DE-optimized LQR for trajectory tracking of omnidirectional mobile robots

Van Vuong Dinh[1*] **and Thi Doan Trang Nguyen**[2]

[1]*Department of Control and Automation, Electric Power University*
[2]*Faculty of Electrical and Electronic Engineering, Hanoi College of High Technology*
[*]*Corresponding author E-mail: vuongdv@epu.edu.vn*
DOI: https://doi.org/10.64032/mca.v30i1.404

Abstract

Omnidirectional mobile robots (OMRs) have become increasingly important in modern automation due to their ability to navigate freely in any planar direction, making them highly suitable for industrial logistics, service robotics, and collaborative systems. However, achieving a balance between tracking accuracy and actuator limitations under physical constraints remains challenging. The Linear Quadratic Regulator (LQR) is an effective optimal controller, but its performance depends heavily on manually tuned weighting matrices. To address this issue, this study integrates the Differential Evolution (DE) algorithm with the LQR controller to automatically optimize these parameters. The proposed controller significantly improves tracking accuracy and eliminates the need for manual tuning making it highly suitable for advanced control of OMRs. Simulation results, compared with those of a conventional Proportional–Integral (PI) controller and a DE-optimized PI controller, confirm that the DE-LQR strategy provides superior convergence behavior, lower tracking error, and enhanced stability.

Keywords: *Differential evolution; Inertial measurement units; Linear quadratic regulator; Omnidirectional mobile robots; Proportional integral*

Abbreviations

OMRs	Omnidirectional mobile robots
LQR	Linear Quadratic Regulator
DE	Differential Evolution
GA	Genetic Algorithms
PI	Proportional Integral
IMUs	Inertial Measurement Units
SMC	Sliding Mode Control
PSO	Particle Swarm Optimization

1. Introduction

Nowadays, mobile robots have become an essential component in modern automation due to their versatility and efficiency across diverse environments, including manufacturing, logistics, defense, and service sectors. Among various ground mobile platforms, the OMRs have received considerable attention for their unique ability to move freely in any planar direction without reorienting the chassis. This property enables flexible navigation in confined or restricted environments. Therefore, OMRs particularly suitable for warehouse automation, material handling, and cooperative robotic systems [1, 2].

A standard three-wheel OMRs features a symmetric chassis equipped with three omnidirectional wheels arranged 120° apart, each driven independently by a dedicated motor. The platform typically integrates onboard sensors, such as cameras or Inertial Measurement Units (IMUs), for motion estimation and feedback, along with an embedded controller for real-time motor command generation [3, 4]. This mechanical configuration offers a mathematically tractable model that enables the development of advanced control and motion-planning strategies.

However, OMRs control remains a challenging problem due to nonlinear dynamics, wheel–ground interaction variability, model parameter uncertainties, and external disturbances [5, 6]. These issues can lead to degraded tracking accuracy, oscillatory responses, or even instability during high-speed maneuvers or operation on uneven surfaces. Consequently, a wide variety of control strategies have been developed to enhance performance and robustness. In general, these methods can be divided into two main categories: model-free and model-based control.

Model-free control approaches such as reinforcement learning, fuzzy logic, and other data-driven techniques operate directly from system data without explicit dynamic modeling [7, 8]. Although these methods offer flexibility and adaptability to nonlinear or time-varying systems, their performance often depends heavily on large datasets and extensive training, which limits their real-time applicability. In addition, their lack of analytical stability guaranties poses a concern in safety-critical robotic operations.

In contrast, model-based control has remained the dominant paradigm for OMRs due to its analytical rigor, predictable performance, and ability to incorporate physical insight into control design [9, 10]. Strategies such as Sliding Mode Control (SMC), adaptive neural control, and dynamic surface control have been applied to address nonlinearities and external disturbances [11, 12]. Further advances, including disturbance observers and passivity-based designs, have improved robustness by compensating for unmodeled dynamics and parameter variations [13, 14]. Nevertheless, many of these approaches rely on centralized architectures that coordinate all

Received: 29 December 2025; Revised: 28 January 2026; Accepted: 11 February 2026.

actuators simultaneously, increasing computational complexity and reducing fault tolerance [15, 16].

Among model-based controllers, the LQR stands out due to its strong theoretical foundation and guaranteed optimality for linear or linearized systems. LQR provides an optimal state-feedback law that minimizes a quadratic cost function, balancing tracking accuracy and control effort [17, 18]. However, its effectiveness is highly dependent on the appropriate selection of the weighting matrices Q and R, which directly influence the transient response, steady-state performance, and actuator effort. Manual tuning of these matrices is often non-intuitive and may yield suboptimal results [19].

To overcome this limitation, researchers have increasingly adopted intelligent optimization techniques for automatic LQR tuning. Among these, the DE algorithm has demonstrated superior global search capability, simplicity, and robustness in handling multimodal nonlinear optimization problems [20, 21]. DE explores the parameter space efficiently without requiring gradient information, typically converging faster and more reliably than other evolutionary algorithms such as Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) [22]. When combined with LQR, DE can automatically optimize the weighting matrices to achieve enhanced transient response, reduced steady-state error, and improved disturbance rejection [23, 24].

This research develops a DE-algorithm-based LQR controller to achieve optimal control performance according to predefined requirements by adjusting the weighting matrix in the LQR design. With this approach, the LQR controller can be tuned to balance control performance and input limitations. By integrating the optimal control formulation of LQR with the adaptive global search capability of DE, the proposed method aims to improve motion accuracy, enhance robustness, and reduce the burden of manual controller tuning, making it suitable for practical robotic applications.

The main contributions of this research are summarized below:

- A DE-optimized LQR controller is developed, where the Differential Evolution algorithm automatically tunes the LQR weighting matrices. This eliminates manual trial-and-error tuning and yields an optimal set of gains for improved tracking accuracy.
- The simulation results, compared with those of the conventional PI and DE-optimized PI controllers, confirm that the proposed controller significantly enhances the control performance of OMRs, demonstrating improved convergence and overshoot responses, reduced tracking error, and lower control effort.

2. Kinematic and dynamic model

Consider a global coordinate frame $X_gO_gY_g$ and a body–fixed coordinate frame $X_rO_rY_r$ attached to the robot's center of mass, as shown in Figure 1. Let the robot's body velocities in Σ_r be denoted by $\mathbf{v}_r = [u_r \;\; v_r \;\; \omega_r]^T$, where u_r and v_r are the forward and lateral velocities, and ω_r is the yaw rate. Linear and angular velocities expressed in the global frame are denoted as $\dot{\mathbf{q}}_g = [\dot{x}_g \;\; \dot{y}_g \;\; \dot{\psi}_g]^T$.

The transformation between the body velocities and global

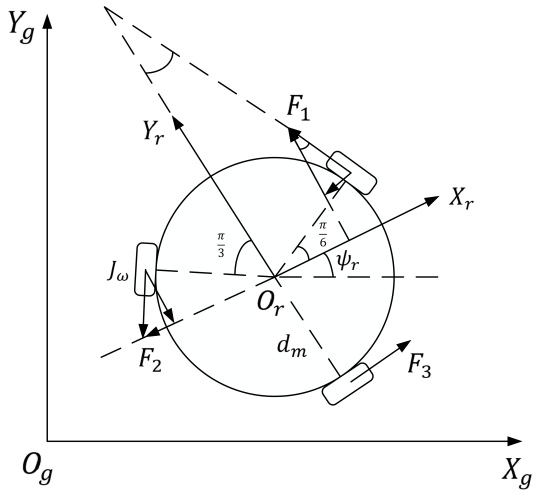

Figure 1: The kinematic model of a three-wheel OMR.

velocities is given by

$$\dot{\mathbf{q}}_g = \begin{bmatrix} \cos\psi_g & -\sin\psi_g & 0 \\ \sin\psi_g & \cos\psi_g & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}_r \tag{1}$$

The robot employs three omnidirectional wheels spaced 120° apart. Let ω_i denote the angular velocity of wheel i, r_w the wheel radius, and d_m the distance from the robot center to each wheel. The mapping from wheel angular velocities to the robot body velocities is

$$\mathbf{v}_r = r_w \begin{bmatrix} -\sin 30^\circ & -\sin 30^\circ & \sin 90^\circ \\ \cos 30^\circ & -\cos 30^\circ & \cos 90^\circ \\ \dfrac{1}{d_m} & \dfrac{1}{d_m} & \dfrac{1}{d_m} \end{bmatrix} \omega \tag{2}$$

where $\omega = [\omega_1 \;\; \omega_2 \;\; \omega_3]^T$.

For each wheel i, the torque balance equation is expressed as

$$J_w\dot{\omega}_i + b_f\omega_i = k_t\tau_i - r_wF_i \tag{3}$$

where J_w is the wheel inertia, b_f is the viscous friction coefficient, k_t is the constant motor torque, τ_i is the applied motor torque and F_i is the interaction force between the wheel and the robot body.

Let the robot's state vectors be

$$\mathbf{x}_1 = [x_g \;\; y_g \;\; \psi_g]^T \qquad \mathbf{x}_2 = [u_r \;\; v_r \;\; \omega_r]^T$$

and the input torque vector be

$$\tau = [\tau_1 \;\; \tau_2 \;\; \tau_3]^T$$

The robot dynamics are written as

$$\begin{cases} m(u_r - v_r\omega_r) &= -\frac{1}{2}F_1 - \frac{1}{2}F_2 + F_3 \\ m(v_r - u_r\omega_r) &= \frac{\sqrt{3}}{2}F_1 - \frac{\sqrt{3}}{2}F_2 \\ J_r\dot{\omega}_r &= (F_1 + F_2 + F_3)d_m \end{cases} \tag{4}$$

where m is the robot mass and J_r is the rotational inertia.

After substituting (3) into (4), the dynamic model becomes

$$\begin{cases} \dot{\mathbf{x}}_1 = R(\psi_g)\mathbf{x}_2 \\ \dot{\mathbf{x}}_2 = A\mathbf{x}_2 + a_2E(\mathbf{x}_2) + B\tau \end{cases} \tag{5}$$

where

$$A = \text{diag}(a_1, a_1, a_3), \qquad E(\mathbf{x}_2) = \begin{bmatrix} \omega_r v_r \\ -\omega_r u_r \\ 0 \end{bmatrix}$$

$$R(\psi_g) = \begin{bmatrix} \cos\psi_g & -\sin\psi_g & 0 \\ \sin\psi_g & \cos\psi_g & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$B = \begin{bmatrix} -p_1 & -p_1 & 2p_2 \\ \sqrt{3}p_1 & -\sqrt{3}p_1 & 0 \\ p_2 & p_2 & p_2 \end{bmatrix}$$

The parameters are

$$a_1 = -\frac{3b_f}{3J_w + 2mr_w^2} \qquad a_2 = \frac{2mr_w^2}{3J_w + 2mr_w^2}$$

$$a_3 = \frac{3b_f d_m^2}{3J_w d_m^2 + J_r r_w^2} \qquad p_1 = \frac{k_t r_w}{3J_w + 2mr_w^2}$$

$$p_2 = \frac{k_t r_w d_m}{3J_w d_m^2 + J_r r_w^2}$$

Equation (5) forms the complete state-space model used to describe the dynamics of the OMRs system and to validate the efficiency of the proposed method.

3. Controller design

In this study, a hierarchical control structure is employed, which is represented as follows:

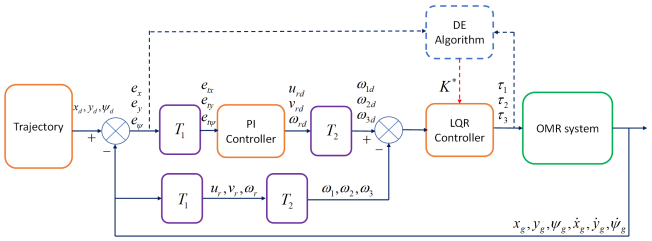

Figure 2: The control structure.

In this architecture, the desired trajectory is predefined, and the tracking errors in the fixed coordinate frame with respect to the system states are determined as follows:

$$\begin{cases} e_x = x_d - x_g \\ e_y = y_d - y_g \\ e_\theta = \psi_d - \psi_g \end{cases} \tag{6}$$

With the transformation matrix:

$$T_1 = \begin{bmatrix} \cos(\psi_g) & -\sin(\psi_g) & 0 \\ \sin(\psi_g) & \cos(\psi_g) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \tag{7}$$

The tracking errors along the body axes of the vehicle are denoted as $e_{tx}, e_{ty}, e_{t\psi}$, which corresponds, respectively, to the position errors along x, y, ψ of the body of the vehicle. The PI controller is then used to generate $u_{rd}, v_{rd}, \omega_{rd}$, which represent the desired linear velocities along the x, y directions and the desired angular velocity of the vehicle.

In this control loop, three PI controllers are designed as follows:

- **PI 1:** Input: e_{tx}, Output: u_{rd}
- **PI 2:** Input: e_{ty}, Output: v_{rd}
- **PI 3:** Input: $e_{t\psi}$, Output: ω_{rd}

With the transformation matrix:

$$T_2 = \frac{1}{r_w} \begin{bmatrix} -\sin(30) & -\sin(30) & \sin(90) \\ \cos(30) & -\cos(30) & \cos(90) \\ 1/d_m & 1/d_m & 1/d_m \end{bmatrix}^{-1} \tag{8}$$

The desired angular velocity of each wheel can then be calculated.

These three outputs of the outer control loop are used to generate the desired velocity vector for each wheel. This requires a transformation matrix defined as:

$$\begin{bmatrix} \omega_{1d} \\ \omega_{2d} \\ \omega_{3d} \end{bmatrix} = T_2 \begin{bmatrix} u_{rd} \\ v_{rd} \\ \omega_{rd} \end{bmatrix} \tag{9}$$

The transformation matrix calculates the reference angular velocities for each wheel. In this case, the notation of reference angular velocities ω_{id}, $i = 1, 2, 3$, implies that the torque control loop aims to provide sufficient torque to the motors at each wheel to achieve the desired reference velocity.

Assuming that the three wheels have identical dynamics, let θ_i denote the angular position of the i-th wheel. The dynamics of one representative wheel can be expressed as:

$$\begin{cases} \dot{\theta}_i = \omega_i \\ J_w \dot{\omega}_i + b_f \omega_i = k_t \tau_i \end{cases} \tag{10}$$

Let $x_{\theta_i} = \begin{bmatrix} \theta_i & \dot{\theta}_i \end{bmatrix}^T$

Then, the state-space equation describing the motor dynamics of each wheel is given by:

$$\dot{x}_{\theta_i} = A_i x_{\theta_i} + B_i \tau_i \tag{11}$$

where:

$$A_i = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{b_f}{J_w} \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ \frac{k_t}{J_w} \end{bmatrix}$$

The objective is to control the torque so that the motor speed reaches the desired value. Therefore, the dynamics are transformed into the error dynamics as follows:

$$\begin{aligned} \dot{e}_{\omega i} &= \dot{x}_{\theta_i} - \dot{x}_{\theta_{id}} = A_i x_{\theta_i} + B_i \tau_i - \dot{x}_{\theta_{id}} \\ &= A_i e_{\theta_i} + A_i x_{\theta_{id}} + B_i \tau_i - \dot{x}_{\theta_{id}} \end{aligned} \tag{12}$$

where: $\dot{x}_{\theta_{id}} = \begin{bmatrix} \omega_{id} & \dot{\omega}_{id} \end{bmatrix}^T$

With this model, it is straightforward to design an LQR controller for each wheel. In the LQR controller design, the objective is to minimize the following cost function:

$$J = \int \left(x_{\theta_i}^T Q_i x_{\theta_i} + \tau_i^T R_i \tau_i \right) dt \tag{13}$$

The goal is to design a state feedback controller such that:

$$\tau_i = K_i e_{\omega_i} + B^{\dagger} \dot{x}_{\theta_{id}} - B_i^{\dagger} A_i x_{\theta_{id}} \tag{14}$$

Here $B_i^{\dagger}$ denotes the pseudo-inverse of B_i, K_i is the controller gain obtained through the LQR design, which can be determined by:

$$K_i = -R_i^{-1} B_i^T P_i \tag{15}$$

where the tracking error is defined as:

$$e_{\omega_i} = \omega_{id} - \omega_i \tag{16}$$

with P_i is a positive definite matrix and represents the solution of the Riccati equation.

It can be observed that, in this design, the performance of the LQR controller can be tuned by adjusting the weighting matrices Q_i and R_i.

Since the three motors are assumed to have identical dynamics, the weighting matrices in the cost function of the LQR controller are defined as follows:

$$Q_i = \text{diag}(q_{1i}, q_{2i}), \quad R_i = r_i \tag{17}$$

To optimize the LQR controller according to a predefined objective, the parameters q_{1i}, q_{2i}, r_i are adjusted to achieve the desired performance using an evolutionary optimization algorithm. The objective function chosen for this design is expressed as:

$$J_A = \sum_{i=1}^{N} a_i Z_i \tag{18}$$

Where a_i are positive weighting coefficients that determine the relative importance of each performance index, and Z_i is defined as:

$$Z_i = \sqrt{\int Z^2(t)\, dt} \tag{19}$$

Here, $Z(t)$ is a time-dependent signal, which may represent the tracking error, control signal, or state variable of the robot over time.

In this study, to reduce the complexity of the Differential Evolution (DE) algorithm, only three parameters q_{1i}, q_{2i}, r_i are optimized.

Remark 1: *In a real robot system, it is often impractical to directly measure the body velocities* (u_r, v_r)*. Therefore, the feedback signals for the inner velocity control loop are usually obtained directly from encoders that measure the angular velocities of individual wheels. However, in the context of this simulation study, a different approach is applied. Since the robot body states* (u_r, v_r, ω_r) *are assumed to be fully known at each time instant, the wheel angular velocities used for feedback are theoretically calculated based on these known states.*

3.1 Principle of the DE algorithm

Initialization:

Define the search space. The DE algorithm allows the search to take place within a predefined region, denoted as Ω, which is bounded by two vectors Ω_u and Ω_l representing the upper and lower limits of the search space, respectively.

The initial population is created with M individuals, and this number remains constant throughout the algorithm. The population can be initialized either randomly or with fixed values, provided that all the initial individuals belong to the domain Ω.

Mutation:

In this step, three distinct vectors are randomly selected from the current population of M individuals, denoted as $X_{m_1}^G, X_{m_2}^G, X_{m_3}^G$, where m_1, m_2, m_3 are different positive integers, and G represents the current generation. A mutant vector is then generated from these three vectors according to the following equation:

$$V_i^{G+1} = X_{m_1}^G + F\left(X_{m_2}^G - X_{m_3}^G\right) \tag{20}$$

where F is a mutation factor chosen within the range 0-2.

Crossover:

In this step, a new population is generated by creating a trial vector U_i^{G+1}. This vector is formed by combining one target vector X_i^G and one mutant vector V_i^{G+1} according to the binomial crossover rule as follows:

$$U_{ji}^{G+1} = \begin{cases} V_{ji}^{G+1}, & \text{if } rand_{ij} \le CR \text{ or } (j = I_{rand}) \\ X_{ji}^G, & \text{if } rand_{ij} > CR \text{ and } (j \ne I_{rand}) \end{cases} \tag{21}$$

The symbols U_{ji}^{G+1}, X_{ji}^G, and V_{ji}^{G+1} represent the j^{th} element of vector i in generations G and $G+1$, respectively.

The variable $rand_{ij}$ is a random number uniformly distributed between 0 and 1, while I_{rand} is a randomly chosen integer from 1 to D, where D is the dimension of the population vectors. This rule ensures that the trial vector U_{ji}^{G+1} always differs from all existing individuals in the population. The crossover rate CR is a control parameter chosen within the range 0-1, typically between 0.5 and 0.9.

Selection:

After obtaining the trial vector, the objective function value of this vector is evaluated and compared with that of the corresponding target vector in the initial population. The population update rule is then performed as follows:

$$X_i^{G+1} = \begin{cases} U_i^{G+1}, & f(U_i^{G+1}) < f(X_i^G) \\ X_i^G, & f(U_i^{G+1}) \ge f(X_i^G) \end{cases} \tag{22}$$

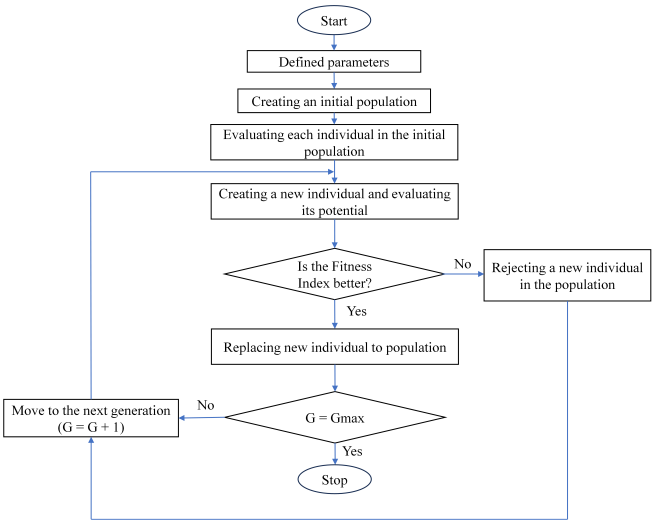

Figure 3: The flowchart of DE algorithm.

3.2 Applying the DE algorithm for LQR controller design

Step 1: Declare the necessary parameters for setting up the algorithm, including: $J_w, b_f, k_t, A_i, B_i, D$

Step 2: Initialize the population:

Define the search space by setting the upper and lower bound vectors:

$$[q_{1u}, q_{2u}, r_u]^T, \quad [q_{1l}, q_{2l}, r_l]^T$$

Set the initial population size M and initialize the population X with M vectors. Determine the number of generations G and algorithm parameters including F and CR.

Step 3: Establish the iteration loop of the algorithm:

Define the objective function: $J_A = \sum_{i=1}^{N} a_i Z_i$

Step 3.1: Evaluate the initial generation

```
Function min(J_A) = Fitness(X)
convCurve = zeros(MaxGen,1);
for  i = 1 : M
  K{i} = lqr(A,B,Q,R);
  J_A{i} = sum_{j=1}^{N} a_j Z_j{i}(K{i});
  BestSol = min(J_A);
end
```

Step 3.2: Find the optimal value

```
for  i = 1 : G
  for  j = 1 : M
    temp = [a b c];%where a,b,c ∈ {1,2,...,M}
    X_m1 = X{temp{1}};
    X_m2 = X{temp{2}};
    X_m3 = X{temp{3}};
    VG1 = X_m1 + F(X_m2 − X_m3);
    I_rand = q;%q ∈ {1,2,...,D}
    U = X{i};
    for  k = 1 : D
      if  (I_rand == k) ∨ (rand ≤ CR)
        U(k) = VG1(k);
      end
    end
    newSol = Fitness(U);
    if  newSol < bestSol
      bestSol = newSol;
    end
  end
  convCurve(g) = bestSol;
end
return bestSol;
```

4. Simulations and results

To demonstrate that the proposed method can improve not only the control performance of the OMR system, this section presents two simulation cases, including nominal (uncertainty-free) and uncertain scenarios. The parameters of system model are presented in Table (1).

In the PI controller design for the position control loop, the controller parameters are selected as follows Table (2):

The objective function is defined as:

$$J_A = e_{xRMS} + e_{yRMS} + e_{\psi RMS} + 0.3\left(e_{\tau_1 RMS} + e_{\tau_2 RMS} + e_{\tau_3 RMS}\right) \tag{23}$$

where e_{xRMS}, e_{yRMS}, $e_{\psi RMS}$, $e_{\tau 1RMS}$, $e_{\tau 2RMS}$, and $e_{\tau 3RMS}$ denote the Root Mean Square Error (RMSE) values of position tracking errors along the x, y, ψ axes in the fixed coordinate frame and the RMSE of the input control signals, respectively.

Table 1: System model parameters

Symbol	Value	Unit
d_m	0.4	m
J_w	0.0418	kg m^2
J_r	11.25	kg m^2
b_f	0.1	kg m^2/s
r_w	0.05	Ω
m	9.4	kg
k_t	1	–

Table 2: PI controller parameters for the position control loop

Parameter	Value
K_{i11}	5
K_{i12}	5
K_{i13}	5
P_{i11}	2
P_{i12}	2
P_{i13}	2

This objective function is determined by calculating the RMSE indices of the signals over a specified time interval. The purpose of this objective function is to ensure that the controller achieves both accurate trajectory tracking and minimized starting torque simultaneously.

Parameter selection for the algorithm:

Parameter limits:

$$[1\ 1\ 1], \quad [40\ 40\ 40]$$

$$Number\ of\ population : 6$$

$$F = 0.95, \quad CR = 0.9$$

$$Number\ of\ generations : 30$$

$$Initial\ population :$$

$$\{[1\ 1\ 1];\ [1\ 2\ 3];\ [1.1\ 1\ 1.5];\ [2\ 2\ 2];\ [1\ 2\ 5];\ [2\ 5\ 7]\}$$

The controller parameters obtained after applying the DE algorithm are:

$$K = [0.158, 0.152].$$

Remark 2: *Intuitively, the optimal controller gain is affected by the selection of the parameters of the DE algorithm. However, a detailed discussion on the selection of these parameters is provided in [27]. Following the discussion in [27], NP is chosen to be at least 4 and typically ranges from 5D to 10D. However, in the application to OMR, the selection NP = 6 is adopted to ensure that the procedure can be executed quickly. Furthermore, although random initialization helps achieve fast convergence, in some cases, the optimal value may converge to a local solution. To adjust the convergence of the DE algorithm using only two parameters, F and CR, the initial population is fixed, and NP is chosen appropriately to ensure acceptable computational time. Normally, following [27], choosing F ∈ [0.4, 1] and CR ∈ [0; 1] is appropriate. Since the random initialization and population size are fixed, F and CR are chosen close to 1 to ensure fast convergence. It is noted that, in practical applications such as OMR, the selection of the objective function is also important to ensure that the search*

space does not include any singularity points. Furthermore the PI and LQR controllers are constructed based on separate structures, and randomly initialized parameters may lead to disruption in the DE algorithm.

To demonstrate the effectiveness of the proposed method, two PI controllers are selected for comparison. The conventional PI controller is tuned using a trial-and-error approach, with the objective of achieving a control effort (in terms of initial torque magnitude) approximately comparable to that of the proposed method; the resulting parameters are listed in Table (3). However, this approach is not technically rigorous or fully fair. Therefore, a DE-optimized PI controller (DE-PI) is also designed by tuning the PI parameters using the DE algorithm based on a common objective function (23). The optimized parameters of the DE-PI controller are provided in Table (4). This comparison between the conventional PI and the DE-PI controllers further demonstrates the performance improvement achieved by employing the DE algorithm for controller parameter tuning.

The value of the cost function after 30 generations (DE-LQR) is:

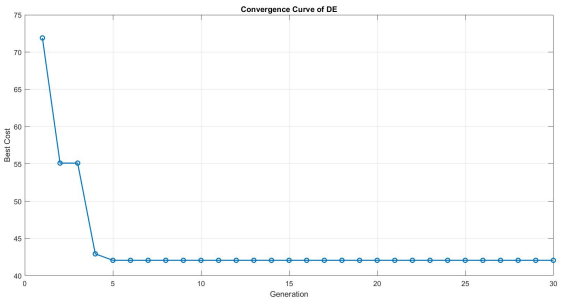

Figure 4: The cost function.

The proposed controller will be used to compare with a PI controller with the following parameters:

Table 3: PI controller parameters

Parameter	Value
K_{i21}	0.2
K_{i22}	0.2
K_{i23}	0.2
P_{i21}	0.1
P_{i22}	0.1
P_{i23}	0.1

Table 4: DE-optimized PI controller parameters

Parameter	Value
K_{i21}	0.2214
K_{i22}	0.2214
K_{i23}	0.2214
P_{i21}	0.1
P_{i22}	0.1
P_{i23}	0.1

First scenario (uncertainty-free):

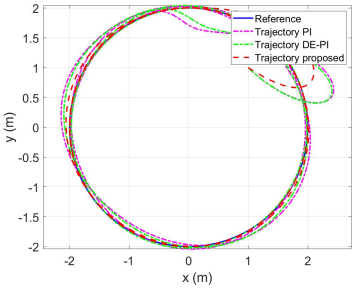

Figure 5: Robot trajectory in the fixed coordinate system.

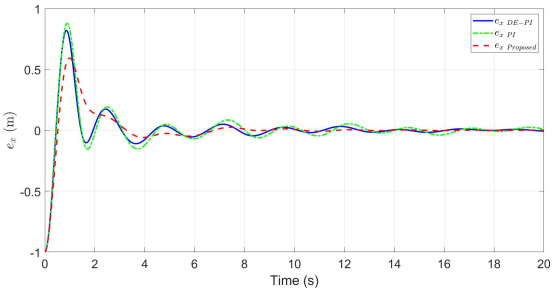

Figure 6: Tracking error along the x-axis.

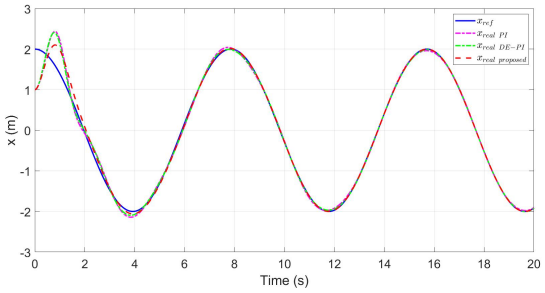

Figure 7: Trajectory along the x-axis.

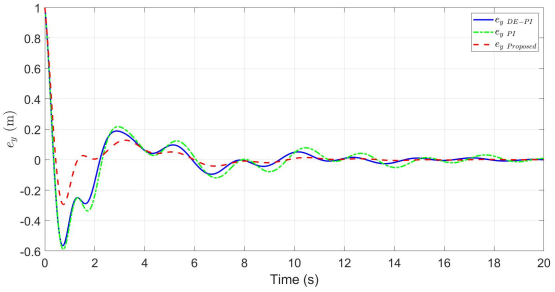

Figure 8: Tracking error along the y-axis.

Table 5: The RMSE values

Parameters	Uncertainty-free		
	PI	DE-PI	Proposed
$e_{x\text{RMSE}}$	0.2546	0.2377	0.2200
$e_{y\text{RMSE}}$	0.2138	0.2002	0.1466
$e_{\psi\text{RMSE}}$	0.3267	0.2823	0.2573
$\tau_{1\text{RMSE}}$	2.3550	2.3015	1.8582
$\tau_{2\text{RMSE}}$	3.4916	3.5206	2.3687
$\tau_{3\text{RMSE}}$	2.5255	2.5452	1.7541

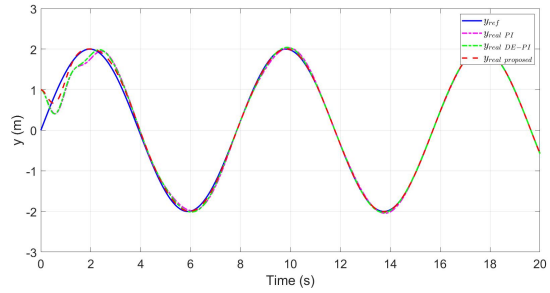

Figure 9: Trajectory along the y-axis.

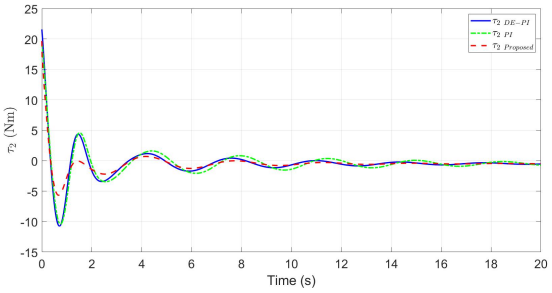

Figure 13: Control signal 2.

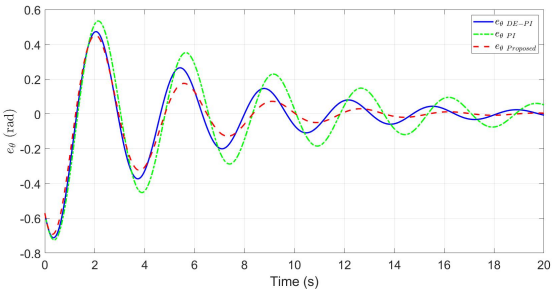

Figure 10: Tracking error along ψ.

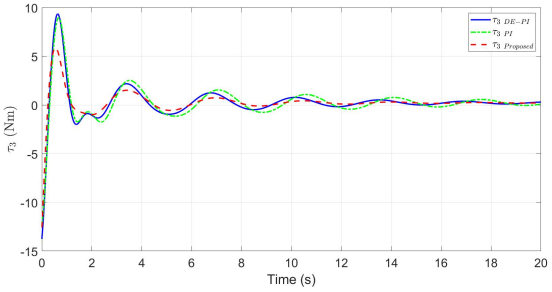

Figure 14: Control signal 3.

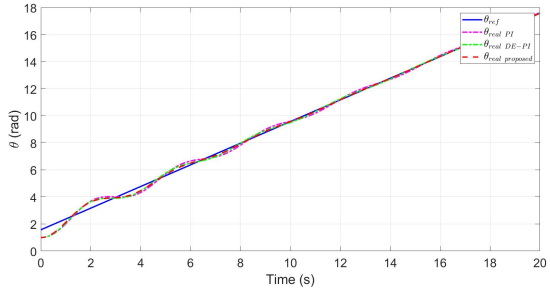

Figure 11: Trajectory along ψ.

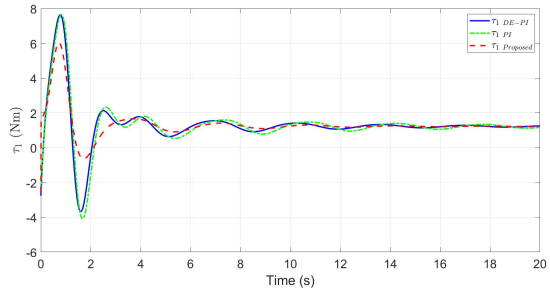

Figure 12: Control signal 1.

The simulation results presented in Figures (5–14) and Table (5) clearly highlight the superior performance of the proposed controller compared to the conventional PI controller and DE-PI controller. From the trajectory plots (Figure 5), the proposed controller enables the robot to follow the circular reference path more accurately, with only minor deviations during the initial phase, whereas the PI controller and DE-PI exhibit significant overshoot and drift before stabilizing, although the DE-PI shows improvement over the PI controller. Along the x- and y-axes (Figures 7 and 9), the proposed method maintains close adherence to the reference with smooth responses after $t \approx 5,\mathrm{s}$, while the PI and DE-PI responses show larger oscillations and slower convergence. The orientation trajectory (Figure 11) further confirms that the proposed controller achieves smoother tracking with less fluctuation compared to the PI and DE-PI methods.

A closer examination of the tracking errors (Figures 6, 8, and 10) shows that the PI and DE-PI controllers reach peak errors close to $1,\mathrm{m}$ on the x-axis, whereas the proposed controller limits this to approximately $0.6,\mathrm{m}$. Similarly, the maximum error on the y-axis decreases from about $0.6,\mathrm{m}$ to $0.3,\mathrm{m}$, and the orientation error exhibits fewer overshoots and faster damping. In terms of control effort (Figures 12–14), the PI and DE-PI controllers produce large initial torque peaks, especially τ_1, which reaches around $8,\mathrm{Nm}$, while the proposed controller keeps all torques below approximately $6,\mathrm{Nm}$, resulting in smoother and more stable actuation.

Table (5) quantitatively supports these observations: the RMSE values of position errors are reduced from 0.2546 to 0.2377 (DE-PI) and 0.2200 (DE-LQR) for e_x, from 0.2138 to 0.2002 (DE-PI) and 0.1466 (DE-LQR) for e_y, and from 0.3267 to 0.2823 (DE-PI) and 0.2573 (DE-LQR) for e_ψ, indicating improvements of approximately 14–31%. For the DE-LQR results, the torque RMSE values also show significant reductions: $\tau_{1\mathrm{RMS}}$ decreases from 2.3550 to 1.8582, $\tau_{2\mathrm{RMS}}$ from 3.4916 to 2.3687, and $\tau_{3\mathrm{RMS}}$ from 2.5255 to 1.7541,

corresponding to reductions of about 25–35%. Meanwhile, although the tracking errors are reduced, the RMSE values of τ_2 and τ_3 slightly increase from 3.4916 to 3.5206 and from 2.5254 to 2.5452, respectively, when using DE-PI, reflecting a trade-off between performance and control effort. These improvements stem from the optimally tuned gains obtained through the DE algorithm, which enhance the transient response, suppress overshoot, and reduce the coupling between translational and rotational motions. As a result, the robot not only tracks the desired trajectory more accurately but also achieves smoother control with lower energy consumption.

Second scenario (under uncertainties): In this case, the friction coefficient varies with time and is defined as $b_f = \overline{b}_f + 0.04\sin(\pi t)\ (kg, m^2/s)$, where $\overline{b}_f = 0.1$ denotes the nominal value used in the design of the proposed method.

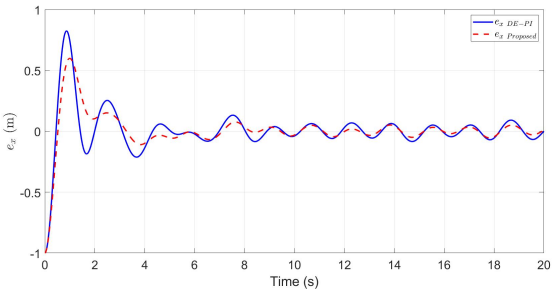

Figure 15: Tracking error along the x-axis.

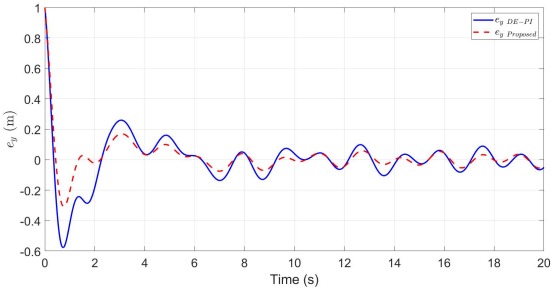

Figure 16: Tracking error along the y-axis.

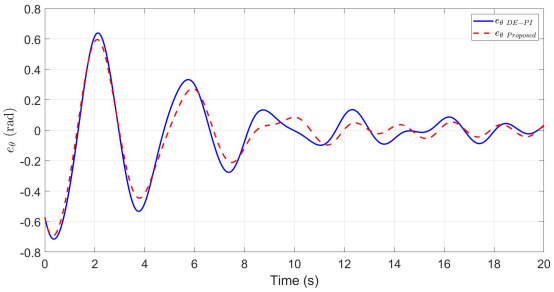

Figure 17: Tracking error along the ψ-axis.

As shown in Figures (15, 16, and 17), the tracking performance obtained using the DE–LQR controller is significantly improved, even under time-varying friction. In particular, notable improvements are observed in terms of overshoot, settling time, and post-settling behavior, indicating that the adverse effects of uncertainties are effectively reduced compared with the DE–PI controller. Furthermore,

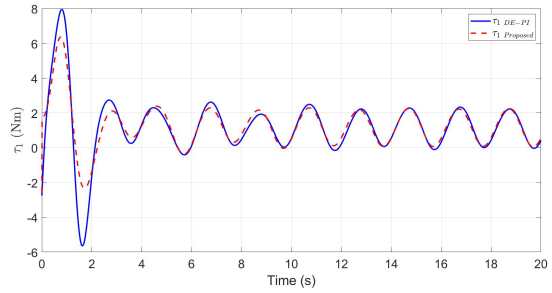

Figure 18: Control signal 1.

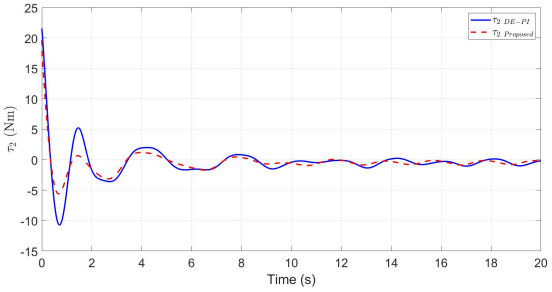

Figure 19: Control signal 2.

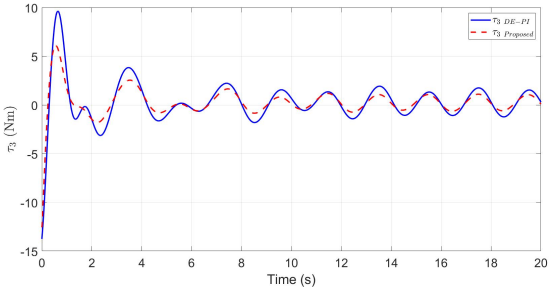

Figure 20: Control signal 3.

Table 6: The RMSE values

Parameters	Under uncertainties	
	DE-PI	Proposed
$e_{x\text{RMSE}}$	0.2503	0.2201
$e_{y\text{RMSE}}$	0.2142	0.1535
$e_{\psi\text{RMSE}}$	0.3391	0.3071
$\tau_{1\text{RMSE}}$	2.5896	2.0727
$\tau_{2\text{RMSE}}$	3.6730	2.4735
$\tau_{3\text{RMSE}}$	2.8696	1.9539

as illustrated in Figures (18, 19, and 20), the proposed method achieves superior performance with lower control effort than the DE–PI controller. The RMSE indices corresponding to the signals in Figures (15–20) are provided in Table (6). These results demonstrate that the DE–LQR approach significantly enhances both control performance and robustness compared with the conventional PI controller.

5. Conclusion

This study integrated a DE-LQR hybrid control strategy to improve the trajectory tracking performance and robustness of OMRs. The proposed control framework exploits the

optimal state-feedback structure of the LQR controller while overcoming its main limitation the manual tuning of weighting matrices by employing the global optimization capability of the DE algorithm. By automatically identifying the most effective Q and R matrices, the DE-LQR controller achieves smoother control actions, faster transient responses, and significantly reduced tracking errors compared to conventional hand-tuned LQR designs. Simulation results further confirm improved stability margins, demonstrating the effectiveness of the proposed approach for practical OMRs application. Future research may extend this work by validating the proposed controller on real hardware platforms, integrating disturbance observers to enhance rejection of external forces, exploring adaptive or learning-based extensions of the DE-LQR framework, and investigating decentralized implementations suitable for fault tolerant multi-wheel actuation systems.

References

[1] Nguyen, V. K., Thi, H. L., Cao, D. T., & Nguyen, T. L. (2024). *Control strategy for liquid transfer using a four-wheel Mecanum mobile robot platform.* IEEE Conference on Applying New Technologies. https://doi.org/10.1109/ATiGB63471.2024.10717655

[2] Shahgholian, S., Akhavan, M., & Kamrani, V. (2024). *Design an omnidirectional autonomous mobile robot based on non-linear optimal control to track a specified path.* IET Control Theory & Applications, 18(5), 982–995. https://doi.org/10.1049/cth2.12656.

[3] Al Mamun, M. A., Nasir, M. T., & Khayyat, A. (2018). *Embedded system for motion control of an omnidirectional mobile robot.* IEEE Access, 6, 58691–58703. https://doi.org/10.1109/ACCESS.2018.2873664.

[4] Wang, D., Wei, W., Yeboah, Y., Li, Y., & Gao, Y. (2020). *A robust model predictive control strategy for trajectory tracking of omni-directional mobile robots.* Journal of Intelligent & Robotic Systems, 98(2), 231–245. https://doi.org/10.1007/s10846-019-01083-1.

[5] Villarreal-Cervantes, M. G. (2012). *Kinematic dexterity maximization of an omnidirectional wheeled mobile robot: A comparison of metaheuristic and SQP algorithms.* International Journal of Advanced Robotic Systems, 9(6), 237. https://doi.org/10.5772/52251.

[6] Guo, T. (2022). *Trajectory tracking control of the Mecanum wheeled mobile robot based on SMC methods.* Proc. SPIE 12081. https://doi.org/10.1117/12.2619487.

[7] Shehata, O. R. (2024). *Development of an intelligent reinforcement learning-based controller for omni-wheel mobile robot.* ResearchGate. https://doi.org/10.13140/RG.2.2.22248.65287

[8] Tuan, P. A., & Linh, N. M. (2024). *Trajectory tracking control of omnidirectional mobile robots: A model-free control-based approach.* Journal of Applied Science and Engineering, 27(12), 1843-1852. https://doi.org/10.6180/jase.202412_27(12).0010.

[9] Tsai, C. C., Wu, H. L., & Tai, F. C. (2016). *Adaptive backstepping decentralized formation control using fuzzy wavelet neural networks for uncertain Mecanum-wheeled omnidirectional multi-vehicles.* IEEE ICIT, 164–171. https://doi.org/10.1109/ICIT.2016.7474914.

[10] Vlantis, P., Bechlioulis, C. P., Karras, G., & Kyriakopoulos, K. J. (2016). *Fault-tolerant control for omni-directional mobile platforms with four Mecanum wheels.* IEEE Transactions on Robotics, 32(6), 1465–1476. https://doi.org/10.1109/TRO.2016.2593428.

[11] Mellah, S., Graton, G., El Adel, E. M., & Ouladsine, M. (2021). *Health state monitoring of Mecanum-wheeled mobile robot actuators and its impact on robot behavior analysis.* Journal of Intelligent & Robotic Systems, 103(3), 56. https://doi.org/10.1007/s10846-021-01360-7.

[12] Zhang, T., & Zhang, X. (2023). *Distributed model predictive control with particle swarm optimizer for collision-free trajectory tracking of multi-wheeled mobile robot formations.* Actuators, 12(3), 127. https://doi.org/10.3390/act12030127.

[13] Xing, H., Xu, Y., Ding, L., Chen, J., & Gao, H. (2025). *Trajectory tracking control of wheeled mobile manipulators with joint flexibility via virtual decomposition approach.* IEEE Transactions on Systems, Man, and Cybernetics, 55(1), 212–226. https://doi.org/10.1109/TSMC.2024.3357768.

[14] Serrano-Pérez, O., & Villarreal-Cervantes, M. G. (2020). *Meta-heuristic algorithms for control tuning of omnidirectional mobile robots.* Engineering Optimization, 52(6), 1013–1034. https://doi.org/10.1080/0305215X.2019.1585834.

[15] Paredes-Ballesteros, J. A., & Villarreal-Cervantes, M. G. (2025). *Optimal tuning of robot–environment interaction controllers via differential evolution: A case study on (3,0) mobile robots.* Mathematics, 13(11), 1789. https://doi.org/10.3390/math13111789.

[16] Rojas-López, A. G., & Villarreal-Cervantes, M. G. (2025). *Online controller tuning for omnidirectional mobile robots using a multivariate-multitarget polynomial prediction model and evolutionary optimization.* Biomimetics, 10(2), 114. https://doi.org/10.3390/biomimetics10020114.

[17] Miah, M. S., & Gueaieb, W. (2015). *RFID-based mobile robot trajectory tracking and point stabilization through on-line neighboring optimal control.* Journal of Intelligent & Robotic Systems, 80(3–4), 467–485. https://doi.org/10.1007/s10846-014-0048-3.

[18] Sira-Ramírez, H., & López-Uribe, C. (2010). *Real-time linear control of the omnidirectional mobile robot.* IEEE CDC, 3427–3432. https://doi.org/10.1109/CDC.2010.5717804.

[19] Luy, N. T. (2017). *Robust adaptive dynamic programming-based online tracking control for real wheeled mobile robots.* Transactions of the Institute of Measurement and Control, 39(5), 681–692. https://doi.org/10.1177/0142331215620267.

[20] Abdelwahab, M., Parque, V., & El Bab, A. M. R. F. (2022). *A study on the optimum design of fuzzy logic control using differential evolution algorithms for omnidirectional mobile robot navigation.* Research Square Preprint. https://doi.org/10.21203/rs.3.rs-1780996/v1.

[21] Lopez-Franco, C. (2018). *Inverse kinematics of mobile manipulators based on differential evolution.* Advanced Robotics, 32(8), 432–447. https://doi.org/10.1177/1729881417752738.

[22] Tan, M., Chai, B., & Zhang, K. (2025). *Optimized trajectory tracking control with disturbance resistance for wheeled mobile robots.* Transactions of the Institute of Measurement and Control, 47(3), 213–226. https://doi.org/10.1177/01423312241283522.

[23] Paredes-Ballesteros, J. A., & Villarreal-Cervantes, M. G. (2025). *Differential evolution-based controller tuning for omnidirectional mobile robots.* Mathematics, 13(11), 1789. https://doi.org/10.3390/math13111789.

[24] Shahgholian, S., Akhavan, M., & Kamrani, V. (2024). *Non-linear optimal path control for omnidirectional robots.* IET Control Theory & Applications, 18(5), 982–995. https://doi.org/10.1049/cth2.12656.

[25] Wang, C., Liu, X., Yang, X., Hu, F., Jiang, A., & Yang, C. (2018). *Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy.* Applied Sciences, 8(2), 231. https://doi.org/10.3390/app8020231.

[26] Paredes-Ballesteros, J. A., & Villarreal-Cervantes, M. G. (2025). *Optimal tuning of robot–environment interaction controllers via differential evolution.* Mathematics, 13(11), 1789. https://doi.org/10.3390/math13111789.

[27] Storn, Rainer & Price, Kennet. (1997). *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces.* Journal of global optimization, 11(4),342-359. https://doi.org/10.1023/A:1008202821328.