sciendo

B&H Electrical Engineering, Volume 19, Issue 1, 2025:36-44
ISSN:2566-3143, eISSN:2566-3151, DOI: 10.2478/bhee-2025-0006

Original scientific paper

Submitted: March 18, 2025

Accepted: Mazy 11, 2025

# METAHEURISTIC OPTIMIZATION METHODS FOR OPTIMAL SUPERTWISTING SLIDING MODE CONTROLLER

Mirza Hodžić[1,2], Naser Prljača[1], Tatjana Konjić[1]

**Abstract:** Designing optimal controllers aims to improve the performance of a control system by minimizing tracking errors,control effort, or other system-specific objectives. However, obtaining analytical solutions for optimal controllers is often intractable for highly nonlinear, coupled multi-input multi-output (MIMO) systems. While gradient-based optimization methods can be employed, they may converge to suboptimal solutions due to the presence of multiple local minima in the cost function. Metaheuristic algorithms, on the other hand, offer a way to search for global optima even in complex, nonlinear systems. This article considers a nonlinear, coupled 2-DOF SCARA robot manipulator and proposes a super-twisting sliding mode controller that generates smooth control inputs compared to conventional sliding mode controllers, which typically produce discontinuous signals. We explore the use of genetic algorithms, particle swarm optimization, and simulated annealing to automatically tune the controller parameters, with the objective of minimizing trajectory and velocity tracking errors along a predefined path.

**Keywords:** Supertwisting sliding mode control, genetic algorithm, particle swarm optimization, simulated annealing, metaheuristic optimization, optimal control

## INTRODUCTION

Optimal control problems involve systems that aim to achieve tracking or stabilization of state variables while optimizing a specific optimality criterion. Controllers that achieve this are referred to as optimal controllers [1], [2]. Controllers can be designed with a number of parameters, which further complicates finding the optimal controller parameters analytically. Furthermore, the model and controller can be nonlinear systems, which further constrains the solution to the optimal control problem. For simpler or linear controllers and models, a classical gradient descent–based method is sufficient to obtain the optimal controller. In the case of nonlinear controllers or controllers with too many design parameters, a gradient descent–based optimization method can yield a false optimal controller due to becoming stuck in a local minimum.

To solve this problem, metaheuristic optimization methods can be used. While manual trial-and-error tuning might work in some cases, it is often inefficient, non-systematic, and may not consistently yield high-performance solutions. Metaheuristic optimization methods, on the other hand, provide an automated and reproducible approach to tuning controller parameters.

They can efficiently explore high-dimensional or non-convex search spaces, reduce human effort, and identify parameter sets that optimize performance according to a defined objective function. Metaheuristic optimization methods are experience-based algorithms that aim to find near-optimal or satisfactory solutions, especially in complex or non-convex search spaces where traditional optimization methods may fail [3].

The controller obtained through metaheuristic optimization is considered optimal in a practical sense, as it achieves excellent performance in the defined objective space, despite the absence of global optimality guarantees. Metaheuristic methods can be divided into three categories: evolutionary, physics-based, and swarm intelligence–based optimization algorithms. Each of these has its own advantages and disadvantages [4], [5].

Similar works use metaheuristic algorithms for linear controller tuning [6]–[8]. This article aims to compare these metaheuristic methods in the design of a sliding mode controller for a 2DOF SCARA manipulator [9]. This article is organized as follows: Section I provides the introduction. Section II explains the genetic algorithm, particle swarm optimization, and simulated annealing metaheuristic optimization methods. Section III presents

[1] University of Tuzla, Faculty of electrical engineering Tuzla, Bosnia and Herzegovina
[4] Correspondence email: mirza.hodzic@fet.ba

the SCARA 2DOF dynamic robot model. Section IV explains conventional sliding mode control. Section V presents a supertwisting sliding mode control for a 2DOF SCARA manipulator. Section VI shows the results and compares the three above-mentioned optimization algorithms. Section VII provides the final conclusion.

## 1. METAHEURISTIC OPTIMIZATION METHODS

### 1.1. Genetic algorithm

The genetic algorithm is a population-based metaheuristic optimization algorithm that imitates natural evolution, during which a population evolves by filtering out the fittest members. Articles [10] and [11] provide a good overview of the genetic algorithm and its parameters. The genetic algorithm is based on natural evolutionary principles: selection, mutation, and crossover. Figure 1 shows the flow diagram of a genetic algorithm. First, an initial population is proposed via an initial solution. For each member of the current population, a fitness value is calculated. This fitness value represents how well a particular solution performs with respect to the optimization objective, such as minimizing tracking error or control effort. The higher the fitness value of a population member, the more likely it is to be selected for the process of crossover. Population members chosen for reproduction are selected through a process called selection. Members are chosen based on their fitness values in such a way that those with higher fitness values have a higher chance of survival and reproduction. Selection can be carried out in several ways. One method is to transfer good genetic material to a new generation, a process referred to as generational selection. Another method is to eliminate population members with low fitness values, known as selection by elimination. Crossover refers to the exchange of genetic material during reproduction.

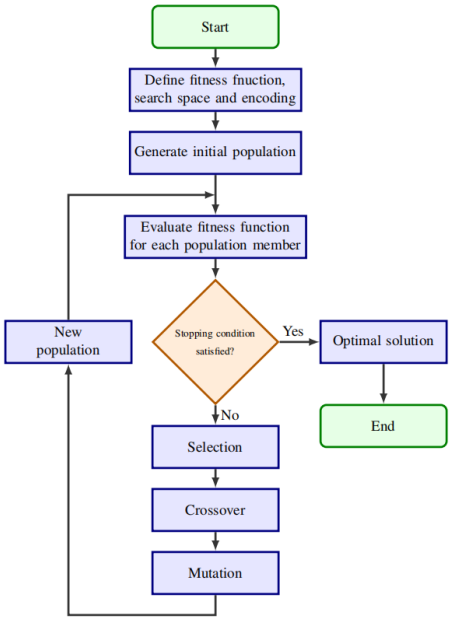


Figure 1: Genetic algorithm flowchart

### 1.2. Particle swarm optimization

The Particle Swarm Optimization (PSO) algorithm is a swarm intelligence optimization method that exhibits behavior characteristic of large groups of individual members communicating and acting together [12], [13]. Population members gather better information about their location within the swarm through interaction. By combining interactions with other members and their own previous knowledge, each member improves its position within the swarm [14]. Particle Swarm Optimization is the most general swarm intelligence optimization algorithm. The basic idea of the PSO algorithm was conceived by simulating a flock of birds, where the flock optimizes a specific goal function. PSO uses a specified number of members who ”fly” through the search space to find the best solution. During the flight, each member searches for its own best solution by comparing its current state with its previous states and the states of other members. Initially, a random swarm (population) is generated. During each algorithm step, the fitness value of each agent is calculated, and every agent keeps track of its own best fitness value. Throughout the optimization process, each agent tries to adjust its velocity based on its current position, current velocity, and the distance from the best agent. This velocity update is given by:

$$v_i(t+1) = wv_i(t) + C_1 r_1 \left(pbest_i(t) - x_i(t)\right) + C_2 r_2 \left(gbest_i(t) - x_i(t)\right) \quad (1)$$

where, $C_1$ and $C_2$ are positive constants called acceleration coefficients. Values $r_1$ and $r_2$ are random numbers in the interval [0, ,1], given by a uniform distribution, and w is the agent's inertia, which is used for balancing between local and global search. In 1, the term $C_1 r_1$ $(pbest_i(t) - x_i(t))$ is called the cognitive part of the PSO algorithm because the particle is updating its velocity based on its best solution and its current position. The term $C_2 r_2$ $(gbest_i(t) - x_i(t))$ is called the societal part of the PSO algorithm because the particle is updating its velocity based on societal knowledge adaptation. After the velocity update, the agent's current position is updated by the following equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Figure 2 shows the particle swarm optimization flowchart. The PSO algorithm stops if the maximum number of iterations is reached. The main disadvantage of the PSO algorithm is the inability to systematically choose algorithm parameters. They are usually chosen as standard values from existing literature or determined by trial and error. Analogous to the genetic algorithm, the fitness of each particle in PSO corresponds to the value of the cost function in the control system. The particles in the swarm move toward solutions that minimize this cost function, guiding the optimization process.

B&H Electrical Engineering, Volume 19, Issue 1, 2025:36-44
ISSN:2566-3143, eISSN:2566-3151, DOI: 10.2478/bhee-2025-0006

### 1.3. Simulated annealing

Simulated annealing is a stochastic optimization method that is analogous to the process of atomic substance rearrangement within the crystal structure during the cooling of a substance. The basic idea of simulated annealing is to imitate the annealing of a melted metal in order to find the minimum of a function. In the simulated annealing method, an objective function is treated as the energy of the cooled metal, and artificial temperatures are introduced and gradually decreased in order to reach a global minimum. Initially, a random solution is generated and the energy is calculated for that solution. A new solution is then generated in the vicinity of the old solution. The energy difference between the old and new solutions is calculated as:

$$\Delta E = E(k+1) - E(k) \tag{3}$$

If the energy is reduced, then the new proposed solution is accepted. If the energy is not reduced, then the acceptance of the new solution is governed by the

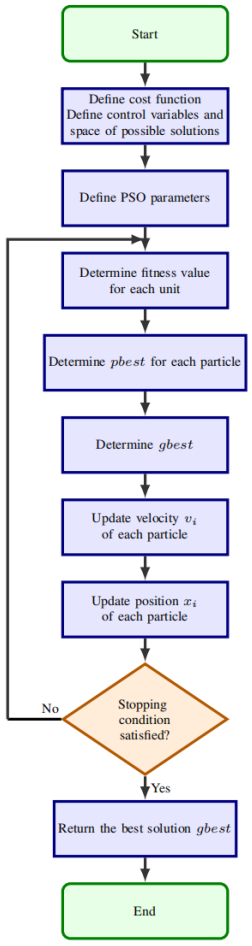


Figure 2: Particle swarm optimization flowchart

Boltzmann distribution, which depends on the current temperature. The Boltzmann distribution is given by:

$$P_r(E) = e^{-\frac{E}{k_B T_A}} \tag{4}$$

where $k_b$ is the Boltzmann constant and $T_A$ is the annealing temperature. The annealing temperatures are predefined and are gradually lowered according to the following equation:

$$T_A^{m+1} = \alpha T_A^m, \quad 0 < \alpha < 1 \tag{5}$$

where $\alpha$ is a design parameter chosen such that the temperature is very low near the convergence point. The previously described steps are repeated until the stopping criterion is satisfied. In this work, the cost function used in the optimal control formulation is treated as the energy in the simulated annealing algorithm. Thus, minimizing the energy corresponds directly to minimizing the control cost. Figure 3 shows the flowchart diagram of the simulated annealing optimization method.

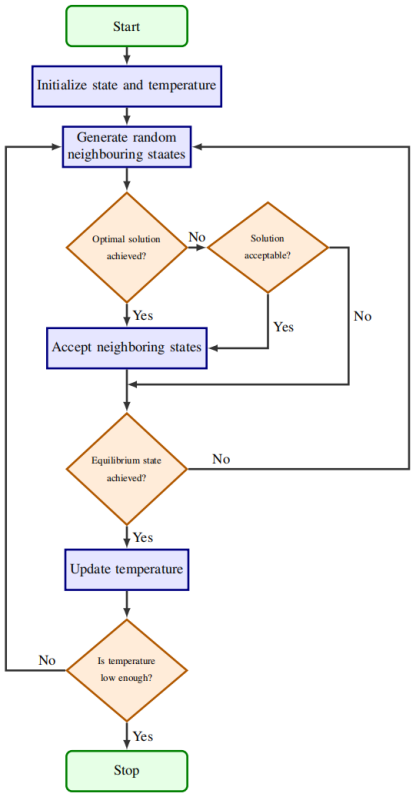


Figure 3: Simulated annealing flowchart

## 2. SCARA 2DOF MODEL

The mathematical model of a 2DOF SCARA manipulator is given by [15], [16]:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) = u(t) \tag{6}$$

where $B(q)$ is a 2 × 2 matrix, called the inertia tensor. Its diagonal terms are the moments of inertia of the robot joints, while the remaining terms are the inertia cross products. $C(q,\dot{q})$ is a 2 × 2 matrix, called the matrix of centrifugal and Coriolis forces acting on the robot joints. $F(\dot{q})$ is a 2×1 vector that models Coulomb friction, and

u represents the external moments acting on the robot joints. The dimensions of the vector u are 2 × 1. This model represents a planar 2-DOF SCARA robot [17], as shown in Figure 4. Since the robot operates in a horizontal plane, the gravitational forces do not influence its dynamics. Therefore, the gravity vector, which is typically present in manipulator models, is omitted. For the 2-DOF SCARA manipulator, these matrix values are given by [18], [19]:

$$B_{11}(q) = 1.7277 + 0.1908\cos q_2 \quad (7)$$

$$B_{12}(q) = 0.0918 + 0.0954\cos q_2 \quad (8)$$

$$B_{21}(q) = 0.334 + 03418\cos q_2 \quad (9)$$

$$B_{22}(q) = 0.9184 \quad (10)$$

$$C_{11}(q,\dot{q}) = 31.8192 - 0.0954\sin q_2 \quad (11)$$

$$C_{12}(q,\dot{q}) = -0.0954(\dot{q}_1 + \dot{q}_2)\sin q_2 \quad (12)$$

$$C_{21}(q,\dot{q}) = 0.344128\dot{q}_1 \sin q_2 \quad (13)$$

$$C_{22}(q,\dot{q}) = 12.5783 \quad (14)$$

$$f_1(\dot{q}) = 1.0256\,\mathrm{sgn}(\dot{q}_1) \quad (15)$$

$$f_2(\dot{q}) = 1.7842\,\mathrm{sgn}(\dot{q}_2) \quad (16)$$

For any input vector $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$, the joint accelerations can now be simulated as follows:

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}^{-1} \left( \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} - \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \right) \quad (17)$$

## 3. SLIDING MODE CONTROL

Sliding mode control [20], [21] is a subset of variable structure control [22], where the phase portrait of the dynamic system slides along a manifold that defines its transient behavior. Let the system dynamics be given by a nonlinear first-order matrix differential equation:

$$\dot{x}(t) = f(x) + g(x)u(t) \quad (18)$$

where $u = \begin{bmatrix} q & \dot{q} \end{bmatrix}^T$, $u \in \mathbb{R}^2$ while $f(x)$ and $g(x) \geq g_0 \geq 0$ are smooth functions defined by:

$$f(x) = \begin{bmatrix} \dot{q} \\ -B^{-1}(q)\left(C(q,\dot{q})\dot{q} + F(\dot{q})\right) \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 & B^{-1}(q) \end{bmatrix}^T$$

The idea of sliding mode control is to force the system, given by 18, to be constrained to the manifold given by:

$$s(x,t) = 0 \quad (19)$$

This function defines the sliding manifold along which the phase portrait will reach the equilibrium point. Let us denote the Lyapunov candidate function as follows:

$$V(x,t) = s^T s \quad (20)$$

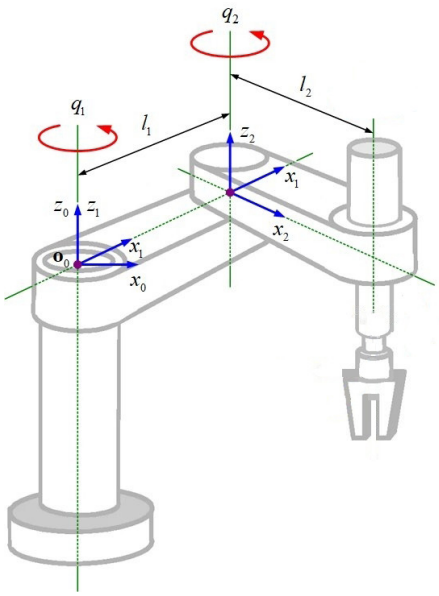


Figure 4: Sketch of the 2DoF SCARA robot

A global reaching condition is given by:

$$\dot{V}(x,t) < 0, \quad \forall s \in \mathbb{R} \setminus \{0\} \quad (21)$$

Therefore, asymptotic stability is guaranteed if the control input $u$ is such that 21 is satisfied. Furthermore, finite reaching time can be obtained if:

$$\dot{V}(x,t) < -\epsilon, \quad \forall s \in \mathbb{R} \setminus \{0\} \wedge \epsilon > 0 \quad (22)$$

One method to always guarantee finite reaching time and global asymptotic stability is to choose the control input such that the derivative of the sliding function always has the opposite sign compared to the current value of the sliding surface:

$$\dot{s} = -\xi sgn(s) \quad (23)$$

Now it follows that:

$$\dot{V} = -s\xi sgn(s) = -\xi |s| = -\xi\sqrt{V} \quad (24)$$

This implies:

$$V^{1/2}(t) = V^{1/2}(0) - \xi t \quad (25)$$

and therefore, at time $t_r = V^{1/2}(0)/\xi$, the phase portrait reaches the manifold in finite time.

## 4. SLIDING MODE CONTROL FOR SCARA MANIPULATOR

For SCARA manipulator, let us define the sliding surface as:

$$s = \beta e + \dot{e} \quad (26)$$

where $e$ is the tracking error defined as:

$$e = \begin{bmatrix} q_{1d} \\ q_{2d} \end{bmatrix} - \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad (27)$$

B&H Electrical Engineering, Volume 19, Issue 1, 2025:36-44
ISSN:2566-3143, eISSN:2566-3151, DOI: 10.2478/bhee-2025-0006

where $q_{1d}$ and $q_{2d}$ are the desired trajectories for the manipulator joints, and the manifold slope and sliding mode constant are given by:

$$\beta = \begin{bmatrix} \beta_1 & 0 \\ 0 & \beta_2 \end{bmatrix} \qquad \xi = \begin{bmatrix} \xi_1 & 0 \\ 0 & \xi_2 \end{bmatrix} \tag{28}$$

It is now possible to obtain the finite time reaching law. Using 27 and 26 and plugging them into 23, it follows:

$$\dot{s} = \beta\dot{e} + \ddot{e} = \beta(\dot{q}_d - \dot{q}) + \ddot{q}_d - \ddot{q} = -\xi sng(s) \tag{29}$$

Using 17 and 29 it follows:

$$-\xi sgn(s) = \beta(\dot{q}_d - \dot{q}) + \left(\ddot{q}_d - B^{-1}(q)\left(u - C(q,\dot{q})\,\dot{q} + F(\dot{q})\right)\right) \tag{30}$$

Finally, the control effort u can be obtained:

$$u = B(q)\,(\xi sgn(s) + \beta(\dot{q}_d - \dot{q}) + \ddot{q}_d) + C(q,\dot{q})\,\dot{q} + F(\dot{q}) \tag{31}$$

It can be seen that the control input now contains both the continuous and the switching part, $B(q)\xi\mathrm{sgn}(s)$. This switching part can induce chattering [23], which is high-frequency oscillation of the control signal. Actuators have their own dynamics and frequency bandwidth, and they cannot produce an output signal of infinite frequency. Furthermore, high-frequency switching can cause dissipation and damage to the actuator circuit. One way to solve this problem is to approximate the switching part of the control law with a continuous approximation function, such as a saturation function or a sigmoid function [24]. However, these methods lack robustness and do not guarantee asymptotic stability. Moreover, this control law requires at least approximate knowledge of the model, as can be seen from 31, which can result in tracking errors. These problems can be solved by using smooth second-order sliding mode control [25]. The idea of higher-order sliding mode is to hide the discontinuity in its higher derivatives by ensuring that the measured output s and its derivatives are equal to zero, i.e.:

$$s(t) = \dot{s}(t) = \ldots = s^{r-1}(t) = 0 \tag{32}$$

where $r$ is the relative order of the sliding variable (output) $s$ relative to the input signal $u$. This yields a decentralized control law, called the supertwisting sliding mode control [26]:

$$u_1 = k_{11}\sqrt{|s|}sgn(s_1) + k_{12}\int sgn(s_1)dt \tag{33}$$

$$u_2 = k_{21}\sqrt{|s|}sgn(s_2) + k_{22}\int sgn(s_2)dt \tag{34}$$

This controller also guarantees finite reaching time [27]. The control produces a smooth signal since the gains $k_{11}\sqrt{|s_1|}$ and $k_{21}\sqrt{|s_2|}$ attenuate the chattering effect as the sliding variable becomes smaller

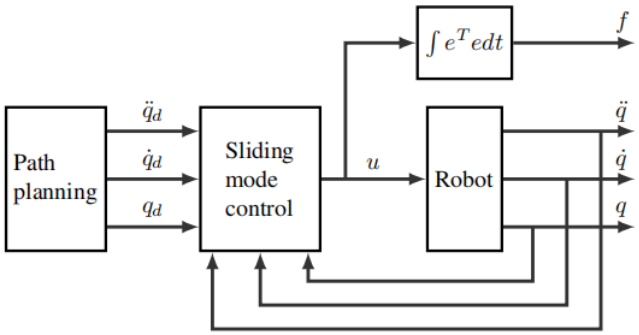


Figure 5: Sliding mode controller

## 5. RESULTS AND DISCUSSION

Controller parameters $\beta_1$, $\beta_2$, $k_{11}$, $k_{12}$, $k_{21}$, and $k_{22}$ now define the dynamics of the transient response. The desired trajectory is a trajectory that interpolates $[0\ \ 1]^T$ and $[1\ \ 0]^T$ via a third-order polynomial in an interval of $T$ = 2s. Initial and final joint velocities are zero. The desired trajectories are:

$$q_{1d} = -\frac{1}{4}t^3 + \frac{3}{4}t^2 \tag{35}$$

$$q_{2d} = \frac{1}{4}t^3 - \frac{3}{4}t^2 + 1 \tag{36}$$

Let us define the cost function that is to be minimized:

$$J(\beta,k) = \int_0^T \begin{bmatrix} e_1 & e_2 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} dt \tag{37}$$

where $I$ is the 2 × 2 identity matrix. This cost function equally minimizes joint position and joint velocity tracking errors. Figure 5 shows the optimization process. Values of the cost function are obtained through the simulation, and the optimization algorithm finds the controller gains that minimize the cost function. The simulation is conducted using the fourth-order explicit Runge-Kutta method. Initial joint positions are $q(0) = [0\ \ 1]^T$ and initial joint velocities are $\dot{q}(0) = [0\ \ 1]^T$ which means that the initial tracking error is zero as well. Controller parameters must be bounded so that high gains are avoided. If the gains are not bounded, the cost function will be minimized, but at the cost of a high-amplitude control signal. These controller gain bounds can be obtained by inspecting the effect of controller gains on system stability. For such nonlinear systems, these bounds are difficult to obtain. Heuristically, these gains can be bounded as:

$$0 \le \beta_i \le 10 \tag{38}$$

$$0 \le k_{ij} \le 100 \tag{39}$$

For $i, j$ = 1, 2, the lower bound on the controller gains is set to zero, as negative gains can lead to an unstable system, while the upper bound is set to 100 to avoid excessively high controller gains. All simulations and modeling are

done in the Simulink environment. Controller parameter values are shown in Table I. Table II shows the minimum value of the cost function obtained by each optimization method. It can be seen that all three methods obtain similar results.

Table I: Obtained controller parameters

| | $k_{11}$ | $k_{12}$ | $k_{21}$ | $k_{22}$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|---|---|---|
| **GA** | 16.7723 | 59.4364 | 2.0025 | 99.8918 | 3.0779 | 3.2430 |
| **PSO** | 16.8943 | 96.6785 | 4.4902 | 99.2437 | 6.3372 | 5.8519 |
| **SA** | 25.9752 | 97.9633 | 2.1898 | 83.3479 | 6.4891 | 4.0865 |

Particle swarm optimization obtains the smallest cost function values. Furthermore, the PSO algorithm obtains the optimal value in 21 iterations, whereas the other algorithms require more execution time.

Table II: Method comparasion

| | Minimum value $10^{-9}$ | Function count | Cycles |
|---|---|---|---|
| **GA** | 1.0802e | 7810 | 40 generations |
| **PSO** | 1.1246 | 1327 | 21 iterations |
| **SA** | 1.2166 | 3127 | 3000 iterations |

Since all three algorithms offer similar minimum cost function values, all three controllers achieve excellent tracking when the initial tracking error is nonexistent. Figure 6 shows the joint positions for all three controllers and the desired joint trajectories, while Figure 7 shows the desired and obtained joint velocities. It can be seen that all three controllers ensure near-perfect trajectory and velocity tracking. Figure 8 shows the control effort for both robot joints. It can be seen that the control signal does not contain chattering due to the supertwisting controller. Furthermore, the gains $k_{11}$ and $k_{21}$ for all three controllers are smaller compared to the gains $k_{12}$ and $k_{22}$, which gives better chattering attenuation. Initially, all three optimization algorithms give similar results for zero initial values. Let us now introduce some different initial values as follows: $q(0) = [-0.1 \;\; 0.9]^T$ and $\dot{q}(0) = [-0.5 \;\; -0.3]^T$

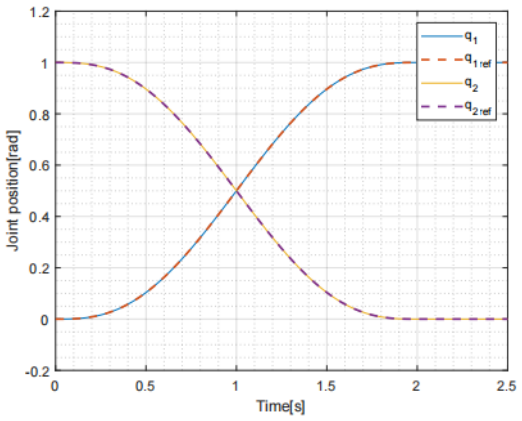


Figure 6: Joint position

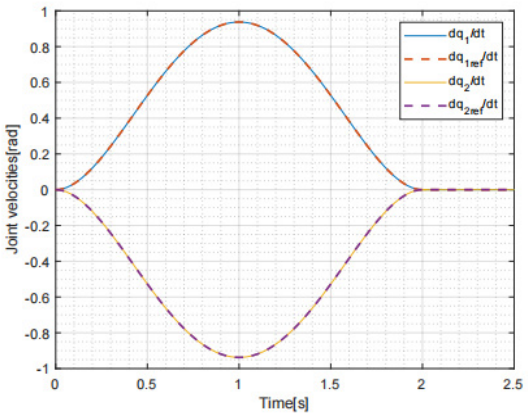


Figure 7: Joint velocities

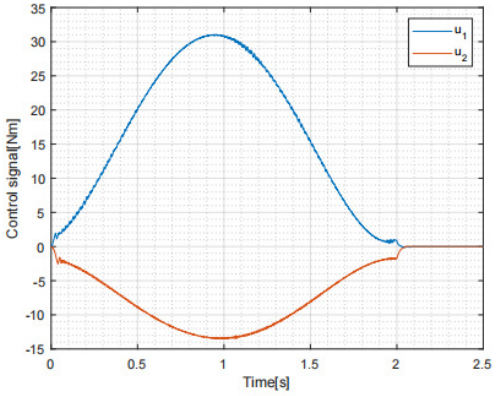


Figure 8: Control signal

and let us now define new desired trajectory given by:

$$q_{1d}(t) = \frac{\pi}{1.5^3}t^2 - \frac{2\pi}{3\,(1.5)^3}t^3 \tag{40}$$

$$q_{2d}(t) = \frac{\pi}{4} - \frac{3\pi}{4\,(1.5)^2}t^2 + \frac{\pi}{2\,(1.5)^3}t^3 \tag{41}$$

Furthermore, in order to show the controller's robustness to disturbances, let a constant input disturbance act at time $t = 1\text{s}$ with the intensity $u_d(t = 1\text{s}) = [10 \;\; -5]^T$ Nm. Figure 9 shows joint positions with nonzero position tracking error, a new desired joint trajectory, and the acting input disturbance, while Figure 10 shows joint velocities with nonzero velocity tracking error, a new desired joint velocity, and the acting input disturbance. It can be seen that all three algorithms ensure steady-state tracking. Figure 11 shows the control signal for all three optimization methods. It can be seen that all three optimization methods require the same amplitude of the input torque. It can also be observed that the existing control parameters do not produce a chattering effect. Figure 12 shows the sliding variable for controllers obtained via all three optimization methods.

B&H Electrical Engineering, Volume 19, Issue 1, 2025:36-44
ISSN:2566-3143, eISSN:2566-3151, DOI: 10.2478/bhee-2025-0006

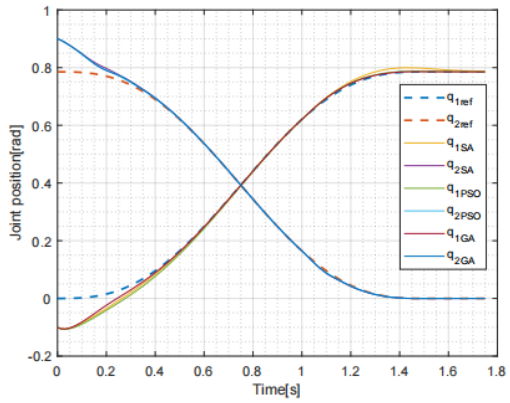


Figure 9: Joint positions for GA, PSO and SA algorithms with nonzero initial errors and acting input disturbance

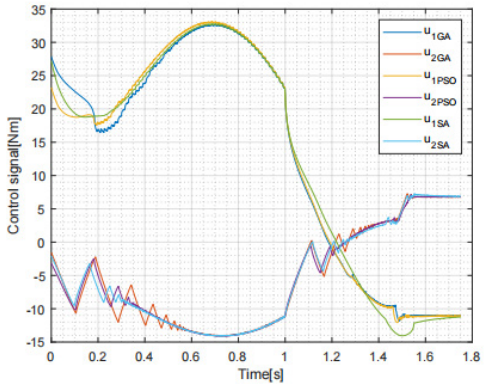


Figure 11: Control signal for GA, PSO and SA algorithms with nonzero initial errors and acting input disturbance

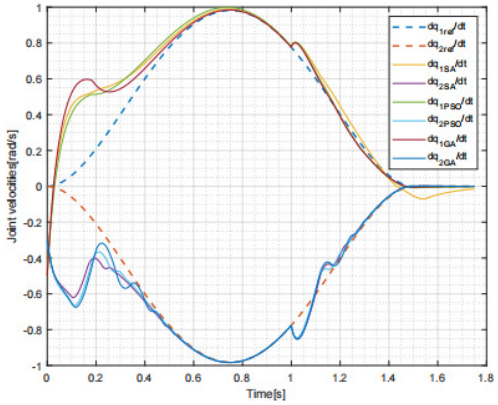


Figure 10: Joint velocities for GA, PSO and SA algorithms with nonzero initial errors and acting input disturbance

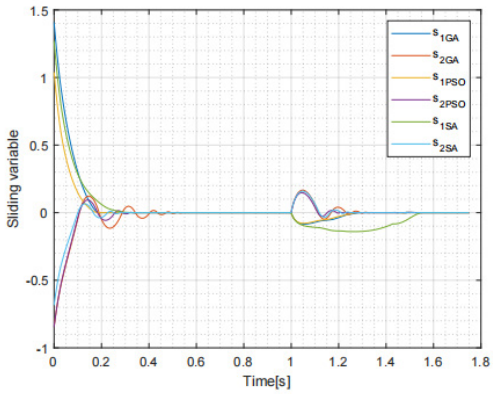


Figure 12: Sliding variable s for GA, PSO and SA algorithms with nonzero initial errors and acting input disturbance

It can be seen that all three controllers drive the sliding variables to zero, but the genetic algorithm introduces oscillations in the sliding variable response, which is an undesired behavior. Figure 13 shows the phase diagram of the robot manipulator with nonzero initial errors for all three optimization algorithms. It can be seen that all three optimization algorithms drive the system to the sliding switching lines and to the origin. The Table III shows the metrics for the obtained controllers with existing initial tracking errors, the new desired joint trajectory and the input disturbance.

Table III: Methods comparasion with nonzero initial tracking error and acting input disturbance

| Algorithm | Cost function | $e1_{RMS}$ | $e2_{RMS}$ | $\dot{e}1_{RMS}$ | $\dot{e}2_{RMS}$ |
|---|---|---|---|---|---|
| GA | 0.0025118 | 0.0025956 | 0.0021994 | 0.059166 | 0.087476 |
| PSO | 0.0029353 | 0.0029045 | 0.0021886 | 0.03865 | 0.086571 |
| SA | 0.0028377 | 0.002734 | 0.0022154 | 0.044351 | 0.076587 |

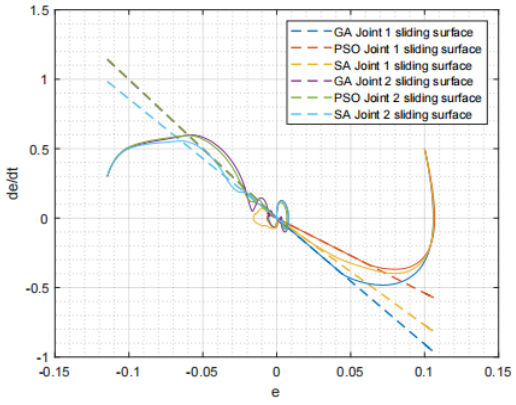


Figure 13: Phase portrait for GA, PSO and SA algorithms with nonzero initial errors and acting input disturbance

The chosen metrics are the root mean square of the joint position tracking error and the root mean square of the velocity tracking error for both joints:

$$e_{RMS} = \sqrt{\frac{1}{T}\int_0^T e(t)e^T(e)dt} \tag{42}$$

$$\dot{e}_{RMS} = \sqrt{\frac{1}{T}\int_0^T \dot{e}(t)\dot{e}^T(e)dt} \tag{43}$$

It can be seen that the genetic algorithm yields the smallest value of the cost function.

## 6. CONCLUSION

This article presented three metaheuristic optimization algorithms—genetic algorithm, particle swarm optimization, and simulated annealing—to determine the optimal supertwisting sliding mode controller for a 2DOF SCARA robot manipulator. Metaheuristic optimization algorithms are particularly useful for combinatorial problems or functions with high complexity. Unlike local optimizers, which may converge to local minima, metaheuristic algorithms are capable of finding global minima, making them well-suited for highly nonlinear systems and controllers. A nonlinear coupled 2DOF SCARA robot manipulator model was presented, and the supertwisting sliding mode controller was selected due to its ability to resolve the chattering problem commonly associated with conventional sliding mode control. It was demonstrated that all three optimization algorithms achieve excellent predefined trajectory and velocity tracking. Furthermore, the genetic algorithm was shown to deliver the best results when initial tracking errors were present.

## REFERENCES

[1] R. Stengel, Optimal Control and Estimation, ser. Dover books on advanced mathematics. Dover Publications, 1994.

[2] D. Kirk, Optimal Control Theory: An Introduction, ser. Dover Books on Electrical Engineering Series. Dover Publications, 2004.

[3] W. Wong and C. I. Ming, “A review on metaheuristic algorithms: Recent trends, benchmarking and applications,” in 2019 7th International Conference on Smart Computing & Communications (ICSCC), 2019.

[4] M. E. C¸ imen, Z. B. Garip, and A. F. Boz, “Comparison of metaheuristic optimization algorithms for numerical solutions of optimal control problems,” Concurrency and Computation: Practice and Experience, vol. 35, 2023.

[5] D. Garc´ıa, M. Herrera, and O. Camacho, “A comparative evaluation of metaheuristic optimization methods for control applications,” in 2023 IEEE Seventh Ecuador Technical Chapters Meeting (ECTM), 2023, pp. 1–6.

[6] M. H. Enad, R. F. Hassan, K. Mahmoud, A. A., and A. J. Humaidi, “2024,” Performance evaluation of a 2DOF-PID controller using metaheuristic optimization algorithms., vol. 57, no. 3, pp. 709–715, 2024.

[7] M. Ekole, O. Abdalla, I. Shalabi, and R. Shalaby, “Metaheuristic approaches to tune pid controller for ball on plate system,” pp. 121–135, 05 2024.

[8] M. Afsar and H. Arslan, “Optimizing pid gains of a vehicle using the state-of-the-art metaheuristic methods,” Academic Platform Journal of Engineering and Smart Systems, vol. 11, 07 2023.

[9] S. M. Mahil and A. Al-Durra, “Modeling analysis and simulation of 2-dof robotic manipulator,” 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4, 2016.

[10] V. Chahar, S. Katoch, and S. Chauhan, “A review on genetic algorithm: Past, present, and future,” Multimedia Tools and Applications, vol. 80, 02 2021.

[11] R. D. Anita Thengade, “Genetic algorithm - survey paper,” National Conference on Recent Trends in Computing, vol. NCRTC, no. 5, pp. 25–29, May 2012.

[12] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[13] A. Kaveh, Particle Swarm Optimization, 03 2014, pp. 9–40.

[14] “Front matter,” in Stochastic Global Optimization Methods and Applications to Chemical, Biochemical, Pharmaceutical and Environmental Processes, C. Venkateswarlu and S. E. Jujjavarapu, Eds. Elsevier, 2020.

[15] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control, 1st ed. Springer Publishing Company, Incorporated, 2008.

[16] P. I. Corke, Robotics, Vision & Control: Fundamental Algorithms in MATLAB, 2nd ed. Springer, 2017, iSBN 978-3-319-54413-7.

[17] J. E. LAV´IN-DELGADO, S. CHAVEZ-V ´ AZQUEZ, J. F. G ´ OMEZ-´AGUILAR, G. DELGADO-REYES, and M. A. RU´IZ-JAIMES, “Fractional-order passivity-based adaptive controller for a robot manipulator type scara,” Fractals, vol. 28, no. 08, p. 2040008, 2020. [Online]. Available: https://doi.org/10.1142/S0218348X20400083

[18] L. A. Soriano, J. D. J. Rubio, E. Orozco, D. A. Cordova, G. Ochoa, R. Balcazar, D. R. Cruz, J. A. Meda-Campana, A. Zacarias, and G. J. ˜Gutierrez, “Optimization of Sliding Mode Control to Save Energy in a SCARA Robot,” Mathematics, vol. 9, no. 24, p. 3160, Dec. 2021.

[19] F. G. Rossomando and C. M. Soria, “Adaptive neural sliding mode control in discrete time for a scara robot arm,” IEEE Latin America Transactions, vol. 14, no. 6, pp. 2556–2564, 2016.

[20] V. Utkin, “Variable structure systems with sliding modes,” IEEE Transactions on Automatic Control, vol. 22, no. 2, pp. 212–222, Apr. 1977.

[21] Utkin, “Sliding mode control design principles and applications to electric drives,” IEEE Transactions on Industrial Electronics, vol. 40, no. 1, pp. 23–36, 1993.

[22] J. Hung, W. Gao, and J. Hung, “Variable structure control: a survey,” IEEE Transactions on Industrial Electronics, vol. 40, no. 1, pp. 2–22, Feb. 1993.

B&H Electrical Engineering, Volume 19, Issue 1, 2025:36-44
ISSN:2566-3143, eISSN:2566-3151, DOI: 10.2478/bhee-2025-0006

[23] V. Utkin and H. Lee, "Chattering problem in sliding mode control systems," in International Workshop on Variable Structure Systems, 2006. VSS'06., 2006, pp. 346–350.
[24] C. Edwards and S. Spurgeon, Sliding Mode Control: Theory And Applications, ser. Series in Systems and Control. CRC Press, 1998.
[25] I. Shkolnikov, Y. Shtessel, and M. Brown, "A second-order smooth sliding mode control," in Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), vol. 3, 2001, pp. 2803–2808 vol.3.
[26] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, Sliding Mode Control and Observation, ser. Control Engineering. Springer New York, 2013.
[27] A. Polyakov and A. Poznyak, "Reaching time estimation for "supertwisting" second order sliding mode controller via lyapunov function designing," IEEE Transactions on Automatic Control, vol. 54, no. 8, pp. 1951–1955, 2009.

## BIOGRAOHY

**Mirza Hodžić** recieved his B.Sc. and M.Sc. degrees in 2020 and 2022 respectively from the Faculty of Electrical Engineering Tuzla, Bosnia and Herzegovina. He works in the industry and at the Faculty of Electrical Engineering at the Department of Control Systems, Robotics and Industrial Informatics. His research interests are in robotics, control systems, machine vision and missile guidance. Currently, he is working towards his PhD degree in the field of control systems at the same faculty.

**Naser Prljača** is full professor and head of department of Control Systems, Robotics and Industrial Informatics at the Faculty of Electrical Engineering, University of Tuzla. He holds BSc, MSc andPhD in Electrical and Electronics Engineering. He has been taking part in numerous research and industrial projects, and is author of a number of academic publications. His research and teaching interests include control systems, embedded systems, robotics, machine learning and computer vision.

**Tatjana Konjić** received her Ph.D. degree in Electrical Engineering from the Faculty of Electrical Engineering at the University of Tuzla, Bosnia and Herzegovina, in 2003. Since 1990, she has been affiliated with the Faculty of Electrical Engineering at the University of Tuzla, where she is currently a professor. She has led international projects and has served as a chairperson or member of scientific committees at international conferences, as well as a reviewer for international journals. Her research interests include the application of artificial intelligence to power systems, load forecasting, distributed generation, and smart grids. She is a member of IEEE, IEEE PES, IEEE CIS, International CIGRE, and BH K CIGRE.