for human working environments [1]. However, in typical working environments, WBRs are frequently subject to various types of disturbances, leading to issues such as slow control response, insufficient vibration suppression, overshoot, or oscillations, which negatively affect their performance [2]. Therefore, it is essential to design an anti-disturbance balance control algorithm that ensures control precision under limited computational resources, further enhancing the anti-disturbance performance of WBRs.

For WBRs, balance capability and anti-disturbance performance are fundamental for executing other actions. Traditional balance control methods can achieve basic balance control for WBRs, such as Proportional–Integral–Derivative (PID), Linear Quadratic Regulator (LQR), and Zero-Moment Point (ZMP) methods. These methods typically simplify the WBR to a Linear Inverted Pendulum (LIP) by ignoring leg dynamics to achieve balance control [3]. Appropriate control algorithms can also be obtained through model reduction

methods. A reduced-order model based on the Cuckoo Search Algorithm can reduce the computational burden of balance control [4]. In nonlinear domains that cannot be simplified, balance control can be achieved using the State-Dependent Riccati Equation (SDRE) control algorithm [5]. For WBRs with complex structures and state-space equations that are difficult to solve, Reinforcement Learning (RL) and Adaptive Dynamic Programming (ADP) can be employed to provide adaptive optimal control solutions based on learning [6]. For WBRs with known state-space equations, LQR can effectively achieve balance control with highly efficient performance [7]. However, the gain coefficients of the aforementioned control algorithms are mostly fixed, resulting in limited real-time responsiveness when the robot's external environment changes.

When the system model is known and the main control device has sufficient computational power, Model Predictive Control (MPC) can be effectively used for state tracking of the model. Thanks to the "online" feature of MPC, the algorithm can update the planned trajectory within the feedback loop at a certain frequency, providing the robot with enhanced robustness against disturbances [8]. Additionally, MPC can incorporate various inequality constraints, enhancing the system's insensitivity to external disturbances [9].

The original MPC method can effectively achieve anti-disturbance balance control for WBRs. A control algorithm is designed based on the robot's state-space model and formulated as a quadratic programming problem, yielding optimal outputs for given prediction and control horizons. In certain scenarios, this method achieves minimal control efforts superior to those of the LQR algorithm [10]. Shahida Khatoon et al. [11] applied MPC with Linear Quadratic Gaussian Controller (LQG) to a wheeled inverted pendulum system and conducted impact tests, with simulation results showing that MPC exhibits strong robustness. Yu Jianqiao et al. [12] applied MPC to the posture control of a WBR, ensuring that the robot tracks the posture trajectory with minimal error and maintains balance. Niloufar Minouchehr et al. [13] treated a two-wheeled inverted pendulum as an underactuated nonlinear system and designed an MPC algorithm to achieve anti-disturbance control. Marco Kanneworff et al. [14] developed an Intrinsically Stable MPC (IS-MPC) algorithm, achieving stable control of an arm-equipped wheeled inverted pendulum robot under explicit constraints. Cao Haixin et al. [15] designed a constrained MPC algorithm to achieve balance control for a WBR on sloped terrains and enhanced the robustness of the control algorithm by developing an Extended State Observer (ESO). These methods applied MPC to WBRs and achieved balance control; however, they did not constrain system states and control inputs based on the robot's hardware performance. This oversight could lead to inaccuracies in the MPC's computed results, rendering the robot uncontrollable due to hardware power limitations. In particular, when the robot is subjected to disturbances, the driving wheel motors often need to output large torques within a short time to maintain balance. However, if the motors operate beyond their rated torque for extended periods, issues such as control delays, overheating, and performance degradation may occur. Unlike hard constraints, which may render the optimization problem infeasible under large disturbances, soft constraints introduce a penalty mechanism that tolerates minor violations. This design ensures the continuous feasibility of the controller while balancing constraint satisfaction and control performance, thereby enhancing robustness and practicality in real-world applications. By introducing soft constraints into the MPC framework, the range of control inputs can be effectively limited, ensuring that the control signals remain within the rated torque and below the peak torque. This not only protects the hardware from overload but also enhances the feasibility and anti-disturbance performance of the control algorithm. To enhance the effectiveness of MPC in practical applications, researchers have proposed various methods to optimize control performance. Daniel C. Fernández et al. [16] conducted simulation experiments with multiple prediction horizons, achieving the control

objectives of reducing control errors and ensuring computational speed. Li Xingjia et al. [17] employed the Transient Search Optimization (TSO) algorithm to optimize the parameters of the MPC objective function, reducing the overshoot and steady-state error of the controlled robot. A. K. Kashyap et al. [18] combined the Ant Colony Optimization (ACO) algorithm with MPC to optimize the robot's position-solving problem in obstacle scenarios. Chen Zhenbin et al. [19] applied the Proximal Policy Optimization (PPO) learning method to adaptively adjust the prediction range within the MPC framework, achieving stable trajectory tracking. Jin Mengtao et al. [20] used the Chaos Particle Swarm Optimization (CPSO) algorithm to optimize the solving capability of MPC, effectively improving trajectory tracking performance. These methods optimized MPC parameters for various target scenarios applied to robots, improving the tracking performance of target state variables. However, the optimized parameters were relatively limited, and the considered scenarios did not comprehensively account for disturbances. For WBRs with high real-time response requirements, the sizes of the MPC's prediction and control horizons affect both the computation of control outputs and the main control device's computation time [21]. Among common heuristic optimization algorithms, the Dung Beetle Optimizer (DBO) algorithm proposed by Xue Jiankai et al. [22] can achieve multi-objective optimization and solutions. Yang Pei et al. [23] applied DBO to solve the optimization model for lightweight objectives, obtaining a global optimal solution with fewer iterations. Li Yanhui et al. [24] applied DBO to optimize wind models, obtaining an optimal set of model parameters. Due to its powerful solution space exploration capability, DBO outperforms classic optimization algorithms such as PSO [25].

Therefore, this paper proposes a soft-constrained MPC algorithm tailored to the hardware specifications of WBRs and optimizes the combination of prediction and control horizons for different disturbance scenarios, further enhancing the robot's stability under disturbances. The main contributions of this paper are as follows:

(1) To address the problem of anti-disturbance control for WBRs, a soft-constrained MPC algorithm considering hardware power limitations is proposed to optimize the feasible domain of control outputs.
(2) The domain parameters of the MPC algorithm are optimized using the heuristic DBO algorithm, reducing the computation time of the main control device while ensuring the robot's anti-disturbance performance.
(3) An SRobo110-II experimental platform and disturbance equipment are constructed to validate the proposed method through comparisons with the LQR balance control algorithm and others.

Besides this section, the structure of this paper is as follows: Section 2 establishes the dynamic model of the WBR through Newtonian mechanics analysis. Section 3 designs the MPC algorithm, sets reasonable control constraints based on the robot's hardware, and optimizes the sizes of the MPC's prediction and control horizons using the heuristic DBO algorithm. Section 4 constructs the experimental platform for the WBR and conducts anti-disturbance experiments with multiple parameter sets, analyzing the anti-disturbance performance under different horizon sizes and control algorithms. Section 5 provides a conclusion of the research presented in this paper.

2. Dynamics Analysis of WBRs

2.1. Dynamic Modeling

The dynamic model determines the fundamental control precision of the control algorithms applied to the robot [26]. The overall motion of a WBR can be decoupled into wheeled motion and legged motion. The wheeled motion includes body balance swinging, forward and backward parallel movements, and turning motions, while the

legged motion primarily involves height adjustment movements. In this paper, the primary control objective for anti-disturbance balance control is the main drive wheels, so the focus is mainly on wheeled motion.

The following assumptions are made when modeling the robot under wheeled motion conditions:

1. The body mass is equivalently concentrated at the center of mass of the entire robot.
2. The mass of the leg links is uniformly distributed.
3. The drive wheels experience rolling friction with the ground, and slippage is neglected.

Since the WBR has multiple joint degrees of freedom, it can be equivalently simplified to a wheeled inverted pendulum model with variable pendulum length under the wheeled motion scenario [27], as shown in Figure 1.

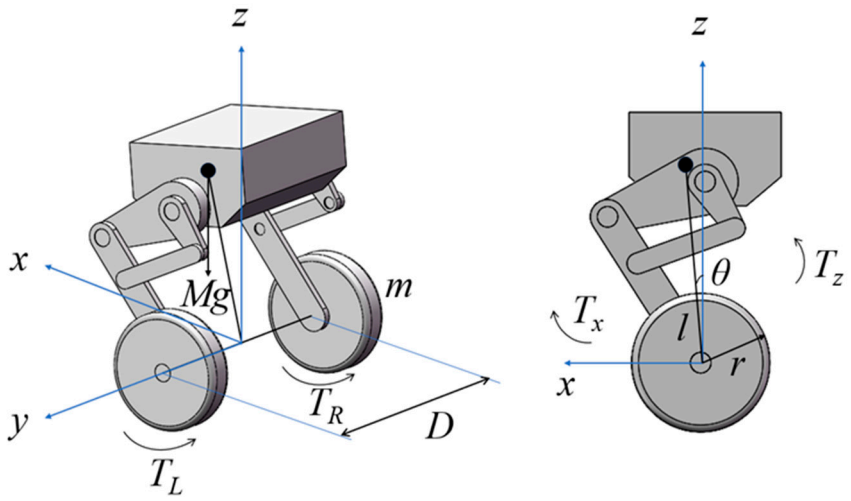


Figure 1. Wheeled Inverted Pendulum Model and its Projection on the x-z Plane.

The Newton-Euler method is a dynamic modeling approach used to derive closed-form dynamic equations [28], which is suitable for the wheeled inverted pendulum model studied in this paper. Since the primary state variable for balance control, the tilt angle of the center of mass, is normal to the y-axis, the force analysis of the robot is conducted in the x-z plane to obtain the dynamic equations for the robot body and the main drive wheels.

$$(J_y + Ml^2)\frac{d^2\theta}{dt^2} = T_z - T_x - (T_L + T_R) \tag{1}$$

$$(M + 2m + \frac{2I_{wheel}}{r^2})\frac{d^2x}{dt^2} = \frac{(T_L + T_R)}{r} + Ml(\frac{d\theta}{dt})^2 sin\theta \tag{2}$$

where J_y is the moment of inertia of the robot body about the y-axis, I_{wheel} is the moment of inertia of the drive wheels about their axes, T_L and T_R are the driving torques of the left and right wheels, respectively, T_z and T_x are the additional force couples generated by the resultant forces of the drive wheels along the z-axis and x-axis, respectively, when translated to the center of mass, M and m are the masses of the robot body and the drive wheels, respectively, l is the length of the pendulum, r is the radius of the drive wheels, θ is the tilt angle of the center of mass.

When the robot is in a balanced posture, the tilt angle of the center of mass is small, allowing the following linearization process:

$$sin\theta = \theta \quad cos\theta = 1 \quad (\frac{d\theta}{dt})^2 = 0 \tag{3}$$

By substituting Equation (3), which neglects higher-order nonlinear terms, into Equations (1) and (2) and performing a linear expansion at the equilibrium point, the dynamic equations for the x-z plane are obtained:

$$\begin{cases} (J_y + Ml^2)\frac{d^2\theta}{dt^2} = Mgl\theta - Ml\frac{d^2x}{dt^2} - (T_L + T_R) \\ r(M + 2m + \frac{2I_{wheel}}{r^2})\frac{d^2x}{dt^2} = (T_L + T_R) - Mlr\frac{d^2\theta}{dt^2} \end{cases} \tag{4}$$

When the control objective is anti-disturbance balance, four state variables are selected from the wheeled inverted pendulum model: displacement, velocity, pitch angle, and pitch angular velocity, forming the system state vector $x = \begin{bmatrix} x\ v\ \theta\ \dot{\theta} \end{bmatrix}^T$. At the same time, the driving torques of the left and right wheels are selected as the system inputs $u = \begin{bmatrix} u_0\ u_1 \end{bmatrix}^T$, and the system state-space model is constructed:

$$\dot{x} = Ax + Bu \tag{5}$$

By substituting Equation (4) into Equation (5), the matrix expression of the state-space model is obtained:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & A_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & A_{43} & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ B_{21} & B_{22} \\ 0 & 0 \\ B_{41} & B_{42} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \tag{6}$$

In Equation (6), the elements A_{23}, A_{43}, and others can be expressed as follows:

$$\begin{cases} A_{23} = \frac{-M^2l^2gr}{r(M+2m+\frac{2I_{wheel}}{r^2})(J_y+Ml^2)-M^2l^2r} \\ A_{43} = \frac{r(M+2m+\frac{2I_{wheel}}{r^2})Mgl}{r(M+2m+\frac{2I_{wheel}}{r^2})(J_y+Ml^2)-M^2l^2r} \\ B_{21} = B_{22} = \frac{Ml^2+Mlr+J_y}{r(M+2m+\frac{2I_{wheel}}{r^2})(J_y+Ml^2)-M^2l^2r} \\ B_{41} = B_{42} = \frac{-r(M+2m+\frac{2I_{wheel}}{r^2})-Ml}{r(M+2m+\frac{2I_{wheel}}{r^2})(J_y+Ml^2)-M^2l^2r} \end{cases} \tag{7}$$

2.2. Robot's Balanced Posture

Among the four state variables of the wheeled inverted pendulum model, the pitch angle is affected by the robot's posture. In most application scenarios, it is desirable for the robot to maintain a horizontal body posture in the balanced state, facilitating the coordinate calculation of extensions mounted on the body [29]. Therefore, to determine the robot's balanced posture, the following constraint is imposed on the projection of the entire system's center of mass in the *x*-*z* plane:

$$x_c = \frac{\sum m_i x_{ci}}{\sum m_i} = 0 \tag{8}$$

where x_c represents the *x*-axis coordinate of the robot's overall center of mass, x_{ci} denotes the *x*-axis coordinates of the centers of mass for individual parts such as the upper leg and lower leg links, and m_i represents the masses of the robot's components. The robot maintains a balanced state when the projection of its center of mass in the x-z plane coincides with the center of the drive wheels. The robot's posture at this time is shown in Figure 2.

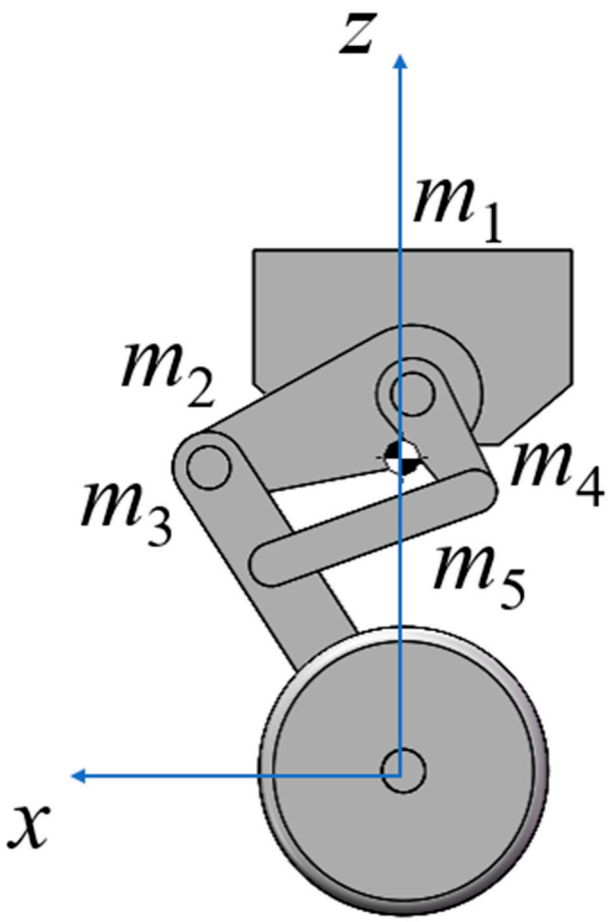


Figure 2. Robot's Balanced Posture.

When the robot's height changes, the overall posture is constrained by Equation (8) to ensure the accuracy of the target state variables in the balanced state.

3. MPC Algorithm Design for WBRs

As discussed in Section 2, under wheeled motion, the WBR can be equivalently simplified to a wheeled inverted pendulum system. The wheeled inverted pendulum system is highly unstable. Without a control algorithm, the system will gradually diverge over time, making it difficult to restore balance. Therefore, a proper control algorithm is required to ensure the system's controllability and allow it to converge over time, maintaining a balanced posture. MPC is widely used in the industrial field, and in the past decade, extensive research has been conducted on MPC's technical selection, variable control, and performance estimation [30]. Therefore, this paper adopts MPC as the balance control algorithm for the WBR and optimizes the horizon parameters of the MPC with anti-disturbance as the control objective. The control algorithm proposed in this paper, along with the control framework applied to the robot, is shown in Figure 3.

In Section 3.1, the state-space equation is discretized, and a soft-constrained MPCler is formulated. Subsequently, Section 3.2 analyzes the stability of the wheeled inverted pendulum system under the MPC framework. In Sections 3.3 and 3.4, the horizon parameters of MPC are embedded into the dung beetle optimization algorithm, and a cost function is devised to iteratively refine these parameters.

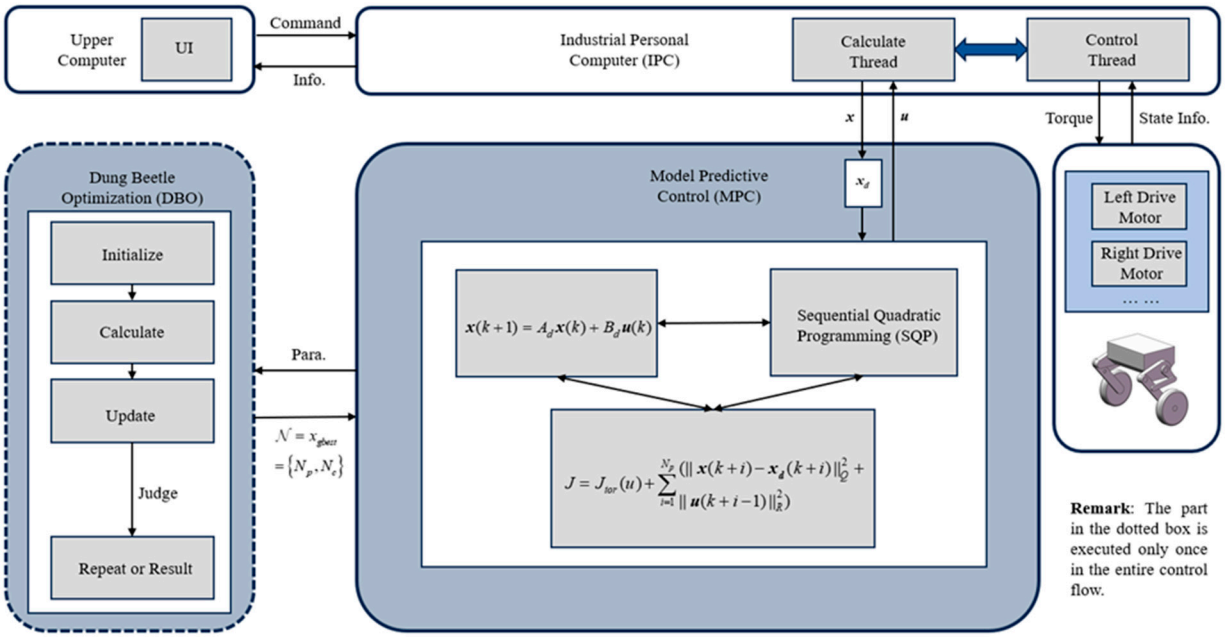


Figure 3. Control Framework of the Robot.

3.1. Soft-Constrained MPC Algorithm

In practical applications, the system state variables are obtained through discrete sampling during the control cycles. By discretizing Equation (5), the discrete state-space equation used for MPC calculations is obtained:

$$x(k+1) = A_d x(k) + B_d u(k) \tag{9}$$

The discretization method is as follows:

$$\begin{cases} A_d = e^{AT_s} \\ B_d = \left(\int_0^{T_s} e^{A\tau} d\tau \right) B \end{cases} \tag{10}$$

where x(k) and u(k) represent the state vector and control input at time step k, x(k + 1) represents the state vector at time step k + 1, T_s is the discretization period, A_d and B_d are the discretized system matrix and input matrix, respectively.

The objective of MPC is to solve an optimization problem at each sampling instant, based on the given prediction horizon N_p and control horizon N_c, generating a sequence of future control inputs $[u(k), u(k+1), u(k+2), \ldots, u(k+N_c-1)]$ such that the system's behavior within the prediction horizon meets certain performance criteria. At time step k, which serves as the initial sampling instant of the system, a time series $[k, k+1, k+2, \ldots, k+N_p-1]$ is selected, yielding the system state expression with a step length of N_p:

$$\left\{ \begin{array}{l} x(k+1) = A_d x(k) + B_d u(k) \\ x(k+2) = A_d x(k+1) + B_d u(k+1) \\ \ldots\ldots\ \ldots\ldots \\ x(k+N_p) = A_d x(k+N_p-1) + B_d u(k+N_p-1) \end{array} \right\} \tag{11}$$

By rearranging Equation (11) into matrix form, it can be rewritten as:

$$X(k) = Mx(k) + CU(k) \tag{12}$$

where $M \in \mathbb{R}^{N_p \cdot [\dim(\mathrm{x(k)})]^2}$ and $C \in \mathbb{R}^{[N_p \cdot \dim(\mathrm{x(k)})][N_c \cdot \dim(\mathrm{u(k)})]}$ are recursive matrices composed of A_d and B_d, respectively; $\mathrm{X} \in \mathbb{R}^{N_p \cdot \dim(\mathrm{x(k)})}$ is the stacked vector of future predicted states, and $\mathrm{U} \in \mathbb{R}^{N_c \cdot \dim(\mathrm{u(k)})}$ is the stacked vector of future control inputs, expressed as:

$$\left.\begin{array}{l} X = \left[x(k+1), x(k+2), \ldots, x\left(k+N_p\right)\right]^T \\ U = \left[u(k), u(k+1), \ldots, u\left(k+N_p-1\right)\right]^T \end{array}\right\} \tag{13}$$

The following cost function is defined for MPC:

$$J = \sum_{i=1}^{N_p} \left(\| \mathrm{x}(k+i) - \mathrm{x_d}(k+i) \|_Q^2 + \| \mathrm{u}(k+i-1) \|_R^2 \right) \tag{14}$$

where $Q \in \mathbb{R}^{[dim(x(k))]^2}$ and $R \in \mathbb{R}^{[dim(u(k))]^2}$ are the weight matrices for the state error and control input, respectively. x_d is the desired state vector, and for anti-disturbance balance control, $x_d = \left[0\ 0\ 0\ 0\right]^T$ holds at any time step in the time series $\left[k+1, k+N_p-1\right]$.

In the control framework of WBRs, the system's control frequency is typically in the microsecond range to meet real-time response requirements. When the robot is subjected to disturbances, the hub motors need to respond quickly and produce large control torques. However, if the motors operate beyond their rated torque for an extended period, issues such as control lag, overheating, and reduced control precision may arise. Therefore, soft constraints need to be designed to limit the solution domain obtained through MPC, ensuring that the control inputs do not exceed the rated torque for extended periods and always remain below the peak torque.

The designed soft constraint is expressed in the form of a penalty function. Let α_J be the constraint penalty weight and n be the nonlinear acceleration factor, both of which control the magnitude of the penalty function. The value of α_J needs to be adjusted according to the motor characteristics. From the motor characteristic curve, it is known that the heating power P_h of the motor is proportional to the square of the current:

$$P_h = I^2 R \tag{15}$$

where I is the motor current, and R is the resistance of the motor windings. Let the motor's temperature rise be denoted as ΔT_r:

$$\Delta T_r = k_r \cdot P_h \tag{16}$$

where k_r is the temperature rise coefficient of the motor, obtained from the motor characteristic curve. Let $\Delta\tau$ be the torque increment corresponding to the temperature rise ΔT_r. Since the current is proportional to the torque increment, the temperature rise-torque coefficient is defined as:

$$k^* = \frac{\Delta T_r}{\Delta \tau^2} = \left(\frac{I}{\Delta\tau}\right)^2 R \cdot k_r \tag{17}$$

Based on the motor's insulation class, the allowable temperature rise range ΔT_{al} can be selected, yielding the constraint penalty weight:

$$\alpha_J = \frac{k^* \cdot \left({\tau_m}^2 - {\tau_n}^2\right)}{\Delta T_{al}} \tag{18}$$

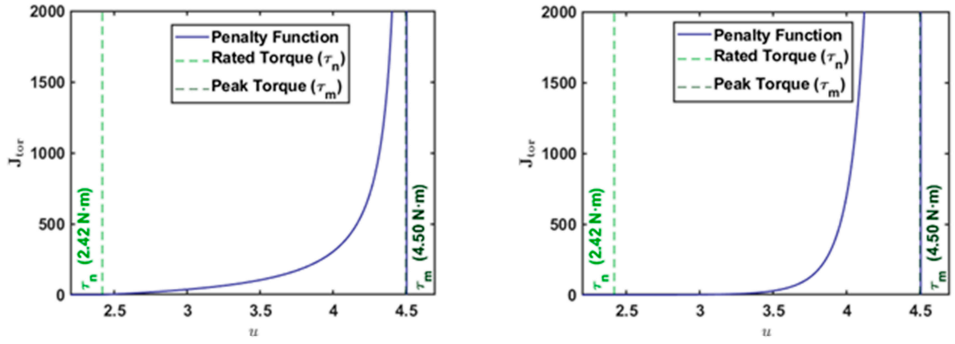
where τ_n and τ_m represent the rated torque and peak torque of the hub motor, respectively.

As for the nonlinear acceleration factor n, its magnitude determines the degree of nonlinearity of the penalty function. For the hub motor of the WBR, it is necessary to provide short-term torque that exceeds the rated torque. To allow for short-term overload,

the ideal range for n should be between $[1, 3]$. Based on the above definitions, the following nonlinear penalty function is designed:

$$J_{tor}(u) = \alpha_J \cdot \left(\frac{1}{\tau_m - u} - \frac{1}{\tau_m - \tau_n} \right)^n \tag{19}$$

The reference curve of the function in Equation (19) is shown in Figure 4.



(**a**) n=1 (**b**) n=3

Figure 4. Example of the Nonlinear Soft Constraint Penalty Function.

By incorporating the designed penalty function from Equation (19) into Equation (14), the final cost function of the MPC is obtained:

$$\begin{aligned} J &= \sum_{i=1}^{N_p} \left\{ (\| x(k+i) - x_d(k+i) \|_Q^2 + \| u(k+i-1) \|_R^2) + \alpha_J \cdot (\tfrac{1}{\tau_m - u} - \tfrac{1}{\tau_m - \tau_n})^n \right\} \\ s.t.\ \ & \mathrm{x}(k+1) = A_d x(k) + B_d u(k) \end{aligned} \tag{20}$$

Using the Sequential Quadratic Programming (SQP) algorithm, Equation (20) is solved iteratively by constructing and solving a series of quadratic programming subproblems to gradually approach the optimal solution, yielding the optimal system control input for the corresponding state vector.

3.2. Stability Analysis

System stability is one of the core issues in control design, that is, whether the system can remain controllable or gradually return to the equilibrium point after being subjected to initial perturbations or external disturbances. To verify the stabilizing effect of the proposed soft-constrained MPC controller on the wheeled biped robot and its equivalent wheeled inverted pendulum system, this study conducts an eigenvalue-based stability analysis for the systems with and without the incorporation of the soft-constrained MPC controller.

In the stability analysis, an eigenvalue discrimination method based on state-space theory is adopted. This method requires only the eigenvalue calculation of the system state matrix A, which enables a rapid determination of stability while avoiding complex computations. Let the eigenvalues of the continuous-time system state matrix A in Equation (5) be denoted as λ_i. The following criteria apply:

1. If the real parts of all eigenvalues satisfy $Re(\lambda_i) < 0$, the system is asymptotically stable.
2. If the real parts of all eigenvalues satisfy $Re(\lambda_i) \leq 0$, and for those with $Re(\lambda_i) = 0$, the corresponding Jacobian matrix has no repeated roots, the system is marginally stable.
3. If there exists any eigenvalue such that $Re(\lambda_i) > 0$, the system is unstable.

First, in the absence of the soft-constrained MPC controller, the main parameters of the robot, such as body mass and moment of inertia, are substituted into matrix A, yielding the following numerical expression:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3.3586 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 26.0544 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{21}$$

The eigenvalue calculation gives:

$$\lambda_{1,2,3,4,5,6} = [0, 0, 5.1044, -5.1044, 0, 0] \tag{22}$$

Since $Re(\lambda_3) > 0$, the system is determined to be unstable. This indicates that, without control, the wheeled inverted pendulum system cannot maintain stability, and thus an appropriate controller must be introduced to ensure state convergence.

After incorporating the MPC controller, the closed-loop system matrix becomes $A_{cl} = A - D$, where D is the equivalent feedback matrix given by:

$$D = \begin{bmatrix} 1 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 2 & -0.3 & 0 & 0 \\ 0 & 0 & -2 & 40 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Thus, the closed-loop state matrix is expressed as:

$$A_{cl} = \begin{bmatrix} -1 & 1.2 & 0 & 0 & 0 & 0 \\ 0 & -6 & -3.8586 & 0 & 0 & 0 \\ 0 & 0 & -2 & 1.3 & 0 & 0 \\ 0 & 0 & 28.0544 & -40 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1.1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{24}$$

The eigenvalues of A_{cl} are calculated as:

$$\lambda_{1,2,3,4,5,6} = [-1.00, -6.00, -1.06, -40.94, -1.00, -1.00] \tag{25}$$

The results show that all eigenvalues have negative real parts, indicating that the system achieves asymptotic stability under the soft-constrained MPC controller. Furthermore, the controllability matrix of the closed-loop system is full rank, confirming that the system remains controllable. Therefore, this analysis not only validates the effectiveness of the controller in stabilizing the wheeled inverted pendulum system but also highlights the necessity of the soft-constrained MPC in ensuring both stability and controllability.

3.3. DBO Algorithm Iteration Method

In the process of solving MPC, in addition to setting appropriate constraints, the sizes of the prediction horizon N_p and control horizon N_c are equally critical to computational complexity and control performance. The optimization problem in MPC essentially in-

volves large-scale matrix computations and the iterative solution of quadratic programming problems. Proper selection of prediction and control intervals can balance computational cost and control precision. When designing the MPC algorithm for wheeled biped robots, it is necessary to determine the combination of prediction and control horizons $\mathcal{N} = \{N_p, N_c\}$ based on the given weight matrices Q and R. This ensures that real-time performance and anti-disturbance capability are achieved while avoiding excessive computational burden.

The Dung Beetle Optimizer (DBO) is a heuristic multi-population optimization algorithm proposed by Xue Jiankai et al. [20]. This algorithm avoids local optimal solutions through various iterative strategies and is suitable for solving multi-objective optimization problems such as $\mathcal{N} = \{N_p, N_c\}$ in MPC. In the iterative computation process, DBO seeks the optimal solution to multi-objective optimization problems by simulating the behavior of dung beetles, generating four subpopulations: ball-rolling, breeding, foraging, and thief. Beetles in different subpopulations update their positions from the initial location according to predefined strategies. During each iteration, they calculate the cost function at their current positions based on the optimization objective, and the final position is taken as the solution to the optimization problem. The position update strategies for the four types of dung beetles are as follows:

(1) **Ball-Rolling Dung Beetle:** Performs the most basic iterative optimization, simulating the process of beetles navigating using sunlight. Random numbers $r_o = rand(0,1)$ and obstacle decision number $r_{op} \in (0,1)$ are defined to determine whether the ball-rolling dung beetle encounters an obstacle during movement. The position update equation for the obstacle-free mode is as follows:

$$\begin{cases} x_i(t+1) = x_i(t) + \alpha_D \cdot k_t \cdot x_i(t-1) + b \cdot \Delta x \\ \Delta x = x_i(t) - x_{worst}(t-1) \end{cases}, r_o < r_{op} \tag{26}$$

where t is the current iteration count, and $x_i(t)$ represents the position of the *i-th* dung beetle at the *t-th* iteration. α_D is the natural coefficient, determined to be either 1 or -1 based on probabilistic distribution, representing whether the beetle deviates from its original direction. $k_t \in (0, 0.2]$ denotes the regular deflection coefficient, and $b \in (0, 0.2)$ denotes the sunlight deflection coefficient. Δx is the light intensity variation, determined by the current position $x_i(t)$ and the globally worst position $x_{worst}(t-1)$ from the previous iteration.

When the ball-rolling dung beetle encounters an obstacle, it updates its direction through a dance. The position update equation for the obstacle mode is as follows:

$$x_i(t+1) = x_i(t) + tan(\theta_t)\left|x_i(t) - x_i(t-1)\right|, r_o \geq r_{op} \tag{27}$$

where $\theta_t = rand(0, \pi)$ determines the degree of deviation between the new direction and the original direction.

(2) **Breeding Dung Beetle:** Uses a boundary selection strategy to simulate the dung beetle's choice of the brooding ball region. Based on the current local optimal solution, it expands both inward and outward to generate two new solutions, thereby searching for a better solution near the local optimum. The position update equation is as follows:

$$\begin{cases} B_i(t+1) = x_{lbest}(t) + b_1 \cdot (B_i(t) - Lb^*) \\ \qquad\qquad + b_2 \cdot (B_i(t) - Ub^*) \\ Lb^* = max(x_{lbest}(t) \cdot (1 - R_e), Lb) \\ Ub^* = min(x_{lbest}(t) \cdot (1 + R_e), Ub) \end{cases} \tag{28}$$

where $B_i(t+1)$ is the position of the i-th brooding ball at the t-th iteration. b_1 and b_2 represent two independent random vectors of size $1 \times \dim(x_i(t))$. $R_e = (1-t)/T_{max}$ determines the extent of the brooding ball region expansion, where T_{max} is the maximum number of iterations. Lb^* and Ub^* are the lower and upper bounds of the brooding ball region, ensuring that the region dynamically adjusts with the number of iterations. Lb and Ub represent the lower and upper bounds of the optimization problem, respectively. $x_{lbest}(t)$ is the local optimal position at the i-th iteration.

(3) **Foraging Dung Beetle:** Similarly to the breeding dung beetle, it simulates the selection of a foraging area by young dung beetles. Based on the current global optimal solution, it generates two new solutions to search for a better solution near the global optimum. The position update equation is as follows:

$$\begin{aligned} x_i(t+1) &= x_i(t) + C_1 \cdot (x_i(t) - Lb^b) \\ &+ C_2 \cdot (x_i(t) - Ub^b) \\ Lb^b &= max(x_{gbest}(t) \cdot (1 - R_e), Lb) \\ Ub^b &= min(x_{gbest}(t) \cdot (1 + R_e), Ub) \end{aligned} \tag{29}$$

where Lb^b and Ub^b represent the lower and upper bounds of the foraging area, respectively. $C_1 \sim N(0,1)$ is a normally distributed random number. $C_2 \sim U(0,1)$ is a random vector following a uniform distribution. $x_{gbest}(t)$ is the global optimal position at the *t-th* iteration. The definitions of the remaining parameters are the same as those for the breeding dung beetle.

(4) **Thief Dung Beetle:** It searches for a potentially better solution by mimicking the features of the current global optimal solution and introducing random perturbations. The position update equation is as follows:

$$x_i(t+1) = x_{gbest}(t) + S \cdot g \cdot (|x_i(t) - x_{lbest}(t)| + |x_i(t) - x_{gbest}(t)|) \tag{30}$$

where S is a constant that determines the magnitude of the random perturbation. g represents a random vector of size $1 \times dim(x_i(t))$ following a normal distribution. The definitions of the remaining parameters are the same as those for the breeding dung beetle and the foraging dung beetle.

3.4. MPC Optimization Using the DBO Algorithm

In the optimization process of DBO, the design of the cost function J_D affects the characteristics of the final result. For anti-disturbance scenarios, we aim for the robot to quickly return to a balanced position after being disturbed, minimizing the amplitude of body oscillation and overall displacement, while ensuring the required torque and computation time. This paper is based on the classical Integral of Time-weighted Absolute Error (ITAE) cost function, which fully accounts for the accumulated absolute values of state variables over a certain time period [31]. J_D is defined as:

$$J_D = \alpha \int_{t_0}^{t_0+T} (t \cdot |\theta|) dt + \gamma \int_{t_0}^{t_0+T} (t \cdot |x|) dt + \lambda \int_{t_0}^{t_0+T} (t \cdot |u|) dt + \delta \int_{t_0}^{t_0+T} (t \cdot |T_c|) dt \tag{31}$$

where t_0 and T represent the start and duration of the experiment, respectively, and T_c is the computation time of a single MPC calculation. α, γ, λ and δ are the weighting parameters for the tilt angle of the center of mass, displacement, average torque, and computation time, respectively, with $\alpha + \gamma + \lambda + \delta = 1$. The remaining parameters are defined in Equation (5).

By setting the initial values for feasible solutions and specifying the population size and number of iterations, the cost function can be iteratively optimized according to Equation (26). The main optimization process is shown in Figure 5.

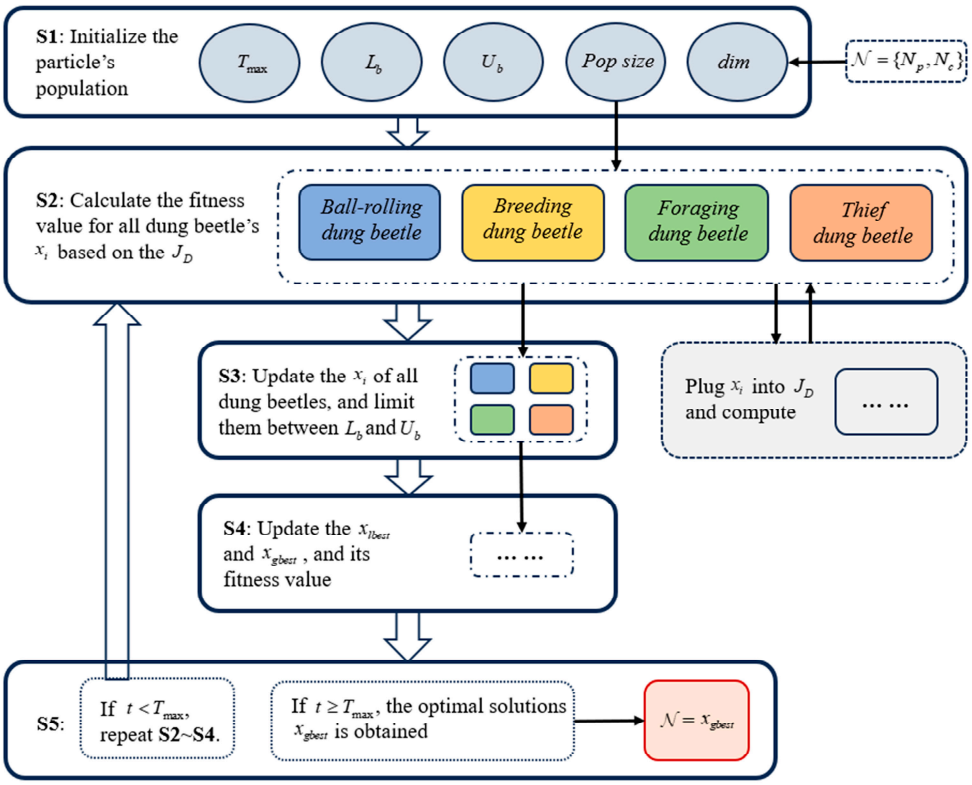


Figure 5. The Process of DBO Optimizing N.

When calculating the position information x_i for each dung beetle, the corresponding horizon parameter combination $\mathcal{N} = \{N_p, N_c\}$ is transmitted into the state-space equations of the wheeled biped robot, and different types of disturbance simulations are conducted. Based on the simulation results, the cost function J_D is calculated, updating the current optimal position and the position of each dung beetle in the next iteration. Among the four position update methods for dung beetles, the ball-rolling dung beetle serves as the basis for iterative optimization to obtain the optimal solution, with the solution given by Equations (21) and (22). Additionally, the breeding, foraging, and thief dung beetles perform extended iterative optimization around the optimal solution, with the solutions given by Equations (23)–(25), further improving the optimization efficiency.

During disturbance simulations, when N_c is much smaller than N_p, MPC may fail to capture key features of the system response, thereby affecting control performance. When N_c exceeds N_p, the insufficient prediction horizon may result in model distortion. Therefore, a larger cost function value should be assigned for these two scenarios. Moreover, during each simulation, the single computation time T_c obtained through the tic and toc methods may vary slightly due to system frequency changes. Thus, the mean value of T_c over the simulation *Step_size* should be calculated before computing the cost function. The disturbance simulation and cost function calculation process are shown in Figure 6.

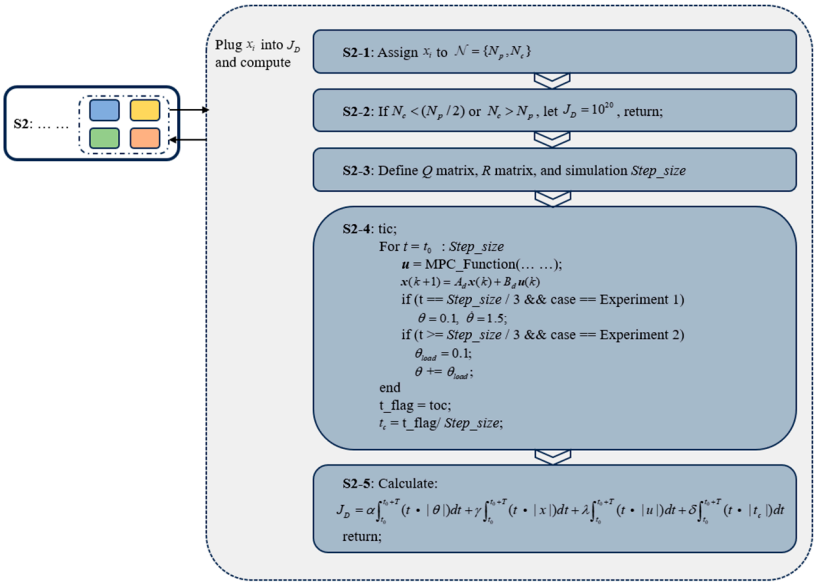


Figure 6. DBO Disturbance Simulation and Calculation Process.

In summary, by reasonably assigning the weight parameters in J_D, the DBO can be used to obtain an MPC algorithm that balances computational cost and control precision, enabling the WBR to achieve better anti-disturbance performance within a shorter computation time.

4. Anti-Disturbance Experiments for the WBR

This section will experimentally validate the ability of the wheeled biped robot using the MPC algorithm to restore balance under disturbance scenarios and compare it with the commonly used LQR algorithm. Additionally, it will analyze the impact of prediction horizon N_p and control horizon N_c on anti-disturbance performance under DBO, PSO, and non-optimized conditions.

4.1. Experimental Platform Setup

To validate the control algorithm proposed in this paper, a serial-structure wheeled bipedal robot (WBR) named SRobo110-II, independently designed and constructed by our laboratory, is used as the experimental platform [32]. The complete structure of SRobo110-II is shown in Figure 7. Its main hardware components include a TB48S battery, an N100 nine-axis Inertial Measurement Unit (IMU), HT-8115 joint motors, MF-9025 hub motors, and an Intel i5-10400H industrial computer connected externally. The IMU transmits data such as the angles and angular velocities of each axis to the industrial computer. The auxiliary wheels allow the robot to perform steering movements when powered off. Two batteries are connected in series to provide a 45.6 V power supply. The joint motors and hub motors respond to commands issued by the industrial computer, driving the links and wheels to achieve posture adjustment and movement. They also return status information such as torque and encoder positions. The serial structure of the robot is shown in Figure 8. The detailed structure and hardware parameters of the robot are listed in Table 1.

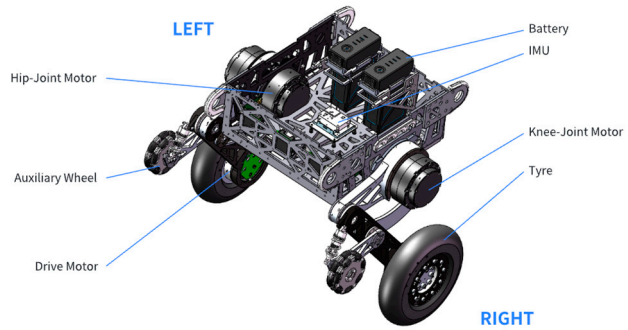


Figure 7. Three-dimensional Model of the Complete Structure of SRobo110-II.

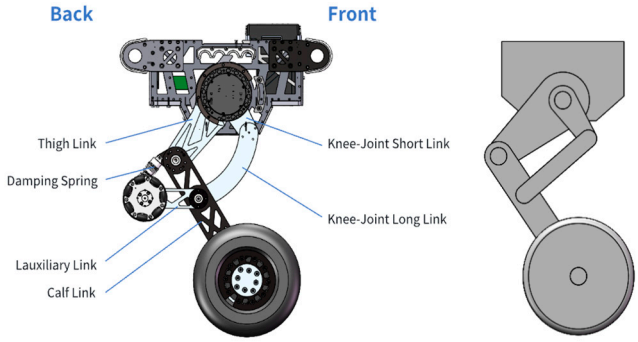


Figure 8. Side View and Schematic of the Linkage Mechanism of SRobo110-II.

Table 1. Structure and Hardware Parameters of SRobo110-II.

Parameter Type	Value
Total Mass (excluding battery)	13.83 kg
Maximum Body Height	544.77 mm
Minimum Body Height	338.39 mm
Battery Voltage	45.60 V
IPC CPU Peak Frequency	2.90 GHz
IMU Sampling Frequency	200.00 Hz
Rated Torque of Hub Motor	2.42 N·m
Peak Torque of Hub Motor	4.50 N·m

4.2. Disturbance Experiment Setup

In practical working environments, wheeled biped robots are often exposed to disturbances such as unstructured terrain, external impacts, and vertical loads. To comprehensively validate the effectiveness and superiority of the MPC horizon parameter optimization method based on DBO, as well as to evaluate the disturbance rejection capability of SRobo110-II, three types of experiments are designed in this study. First, in Experiment I, DBO is compared with PSO and ACO [33] on a simulation platform for optimizing the prediction and control horizons of MPC, and the control performance of different optimization algorithms under disturbance conditions is analyzed. Second, in Experiment II, a pendulum device is constructed using a rope and a 2 kg metal ball. The ball is pulled 1 m horizontally to the front of the robot and then released, so that it collides with the robot's chest region—while in a balanced state—at the lowest point of its free swing. The impact velocity is determined by the pendulum length and release height, simulating sudden external forces encountered by the robot during locomotion. Finally, in Experiment III, additional mass blocks are gradually mounted onto the robot body to generate vertical load disturbances, in order to observe its posture adjustment process under load. The setups and procedures of the three experiments are illustrated in Figures 9 and 10.

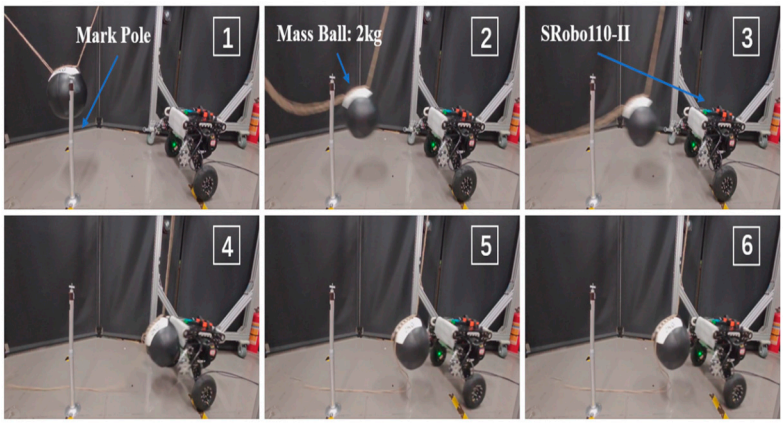


Figure 9. Setup and Process of Expt. 1.

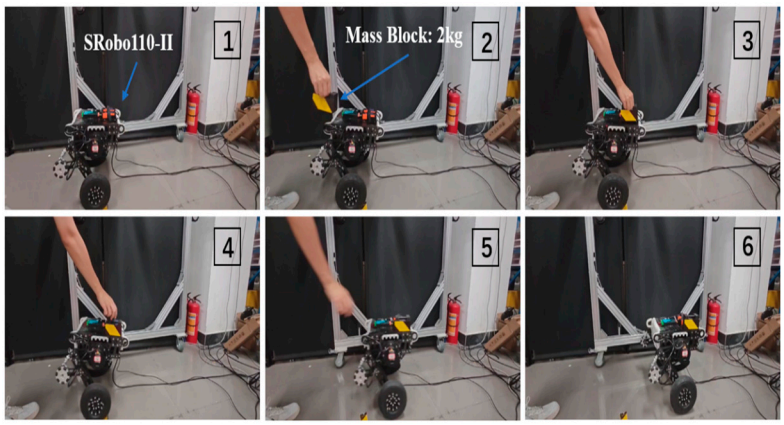


Figure 10. Setup and Process of Expt. 2.

4.3. Expt. 1 (Simulation): Process and Result Analysis

To validate the performance of the DBO algorithm in MPC horizon parameter optimization and to systematically assess the applicability of its optimization results for disturbance rejection control of wheeled biped robots, preliminary experiments were conducted in the Webots simulation environment. As an open-source 3D robotics platform, Webots is equipped with a high-precision physics engine capable of accurately simulating essential physical effects such as gravity, friction, collision, and inertia, thus providing a reliable virtual testing environment for control algorithm verification.

The simulation experiments were carried out on a hardware platform equipped with an AMD Ryzen 7 5800H CPU (base frequency 3.20 GHz). In the Webots project, a standard floor was set as the primary interaction environment for the robot, with physical parameters carefully configured to ensure compliance with the rolling friction model assumptions between the driving wheels and the ground. Based on the wheeled biped robot platform described in Section 4.2, its simplified model was imported into Webots via the Unified Robot Description Format (URDF), with precise configuration of physical properties such as mass and inertia tensors. The structure of the imported simplified model is shown in Figure 11.

Webots also provides a comprehensive controller interface that supports co-simulation with the Visual Studio development environment. In addition to the 3D model, virtual IMU sensors, joint motors, and wheel-driving motors were configured at key positions on the robot's body and joints. The IMU sensor continuously captured the robot's pitch angle, which was used as a key state variable for the wheeled inverted pendulum system. Through the Visual Studio controller project, the device driver functions and state feedback functions

provided by Webots enabled closed-loop interaction between robot motion control and sensor data acquisition.

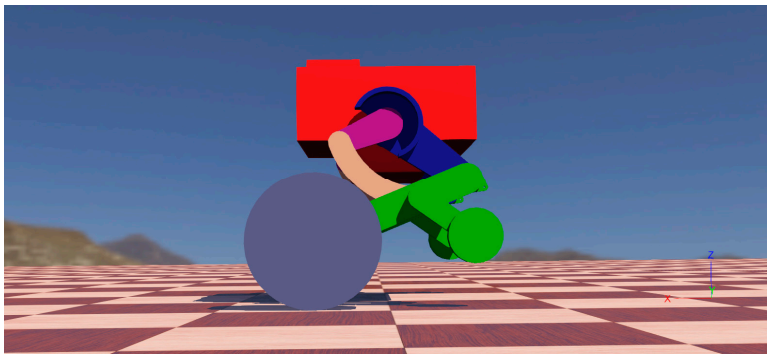


Figure 11. Simplified model of the wheeled biped robot in Webots.

To comprehensively evaluate the optimization efficiency of DBO, PSO and ACO were selected as benchmark heuristic algorithms for comparison. All three algorithms were executed under identical initial conditions, with parameter settings summarized in Table 2. To highlight the advantage of heuristic algorithms in reducing computational complexity, the maximum search count was set to 1/50 of the theoretical exhaustive count and then converted into the maximum number of iterations according to the population size. During the simulation, the step size was set to $Step_{size} = 1500$, and once the simulation time reached $t \geq Step_{size}/3$, the system parameter θ was dynamically adjusted to continuously simulate load disturbances. The final optimization results and performance comparisons are presented in Figure 12.

Table 2. Initial condition parameters of the three heuristic algorithms.

Parameters	DBO	PSO	ACO
Population size	10	10	10
Lower bound of feasible domain	1	1	1
Upper bound of feasible domain	100	100	100
Problem dimension	2	2	2
Maximum iterations	20	20	20
Subpopulation size (rolling/breeding/foraging/stealing dung beetles)	2/2/3/3	-	-
Inertia weight coefficient/damping ratio	-	1.2/0.99	-
Cognitive/social acceleration coefficients	-	1.5/2.5	-
Pheromone importance factor/heuristic information importance factor/pheromone evaporation rate/pheromone constant	-	-	1/2/0.5/100

From Figure 12, it can be seen that all three methods achieve a certain degree of iterative optimization. After 20 iterations, the optimal solution obtained by DBO yields a lower cost function value compared to PSO and ACO. The experimental results indicate that when the initial conditions are similar, the number of iterations is limited, and the computational time is comparable, DBO demonstrates superior exploration capability of the solution space. This advantage comes from its use of multiple subpopulations to perform extended iterative optimization around the optimal solution region. As a result, within a limited number of iterations, DBO exhibits better spatial search ability and is more suitable for multi-objective optimization tasks in solving MPC domain parameter combinations.

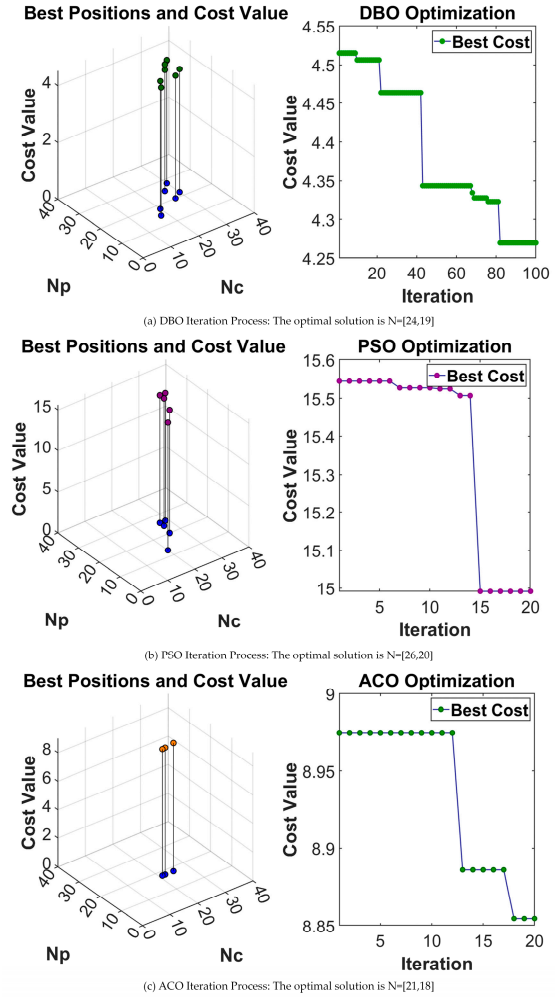


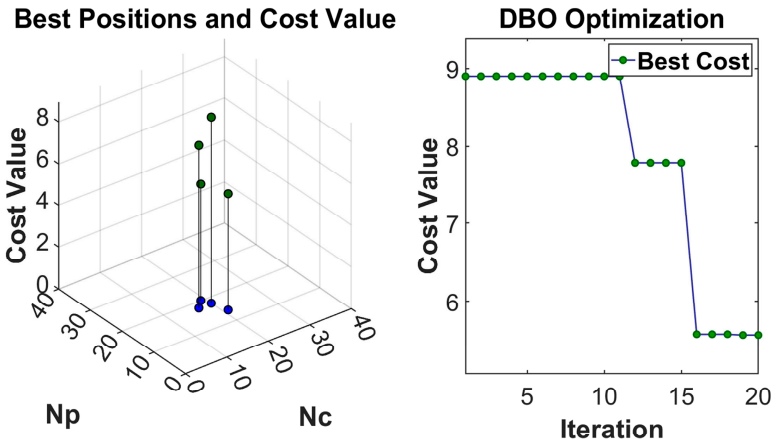
Figure 12. Iteration under Load Disturbance and Optimal Solution Distribution.

4.4. Expt. 2 (Impact): Process and Result Analysis

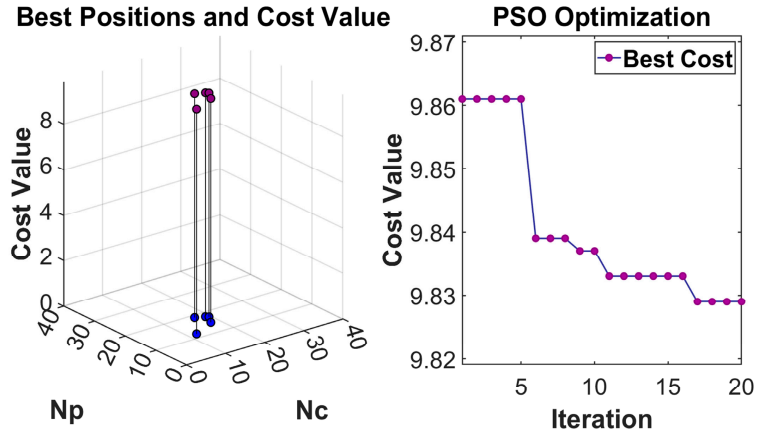
After completing the simulation verification in Experiment 1, further tests were conducted on the scenario where the robot is subjected to external impacts. The controller was iteratively optimized using DBO and the commonly used heuristic optimization algorithm PSO, and their control performances under impact disturbances were compared experimentally. The pseudocode of the specific implementation process is shown in Figure 6. The detailed parameter settings of the two optimization algorithms are consistent with those in Experiment 1, and the final optimization results are presented in Figure 13 and Table 3.

Table 3. Parameter Combinations for Impact Experiments.

Experiment Group	Value
Group 1: Parameters Optimized by DBO	[21, 14]
Group 2: Parameters Optimized by PSO	[18, 15]
Group 3: Non-optimized Parameters—Larger Values	[22, 19]
Group 4: Non-optimized Parameters—Smaller Values	[17, 14]



(a)DBO Iteration Process: The optimal solution is N=[21,14]



(b)PSO Iteration Process: The optimal solution is N=[18,15]

Figure 13. Iterative Process of Impact Simulation Using DBO and PSO.

Additionally, two sets of non-optimized parameters are provided—one greater than the maximum optimized value and one smaller than the minimum optimized value—to expand the experimental control groups. The four parameter combinations from Table 3 are applied to MPC for the impact experiments. The weight matrix is configured as:

$$Q = \begin{bmatrix} 70 & 0 & 0 & 0 \\ 0 & 0.55 & 0 & 0 \\ 0 & 0 & 35 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{32}$$

The experimental results are shown in Figure 14 and Tables 4 and 5.

Table 4. Impact Experiment Data for Different Parameter Groups—First Overshoot σ.

Experiment Group	Pitch (rad)	Displacement (m)	Torque (N·m)	Computation Time (ms)
Group 1	0.029	0.189	2.485	0.287
Group 2	0.078	0.368	2.295	0.313
Group 3	0.087	0.373	2.065	0.325
Group 4	0.069	0.236	2.653	0.266

Table 5. Impact Experiment Data for Different Parameter Groups—Adjustment Time t_s.

Experiment Group	Pitch (s)	Displacement (s)	Torque (s)
Group 1	0.494	2.142	0.700
Group 2	0.820	2.506	0.680
Group 3	1.212	2.108	-
Group 4	0.718	2.292	0.892

(Remark: "-" indicates that the data for this item could not be collected in the corresponding experimental group).

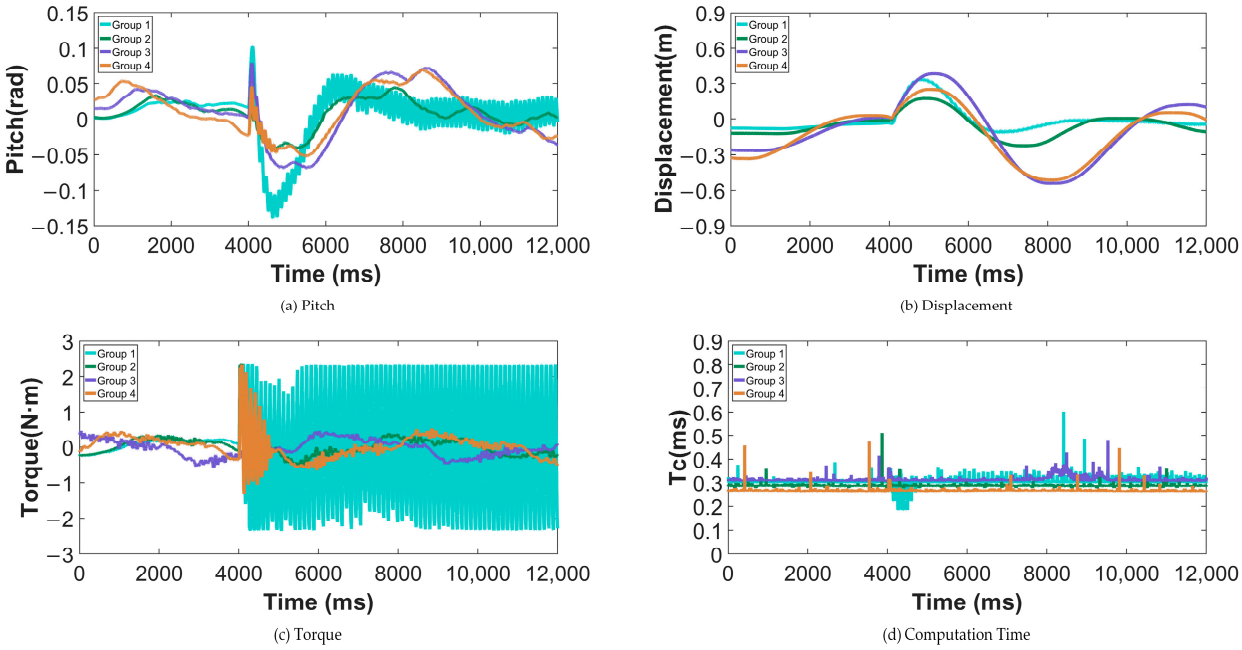


Figure 14. Impact Experiment Results for Different Parameter Groups.

By comparing the above experimental results, it can be seen that in all four groups of experiments, the robot was subjected to a horizontal impact from a pendulum mass at a specified moment, and data were analyzed over an 8 s interval following the disturbance. The results show that Group 1 (MPC optimized by DBO) achieved smaller first overshoots in both pitch angle and displacement compared with the other three groups, indicating a stronger capability in suppressing sudden attitude deviations and positional shifts. However, its peak torque and computation time were slightly higher. During the dynamic recovery process, Group 1 achieved the shortest adjustment time for pitch angle, while the displacement adjustment time remained at a normal level, and the torque adjustment time was slightly longer.

Further analysis suggests that this performance difference primarily stems from DBO's multi-subpopulation cooperative search mechanism, which effectively prevents premature convergence to local optima within limited iterations and enables the discovery of superior control parameter combinations on a global scale. This allows MPC to more proactively correct disturbance errors during prediction and constraint scheduling, thereby ensuring faster convergence of pitch angle and displacement. Although this mechanism leads to an increase in peak torque and some computational overhead, both remain within a reasonable range, demonstrating that MPC optimized by DBO exhibits remarkable advantages in enhancing dynamic performance and impact disturbance rejection.

After comparing the MPC performance optimized by different heuristic algorithms, it can be observed that MPC optimized by DBO shows certain advantages in disturbance suppression. However, to further verify its superiority over traditional controllers, this

study also designs an LQR-based control algorithm and conducts comparative experiments against the MPC algorithm using the Group 1 parameters. The cost function is as follows:

$$J_L = \frac{1}{2}\int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{33}$$

The experimental results are shown in Figure 15 and Tables 6 and 7.

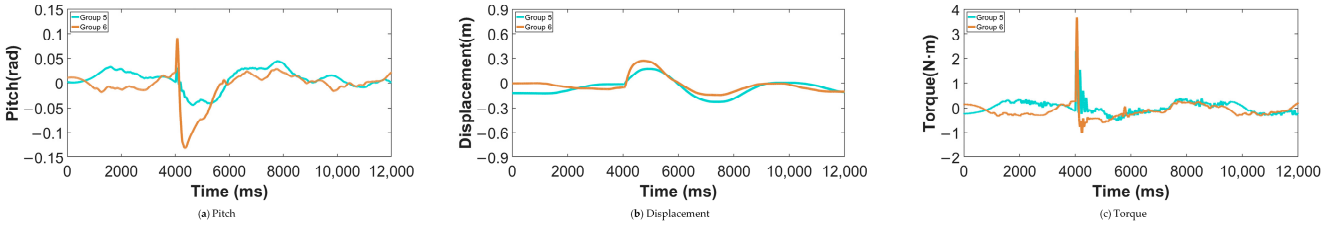


Figure 15. Impact Experiment Results for Different Control Algorithms.

Table 6. Impact Experiment Data for Different Control Algorithms—First Overshoot.

Experiment Group	Pitch (rad)	Displacement (m)	Torque (N·m)
Group 5: DBO-MPC	0.029	0.189	2.485
Group 6: LQR	0.070	0.327	3.375

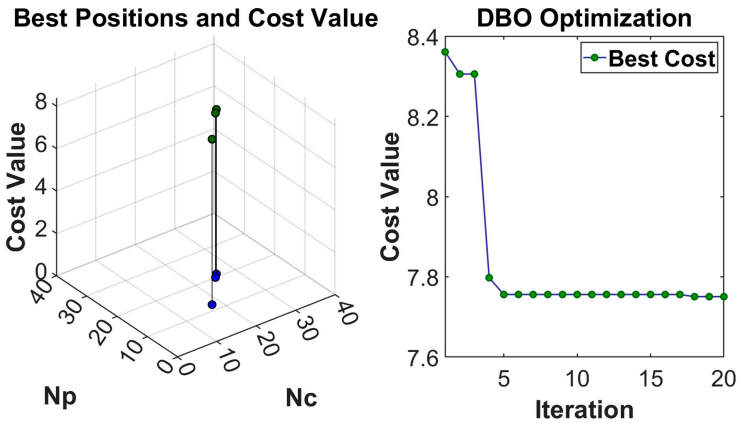
Table 7. Impact Experiment Data for Different Control Algorithms—Adjustment Time.

Experiment Group	Pitch (s)	Displacement (s)	Torque (s)
Group 5: DBO-MPC	0.494	2.142	0.700
Group 6: LQR	0.718	2.292	0.892

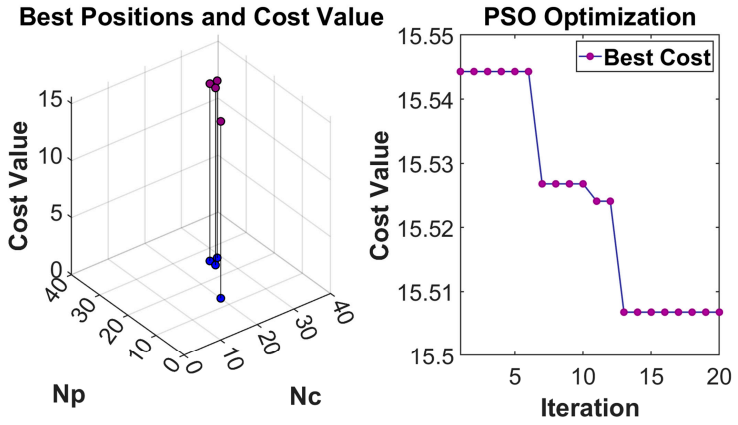
The results of the two sets of experiments show that, after being subjected to a horizontal impact at time $t = 4$ s, the robot controlled by MPC optimized with DBO exhibits significantly smaller overshoot in pitch angle, displacement, and torque compared with the robot controlled by LQR. Moreover, its pitch angle adjustment time is shorter, while the adjustment times for displacement and torque are slightly longer. The main reason for this difference lies in the rolling optimization mechanism of MPC, which explicitly considers future states and constraints during the control process, enabling the controller to plan ahead and suppress deviations caused by disturbances. In contrast, LQR relies on fixed feedback gains and can only provide a linear response to the current state, lacking foresight for future disturbances, which results in larger overshoot under strong perturbations. Furthermore, the DBO optimization process enables MPC to search for more suitable prediction and control horizons, as well as weight matrices, thereby achieving a better balance between rapid pitch recovery and smooth displacement adjustment, ultimately demonstrating superior disturbance rejection capability.

4.5. Expt. 3 (Load): Process and Result Analysis

Similarly to the process in Expt. 2, simulations and iterative optimizations are conducted for the robot under load disturbance using DBO and PSO. The pseudocode for the process is shown in Figure 6, and the initial values for both optimization algorithms are the same as in Expt. 1. In the cost function calculation, the simulation *Step_size* is also set to 1500, and θ is continuously adjusted at all times after $t >= Step_size/3$ to simulate load disturbance. The final optimization results are shown in Figure 16 and Table 8.



(a) DBO Iteration Process: The optimal solution is N=[24,19]



(**b**) PSO Iteration Process: The optimal solution is N=[20,13]

Figure 16. Iterative Process of Load Simulation Using DBO and PSO.

Table 8. Parameter Combinations for Load Experiments.

Experiment Group	Value
Group 7: Parameters Optimized by DBO	[24, 19]
Group 8: Parameters Optimized by PSO	[20, 13]
Group 9: Non-optimized Parameters—Larger Values	[26, 20]
Group 10: Non-optimized Parameters—Smaller Values	[18, 16]

Similarly, two sets of non-optimized parameters are provided—one greater than the maximum optimized value and one smaller than the minimum optimized value—to expand the experimental control groups. The four parameter combinations from Table 8 are applied to the MPC algorithm for the load disturbance experiments. The weight matrix configuration is provided in Equation (23). The experimental results are shown in Figure 17 and Tables 9 and 10.

Table 9. Load Experiment Data for Different Parameter Groups—First Overshoot σ.

Experiment Group	Pitch (rad)	Displacement (m)	Torque (N·m)	Computation Time (ms)
Group 7	−0.097	0.397	0.964	0.337
Group 8	−0.131	0.756	0.732	0.285
Group 9	−0.100	0.315	0.843	0.342
Group 10	−0.126	1.149	0.688	0.266

Table 10. Load Experiment Data for Different Parameter Groups—Adjustment Time.

Experiment Group	Pitch (s)	Displacement (s)	Torque (s)
Group 7	2.982	2.986	1.368
Group 8	3.950	-	5.506
Group 9	4.088	4.670	1.820
Group 10	-	-	4.744

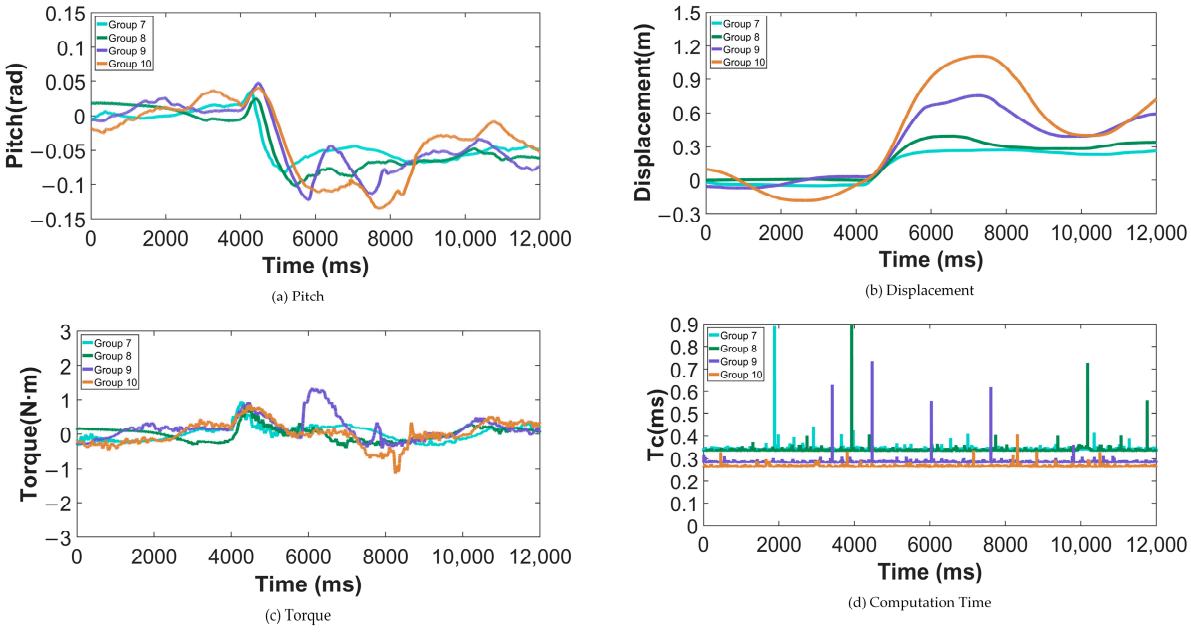


Figure 17. Load Experiment Results for Different Parameter Groups.

The results of the four load disturbance experiments show that MPC optimized with DBO (Group 7) outperforms the other three groups in terms of both the first overshoot and adjustment time of the pitch angle, demonstrating stronger posture recovery capability. The displacement index remains at a normal level, while the torque exhibits certain overshoot despite having a relatively better adjustment time, with computation time being slightly higher. Overall, the remarkable advantage of DBO-optimized MPC in pitch angle control stems from its subpopulation parallel search and global–local balance mechanism, which enable parameter configurations to better reconcile prediction accuracy and control stability, thereby effectively suppressing posture deviations caused by load disturbances. In contrast, the displacement and torque indicators show relatively neutral performance, reflecting that under complex coupled disturbances, the optimization mainly concentrates its performance gains on the posture dimension to ensure overall balance and safety of the robot when subjected to vertical loads.

Similarly, a comparative experiment is conducted using the designed LQR control algorithm and the MPC algorithm with the aforementioned seven-parameter configuration, with the experimental results shown in Figure 18 and Tables 11 and 12.

Table 11. Load Experiment Data for Different Control Algorithms—First Overshoot σ.

Experiment Group	Pitch (rad)	Displacement (m)	Torque (N·m)
Group 11: DBO-MPC	−0.097	0.397	0.964
Group 12: LQR	−0.117	0.470	0.887

Table 12. Load Experiment Data for Different Control Algorithms—Adjustment Time t_s.

Experiment Group	Pitch (s)	Displacement (s)	Torque (s)
Group 11: DBO-MPC	2.982	2.986	1.368
Group 12: LQR	2.242	2.524	1.972

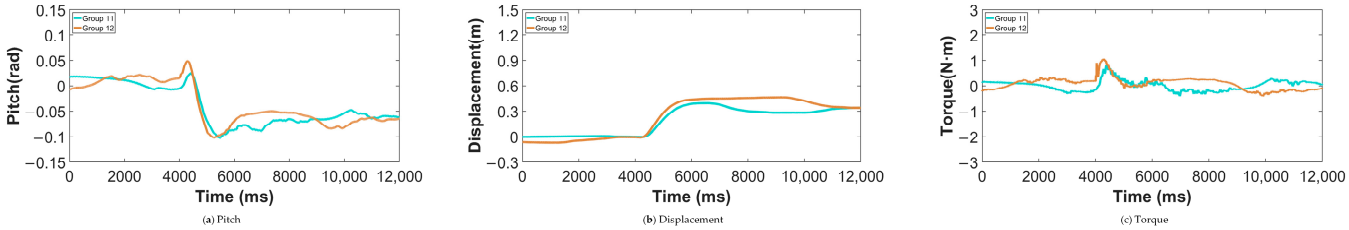


Figure 18. Load Experiment Results for Different Control Algorithms.

The comparison results of the two load disturbance experiments show that MPC significantly outperforms LQR in terms of overshoot of pitch angle and displacement, effectively reducing transient deviations caused by load disturbances, though its adjustment time is slightly longer; the torque adjustment time, however, performs well. This is because MPC optimizes future states in advance based on a predictive model, suppressing abrupt changes in attitude and displacement during the initial phase of the disturbance, whereas LQR, as a linear feedback control, struggles to timely compensate for transient deviations caused by nonlinear disturbances, though it has certain advantages in convergence speed. This indicates that MPC is more suitable for maintaining robot attitude stability under complex load disturbances, while LQR has relative advantages in rapid convergence.

5. Conclusions

To address the trade-off between computational cost and control precision, while enhancing the anti-disturbance capability of wheeled bipedal robots, this paper proposes a constrained Model Predictive Control (MPC) algorithm tailored to the hardware of the robot, with parameter optimization and experimental validation. The full text is summarized as follows:

(1) For the wheeled motion of the wheeled bipedal robot, it is equivalently simplified to a wheeled inverted pendulum model with a variable pendulum length. A dynamic model is constructed to derive the system's state-space model, with the balanced posture defined.

(2) Based on the hardware characteristics of the robot's drive motors, a soft-constrained MPC algorithm is proposed to prevent long-term control inputs from exceeding the rated torque, ensuring that the control input remains below the peak torque.

(3) The DBO algorithm is used to appropriately select the prediction and control horizons. A cost function is defined with the goal of balancing the computational cost and control precision of the MPC, and the optimal combination for anti-disturbance is obtained through iterative optimization.

(4) The SRobo-II experimental platform is constructed, and impact and load disturbance experiments are conducted on the wheeled bipedal robot. Parameters obtained through different optimization methods and control algorithms are compared and analyzed.

The experimental results show that the MPC algorithm optimized with DBO achieves smaller overshoot and faster adjustment time under both impact and load disturbances, with overall performance superior to PSO, ACO, the non-optimized version, and the LQR control algorithm. It should be noted that the MPC algorithm designed and optimized in this work is still based on a simplified equivalent first-order wheeled inverted pendulum

model. When the robot's height or posture changes significantly, the model needs to be reconstructed or adjusted. Therefore, future research will focus on further refining the dynamic model, fully accounting for the influence of structural variations such as the torso, and exploring hierarchical control architectures to achieve stable and disturbance-resistant control under varying pose conditions.

Author Contributions: Conceptualization, W.C. and Y.F.; methodology, W.C.; software, W.C.; validation, W.C., Y.F. and T.Z.; formal analysis, W.C.; investigation, W.C.; resources, C.P.; data curation, W.C.; writing—original draft preparation, Y.F.; writing—review and editing, W.C. and C.P.; visualization, W.C.; supervision, T.Z.; project administration, T.Z.; funding acquisition, W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Guangdong Province (Project No. 2024A1515012637) and the Key Research and Development Project of Guangdong Province (Project No. 2021B0101420003), China. The APC was funded by Doctoral Fund of Guangzhou City University of Technology (Grant No. KY200102).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: The authors would like to thank the Robotics Laboratory of South China University of Technology for providing the experimental facilities. During the preparation of this manuscript, the authors used language editing tools for polishing and MATLAB R2024b for data analysis. All generated content has been carefully reviewed and edited by the authors, who take full responsibility for the final version of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Boston Dynamics. Legacy Robot: Handle. 2019. Available online: http://www.bostondynamics.com/legacy (accessed on 1 May 2024).
2. Mao, N.; Chen, J.; Spyrakos-Papastavridis, E.; Dai, J.S. Dynamic modeling of wheeled biped robot and controller design for reducing chassis tilt angle. *Robotica* **2024**, *42*, 2713–2741. [CrossRef]
3. Xin, Y.; Li, Y.; Chai, H.; Rong, X.; Ruan, J. Planning and Execution of Dynamic Whole-body Locomotion for a Wheeled Biped Robot on Uneven Terrain. *Int. J. Control. Autom. Syst.* **2024**, *22*, 1337–1348. [CrossRef]
4. Sikander, A.; Prasad, R. Reduced order modelling based control of two wheeled mobile robot. *J. Intell. Manuf.* **2017**, *30*, 1057–1067. [CrossRef]
5. Karthika, B.; Jisha, V.R. Nonlinear Optimal Control of a Two Wheeled Self Balancing Robot. In Proceedings of the 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 1–3 December 2020; pp. 1–6.
6. Cui, L.; Wang, S.; Zhang, J.; Zhang, D.; Lai, J.; Zheng, Y.; Zhang, Z.; Jiang, Z.P. Learning-Based Balance Control of Wheel-Legged Robots. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7667–7674. [CrossRef]
7. Dong, J.Y.; Liu, R.; Lu, B.; Guo, X.; Liu, H.W. LQR-based Balance Control of Two-wheeled Legged Robot. In Proceedings of the 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022; pp. 450–455.
8. Katayama, S.; Murooka, M.; Tazaki, Y. Model predictive control of legged and humanoid robots: Models and algorithms. *Adv. Robot.* **2023**, *37*, 298–315. [CrossRef]
9. Cychowski, M.; Szabat, K.; Orlowska-Kowalska, T. Constrained Model Predictive Control of the Drive System with Mechanical Elasticity. *IEEE Trans. Ind. Electron.* **2009**, *56*, 1963–1973. [CrossRef]
10. Mishra, A.; Bansal, K. Control of Two-Wheel Self-Balancing Robot: LQR and MPC Performance Analysis. In Proceedings of the 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, 24–25 February 2024; pp. 1–6.
11. Khatoon, S.; Chaturvedi, D.K.; Hasan, N.; Istiyaque, M. Optimal Controller Design for Two Wheel Mobile Robot. In Proceedings of the 2018 3rd International Innovative Applications of Computational Intelligence on Power, Energy and Controls with their Impact on Humanity (CIPECH), Ghaziabad, India, 1–2 November 2018; p. 5.

12. Yu, J.; Zhu, Z.; Lu, J.; Yin, S.; Zhang, Y. Modeling and MPC-Based Pose Tracking for Wheeled Bipedal Robot. *IEEE Robot. Autom. Lett.* **2023**, *8*, 7881–7888. [CrossRef]
13. Minouchehr, N.; Hosseini-Sani, S.K. Design of Model Predictive Control of Two-Wheeled Inverted Pendulum Robot. In Proceedings of the 3rd RSI/ISM International Conference on Robotics and Mechatronics (ICROM), Tarbiat Modares Univ, Tehran, Iran, 7–9 October 2015; pp. 456–462.
14. Kanneworff, M.; Belvedere, T.; Scianca, N.; Smaldone, F.M.; Lanari, L.; Oriolo, G. Task-Oriented Generation of Stable Motions for Wheeled Inverted Pendulum Robots. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 214–220.
15. Cao, H.X.; Lu, B.; Liu, H.W.; Liu, R.; Guo, X. Modeling and MPC-based balance control for a wheeled bipedal robot. In Proceedings of the 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022; pp. 420–425.
16. Fernandez, D.C.; Hollinger, G.A. Model Predictive Control for Underwater Robots in Ocean Waves. *IEEE Robot. Autom. Lett.* **2017**, *2*, 88–95. [CrossRef]
17. Li, X.; Gu, J.; Huang, Z.; Wang, W.; Li, J. Optimal design of model predictive controller based on transient search optimization applied to robotic manipulators. *Math. Biosci. Eng.* **2022**, *19*, 9371–9387. [CrossRef]
18. Kashyap, A.K.; Parhi, D.R. Optimization of stability of humanoid robot NAO using ant colony optimization tuned MPC controller for uneven path. *Soft Comput.* **2021**, *25*, 5131–5150. [CrossRef]
19. Chen, Z.; Lai, J.; Li, P.; Awad, O.I.; Zhu, Y. Prediction Horizon-Varying Model Predictive Control (MPC) for Autonomous Vehicle Control. *Electronics* **2024**, *13*, 1442. [CrossRef]
20. Jin, M.; Li, J.; Chen, T. Method for the Trajectory Tracking Control of Unmanned Ground Vehicles Based on Chaotic Particle Swarm Optimization and Model Predictive Control. *Symmetry* **2024**, *16*, 708. [CrossRef]
21. Qazani, M.R.C.; Tabarsinezhad, F.; Asadi, H.; Khanam, S.; Arogbonlo, A.; Nahavandi, D.; Mohamed, S.; Lim, C.P.; Nahavandi, S. Optimal MPC Horizons Tunning of Nonlinear MPC for Autonomous Vehicles Using Particle Swarm Optimisation. In Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 9–12 October 2022; pp. 635–641.
22. Xue, J.; Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **2022**, *79*, 7305–7336. [CrossRef]
23. Yang, P.; Sun, L.; Zhang, M.; Chen, H. A lightweight optimal design method for magnetic adhesion module of wall-climbing robot based on surrogate model and DBO algorithm. *J. Mech. Sci. Technol.* **2024**, *38*, 2041–2053. [CrossRef]
24. Li, Y.; Sun, K.; Yao, Q.; Wang, L. A dual-optimization wind speed forecasting model based on deep learning and improved dung beetle optimization algorithm. *Energy* **2024**, *286*, 129604. [CrossRef]
25. Cao, W.; Liu, Z.; Song, H.; Li, G.; Quan, B. Dung Beetle Optimized Fuzzy PID Algorithm Applied in Four-Bar Target Temperature Control System. *Appl. Sci.* **2024**, *14*, 4168. [CrossRef]
26. Liu, F.; Luo, J.; Mo, J.; Gao, C.; Song, Z. Modeling and analysis of rigid-flexible coupling dynamics of a cable-driven manipulator. *J. Mech. Sci. Technol.* **2024**, *38*, 4377–4384. [CrossRef]
27. Zhao, H.; Yu, L.; Qin, S.; Jin, G.; Chen, Y. Design and Control of a Bio-Inspired Wheeled Bipeda Robot. *IEEE/ASME Trans. Mechatron.* **2025**, *30*, 2461–2472. [CrossRef]
28. Driels, M.R.; Fan, U.J.; Pathre, U.S. The application of newton-euler recursive methods to the derivation of closed form dynamic equations. *J. Robot. Syst.* **2007**, *5*, 229–248. [CrossRef]
29. Liu, T.; Zhang, C.; Wang, J.; Song, S.; Meng, M.Q.H. Towards Terrain Adaptablity: In Situ Transformation of Wheel-Biped Robots. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3819–3826. [CrossRef]
30. Qin, S.J.; Badgwell, T.A. A survey of industrial model predictive control technology. *Control. Eng. Pract.* **2003**, *11*, 733–764. [CrossRef]
31. Mudi, R.K.; Pal, N.R. A robust self-tuning scheme for PI- and PD-type fuzzy controllers. *IEEE Trans. Fuzzy Syst.* **1999**, *7*, 2–16. [CrossRef]
32. Zhang, A.; Zhou, R.; Zhang, T.; Zheng, J.; Chen, S. Balance Control Method for Bipedal Wheel-Legged Robots Based on Friction Feedforward Linear Quadratic Regulator. *Sensors* **2025**, *25*, 1056. [CrossRef] [PubMed]
33. Dorigo, M.; Gambardella, L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Computat.* **1997**, *1*, 53–66. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.