Comparison of Optimization Approaches on Linear Quadratic Regulator Design for Trajectory Tracking of a Quadrotor

Baris Ata
Cukurova University
Mashar Gencal
masharcenkgencal@ardahan.edu.tr
Ardahan University

Research Article

Keywords: LQR, Quadrotor, Trajectory Tracking, Optimal Control, Optimization

Posted Date: December 20th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3762975/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Evolutionary Intelligence on April 4th, 2024.
See the published version at https://doi.org/10.1007/s12065-024-00928-5.

Comparison of optimization approaches on linear quadratic regulator design for trajectory tracking of a quadrotor

Baris Ata · **Mashar Cenk Gencal**

Received: date / Accepted: date

Abstract Linear Quadratic Regulator (LQR) is one of the most prevalent methods used in the control of unmanned aerial vehicles. LQR controllers are commonly employed in the control of both linear and non-linear systems due to their advantages such as easy-to-apply and high-performance structure. However, there is one main difficulty that plays a significant role in the manner of determining the gain for the control signal: choosing appropriate weighting matrices. The selection of these matrices that directly affect the controller performance is generally performed by trial and error, which is laborious and time-consuming. Accordingly, various optimization algorithms have been utilized to determine the weighting matrices of the LQR controller. In this paper, the weighting matrices of the designed LQR controller were obtained using Standard Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, and Grey Wolf Optimization algorithms. The obtained weighting matrices of the LQR controller were tested on an unmanned aerial vehicle simulation, and the performance of optimization algorithms were presented comparatively.

Keywords LQR · Quadrotor · Trajectory Tracking · Optimal Control · Optimization

1 Introduction

Unmanned Aerial Vehicles (UAVs) are aircraft with no crew on board and whose flight control is provided from the ground or automatically. Due to recent technological developments, the interest in UAVs in both civil aviation and the defense industry has increased; therefore, academic studies on UAVs in the literature are growing in parallel with this interest.

UAVs, which can be used for armed tasks as well as for functions such as search, rescue, research, and observation, are divided into two basic categories; rotary-wing and fixed-wing UAVs [30]. According to their weights and tasks, they can also be branched into different categories such as tactical, mini, micro, high-altitude, medium-altitude and combat UAVs [55]. In this study, an automatic flight controller is designed for a micro UAV with rotary wings, also called a quadrotor or quadcopter, with a weight of less than 2.5 kg.

Various control methods are proposed to control quadrotors. Among these methods, there are linear and nonlinear control methods such as Proportional Integral Derivative (PID) control [16,9,1], Sliding Mode Control [10,53], Backstepping control [26,6], and Linear Quadratic Regulator (LQR) control.

In the literature, one of the first papers that utilizes an LQR controller for quadrotor control is [14]. In this paper, a six degrees of freedom model of the quadrotor is utilized for trajectory optimization. In addition, a modified version of the LQR controller was obtained to control a quadrotor in [46] by adding an integral term to the LQR controller to minimize the steady-state error of the output. Moreover, in [36], an integral effect LQR controller was proposed for the trajectory tracking of a quadrotor, and a Kalman Filter is added to the system to minimize sensor errors.

On the other hand, in [43], to control a quadrotor, an LQR controller was modeled and the state responses of the system to the changes in the weighting matrices of the LQR controller were investigated. In addition, the importance of selecting weighting matrices to improve performance criteria such as overshoot and steady-state error was analyzed. Also, a Linear Quadratic Gaussian controller that utilizes an

B. Ata
Cukurova University Computer Engineering Department, Adana, Turkey, ORCID 0000-0003-4773-0564

MC. Gencal
Ardahan University Computer Engineering Department, Ardahan, Turkey, ORCID 0000-0002-1317-3950
Tel.: +904782117575
E-mail: masharcenkgencal@ardahan.edu.tr

LQR controller was presented for the quadrotor, which can land on a moving platform, in a recent study [11].

As can be seen from previous studies in the literature, the LQR controller is frequently utilized in the control of UAVs. However, the most important and troublesome part of the LQR control method is the determination of the weighting matrices, which directly affect the control signal. These parameters are determined by the trial-and-error method in the classical approach. However, manually tuning the values of the matrices is laborious and time-consuming. For this reason, researchers are employing optimization algorithms more often to adjust the parameters of the weighting matrices in recent studies.

One of the recent studies that used an optimization algorithm to adjust the weighting matrices is [51]. In this paper, the Pigeon Inspired Optimization (PIO) algorithm was utilized to adjust the parameters of the LQR controller weighting matrices. Its results were compared with the results of the Particle Swarm Optimization (PSO) algorithm. Furthermore, in [4], the Genetic Algorithm (GA) and the PSO algorithm were employed to adjust the parameters of the weighting matrices of an LQR controller.

In [40] and [24], the weighting matrices of the LQR controller were determined by Artificial Neural Networks (ANN) and PSO; however, in both studies, the controllers were designed to control UAVs with three degrees of freedom. In [35], Multi-Objective High Exploration PSO was used to adjust the weighting matrices of the LQR controller for a nonlinear four degrees of freedom quadrotor model, while six degrees of freedom quadrotor models were used for the optimization of UAV trajectory tracking utilizing the PSO algorithm in [45,21].

Moreover, in [13], the classical inverted pendulum problem was extended to the inverted pendulum on a quadrotor, and Standard GA (SGA) and Improved GA methods were used to solve the main difficulty: determining the proper weighting matrices. In a recent study, the PSO algorithm was also operated to find the suitable LQR controller weighting matrices [34]. In that study, a quadrotor has an inverted pendulum which was controlled by the LQR controller. In [44], SGA, PSO, and Artificial Bee Colony (ABC) algorithms were also employed to tune the weighting matrices of the LQR controller design for position control of a double inverted pendulum system on a cart. On the other hand, while SGA was used to adjust the weighting matrices of the LQR controller in [28], Grey Wolf Optimization (GWO) algorithm was utilized for the same purpose in [48].

In this paper, an LQR controller, whose weighting matrices are determined by SGA, Differential Evolution (DE), PSO, and GWO algorithms, is designed to control six degrees of freedom quadrotor on tracking a predetermined three-dimensional trajectory in the simulation environment. In the literature, there is no study that used these optimization algorithms was found in which the weighting matrices of the LQR control method are optimized for trajectory tracking of a six degree of freedom quadrotor.

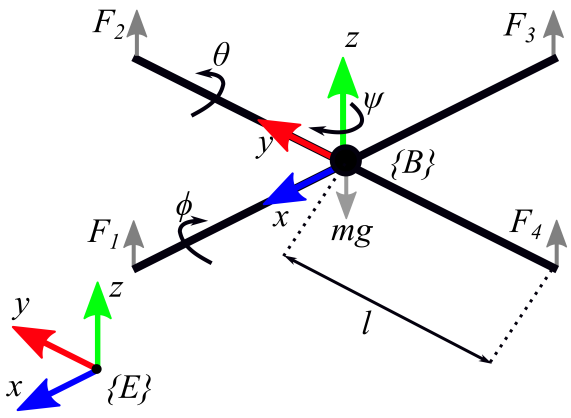


Fig. 1: Body diagram of the quadrotor with reference frames

The rest of the paper is organized as follows: Section 2 describes the dynamic system model of the quadrotor while Section 3 gives basic information about the LQR controller. In Section 4, the optimization algorithms are presented, and the obtained results from these algorithms are shown in Section 5. The paper is concluded with Section 6.

2 Dynamic Model of Quadrotor

In this section, the dynamic model of the quadrotor is introduced based on the Newton-Euler formalism. The non-linear dynamics of the quadrotor can be interpreted by defining the body frame, indicated by B, and the inertial or earth frame, indicated by E, as shown in Figure 1 where m, l are the mass and arm length of the quadrotor, respectively; g is the gravitational acceleration; F_i $(i = 1,...,4)$ is the thrust force generated by the ith rotor of the quadrotor. Additionally, the relationship between the control inputs U_i $(i = 1,...,4)$ and the thrust forces generated by the four rotors can be defined as [32,2]:

$$\begin{aligned} U_1 &= F_1 + F_2 + F_3 + F_4 \\ U_2 &= F_2 - F_4 \\ U_3 &= F_3 - F_1 \\ U_4 &= -F_1 + F_2 - F_3 + F_4. \end{aligned} \tag{1}$$

Furthermore, it is assumed that (I) the structure of the quadrotor is symmetric and rigid, and (II) the origin of the coordinate system of the earth frame is fixed to the initial position of the quadrotor, which is a specific point in the three-dimensional space [56]. To take these into consideration, the orientation and position dynamics of the quadrotor can be derived in a three-dimensional space [41].

Considering the reference frames E and B, the six degrees of freedom of the quadrotor can be defined as $[x,y,z,\phi,\theta,\psi]$ where $[x,y,z]$ indicates the position vector and $[\phi,\theta,\psi]$ denotes the orientation vector.

The location of the quadrotor to the earth frame in three-dimensional space is defined by the position vector, and the angular position of the quadrotor is defined by the orientation vector. ϕ, θ and ψ in the orientation vector refer to the roll angle rotating around the x axis, the pitch angle rotating around the y axis, and the yaw angle rotating around the z axis, respectively. The rotational matrices R_ϕ, R_θ, and R_ψ around each of the three axes from frame B to frame E can be defined as follows [32]:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix}, \tag{2}$$

$$R_\theta = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix}, \tag{3}$$

$$R_\psi = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where $c(.)$ and $s(.)$ denote $cos(.)$ and $sin(.)$, respectively.

The transition matrix R_{BE} from the frame B to the E may be obtained by multiplying the rotational matrices R_ϕ, R_θ, and R_ψ as follows [41,31]:

$$\begin{aligned} R_{BE} &= R_\phi R_\theta R_\psi \\ &= \begin{bmatrix} c(\psi)c(\phi) & c(\psi)s(\theta)s(\phi)-s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi)+s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi)+c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi)-s(\phi)c(\psi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \end{aligned} \tag{5}$$

The translational dynamics of the quadrotor can be derived by employing the transition matrix R_{BE} introduced in Equation 5. To obtain translational equations of motion, the total thrust force generated by the rotors in the system B can be considered as;

$$F_T^B = \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}. \tag{6}$$

Using the transition matrix R_{BE} in Equation 5, the total thrust force in the body frame F_T^B can be transformed into the earth frame E as;

$$F_T^E = R_{BE} F_T^B. \tag{7}$$

Furthermore, the gravitational force F_G^E in the earth frame E can be described as follows:

$$F_G^E = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{8}$$

Consequently, according to Newton's second law, the total forces applied to the system in the vertical direction can be considered as [18];

$$F_T^E + F_G^E = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}. \tag{9}$$

Substituting Equations 7 and 8 into Equation 9 yields [47]:

$$R_{BE} \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}. \tag{10}$$

By rewriting the Equation 10, one can get the translational motion equations of the quadrotor as;

$$\begin{aligned} \ddot{x} &= \frac{1}{m}\left(s(\theta)c(\phi)c(\psi)+s(\phi)s(\psi)\right)U_1 \\ \ddot{y} &= \frac{1}{m}\left(s(\theta)c(\phi)s(\psi)-s(\phi)c(\psi)\right)U_1 \\ \ddot{z} &= \frac{1}{m}\left(c(\theta)c(\phi)\right)U_1 - g. \end{aligned} \tag{11}$$

On the other hand, the rotational motion equations of the quadrotor can be derived according to Newton's second law, likewise the translational motion equations. Considering the angular velocity as $\omega = [p,q,r]^T$ where p, q and r are the angular velocities of roll, pitch, and yaw with respect to the body frame; the relationship between angular velocity and Euler angles can be defined as follows [41]:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{12}$$

Thus, the total torques M_B applied to the quadrotor in the body frame system can be defined according to the Euler formulation as follows [54]:

$$M_B = J\dot{\omega} + \omega \times J\omega \tag{13}$$

where J is the inertia matrix. Furthermore, the positive symmetric constant inertia matrix J in Equation 13 can be written as;

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{14}$$

where J_x, J_y, and J_z are the rotary inertia with respect to the x, y and z axes, respectively.

Solving Equation 13 by substituting the inertia matrix J in Equation 14 into it yields [23]:

$$M_B = \begin{bmatrix} J_x\dot{p} + (J_z - J_y)\,qr \\ J_y\dot{q} + (J_x - J_z)\,pr \\ J_z\dot{r} + (J_y - J_x)\,pq \end{bmatrix}. \tag{15}$$

Consequently, considering the total torques M_B applied to the quadrotor in the earth frame E as;

$$M_B = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{16}$$

and substituting it into Equation 15, the complete rotational equations of motion become [18,47];

$$\begin{aligned} J_x\dot{p} &= (J_y - J_z)\,qr + U_2 \\ J_y\dot{q} &= (J_z - J_x)\,pr + U_3 \\ J_z\dot{r} &= (J_x - J_y)\,pq + U_4. \end{aligned} \tag{17}$$

The angular velocity $[p,q,r]$ in the body frame system B can be transformed to the angular velocity $[\dot{\phi},\dot{\theta},\dot{\psi}]$ in earth frame system E by the inverse of the transformation matrix introduced in Equation 12. This inverse transformation matrix can be expressed as [41]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{18}$$

where $t(.)$ denotes $tan(.)$.

Nevertheless, the attitude angles of the quadrotor are always kept small to ensure flight safety, thus the angles ϕ and θ can be considered as close to 0 around the hover position. Herewith, the transformation matrix introduced in Equation 18 converges to a unit matrix, and the relationship between the Euler angles in the system E and the angular velocity in the system B can be considered linear as follows [12]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \cong \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{19}$$

Thus, rewriting Equation 17 for angular acceleration using Equation 19, the rotational motion equations of the quadrotor can be acquired as:

$$\begin{aligned} \ddot{\phi} &= \left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} + \frac{1}{J_x}U_2 \\ \ddot{\theta} &= \left(\frac{J_z - J_x}{J_y}\right)\dot{\phi}\dot{\psi} + \frac{1}{J_y}U_3 \\ \ddot{\psi} &= \left(\frac{J_x - J_y}{J_z}\right)\dot{\theta}\dot{\phi} + \frac{1}{J_z}U_4. \end{aligned} \tag{20}$$

Finally, neglecting the aerodynamic and gyroscopic effects, the dynamic model of the quadrotor in the earth frame can be obtained by combining the translational equations of motion introduced in Equation 11 and the rotational equations of motion introduced in Equation 20 together as;

$$\begin{aligned} \ddot{x} &= \frac{1}{m}\left(s(\theta)c(\phi)c(\psi) + s(\phi)s(\psi)\right)U_1 \\ \ddot{y} &= \frac{1}{m}\left(s(\theta)c(\phi)sin(\psi) - s(\phi)c(\psi)\right)U_1 \\ \ddot{z} &= \frac{1}{m}\left(c(\theta)c(\phi)\right)U_1 - g \\ \ddot{\phi} &= \left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} + \frac{1}{J_x}U_2 \\ \ddot{\theta} &= \left(\frac{J_z - J_x}{J_y}\right)\dot{\phi}\dot{\psi} + \frac{1}{J_y}U_3 \\ \ddot{\psi} &= \left(\frac{J_x - J_y}{J_z}\right)\dot{\theta}\dot{\phi} + \frac{1}{J_z}U_4. \end{aligned} \tag{21}$$

The dynamic system model of the quadrotor described in Equation 21 must be linearized around an equilibrium point to design an LQR controller [43]. This equilibrium point which is also called the "trim" condition can be chosen as the hover position of the quadrotor in which the quadrotor stays stable at a certain altitude [29]. Also, solving the system presented in Equation 21 is difficult due to the trigonometric functions. To avoid this drawback, the linearization of the system can be performed on a simplified version of the dynamic model using the small-disturbance theory based on a small-angle approximation [41]. This method can be employed by approximating $cos(.)$ function to unity as $cos(\alpha) = 1$ and $sin(.)$ function to its argument as $sin(\alpha) = \alpha$ where α is a small angle.

Translational position of the quadrotor must be stable on the hover position. Also, the roll, pitch, and yaw angles need to be zero to keep the quadrotor on a stationary position. Therefore, considering the state vector as;

$$x^T = \begin{bmatrix} x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi} \end{bmatrix}^T \tag{22}$$

the equilibrium state vector x_e^T can be simply defined as:

$$x_e^T = \begin{bmatrix} x_e\ 0\ y_e\ 0\ z_e\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{bmatrix}^T. \tag{23}$$

To stabilize the quadrotor in hover position, a particular force must be applied to the quadrotor to compensate for its own weight. This constant input value for the equilibrium point u_e can be defined as:

$$u_e = \begin{bmatrix} mg \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{24}$$

Finally, by virtue of the small-disturbance theory, the linear dynamics of the quadrotor model around the equilibrium

point described in Equations 23 and 24 can be defined as follows [52,3]:

$$\begin{aligned}
\ddot{x} &= g\theta \\
\ddot{y} &= -g\phi \\
\ddot{z} &= \frac{1}{m}U_1 - g \\
\ddot{\phi} &= \frac{1}{J_x}U_2 \\
\ddot{\theta} &= \frac{1}{J_y}U_3 \\
\ddot{\psi} &= \frac{1}{J_z}U_4.
\end{aligned} \tag{25}$$

The linear dynamics of the quadrotor model in Equation 25 can be redefined as:

$$\begin{aligned}
\ddot{x} &= gU_x \\
\ddot{y} &= -gU_y \\
\ddot{z} &= \frac{1}{m}U_1 - g \\
\ddot{\phi} &= \frac{1}{J_x}U_2 \\
\ddot{\theta} &= \frac{1}{J_y}U_3 \\
\ddot{\psi} &= \frac{1}{J_z}U_4.
\end{aligned} \tag{26}$$

where $U_x = \theta$ and $U_y = \phi$ are virtual control signals in the x and y directions, respectively.

Therefore, considering the state vector presented in Equation 22, the state space form of the linear dynamic system model of the quadrotor can be represented by [52,3,49]:

$$\begin{aligned}
\dot{x} &= Ax + Bu \\
y &= Cx + Du
\end{aligned} \tag{27}$$

where

$$A = \begin{bmatrix}
0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix},$$

$$B = \begin{bmatrix}
0&0&0&0&0&0\\
g&0&0&0&0&0\\
0&0&0&0&0&0\\
0&-g&0&0&0&0\\
0&0&0&0&0&0\\
0&0&1/m&0&0&0\\
0&0&0&0&0&0\\
0&0&0&1/J_x&0&0\\
0&0&0&0&0&0\\
0&0&0&0&1/J_y&0\\
0&0&0&0&0&0\\
0&0&0&0&0&1/J_z
\end{bmatrix},$$

$$C = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0
\end{bmatrix},$$

and D is a (4×6) null matrix with the control matrix

$$u^T = \begin{bmatrix} U_x & U_y & U_1 & U_2 & U_3 & U_4 \end{bmatrix}^T.$$

3 LQR Control

The LQR is a robust optimal control method that consider the system's states and the feedback control gain K to derive a control signal with minimum control effort. Based on this optimal feedback gain K, the feedback control rule may be specified as follows [42,8]:

$$u(t) = -Kx(t) \tag{28}$$

The performance of the LQR controller relies on obtaining an optimal feedback control gain K to minimize a quadratic cost function J. To this end, a quadratic cost function can be defined as

$$J = \int_0^\infty \left(x^T(t)Qx(t) + u^T(t)Ru(t)\right) dt \tag{29}$$

where the control weighting matrix R is a positive definite symmetric matrix and the state weighting matrix Q is a positive semi-definite symmetric matrix [42]. The superscript (T) indicates the transpose of a matrix.

In order to obtain the feedback gain matrix K the Ricatti equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{30}$$

must be solved for the matrix P after selecting the weighting matrices Q and R [42]. One can get the feedback gain matrix K by substituting the derived matrix P into the following equation [42,8]:

$$K = R^{-1}B^T P \tag{31}$$

4 Optimization Algorithms

Choosing the appropriate weighting matrices Q and R of the LQR controller directly affects the controller performance. Thus, instead of the trial and error method, SGA, DE, PSO and GWO algorithms are employed to determine weighting matrices. In this section, the details of the optimization algorithms used in this paper are presented.

4.1 Standard Genetic Algorithm

By inspiring Darwin's Theory of Evolution, John Holland introduced GAs to create a new model for optimization problems [22]. Especially, due to the successes of GAs in the search space [20], his idea has inspired many researchers; therefore, GAs have become algorithms that are frequently used in optimization problems.

a GA commences its procedure, which is stochastic, by randomly creating the initial population. Afterwards, evaluation process starts via a fitness function; thus, the fitness value of each individual in the population is computed by utilizing the fitness function. After the initial population is established, its bio-inspired searching steps starts: selection, crossover (also called recombination), and mutation.

Algorithm 1 Standard Genetic Algorithm

```
Population size = N;
Randomly create the initial population
Evaluate the fitness value of individuals in the population
count ← 1
while count ≤ The number of iteration do
    % Selection and Crossover
    for i = 1 : N do
        Choose individual i via ST
        if p_i ≤ p_c then
            i attends Crossover
            Randomly choose a partner for individual i via ST
            Mate them
        else
            i is saved for the next generation
        end if
    end for
    % Mutation
    for j = 1 : N do
        Randomly choose individual j
        if p_j ≤ p_m then
            j attends Mutation
            Changes the genetic structures of j
        else
            j passes Mutation step
        end if
    end for
    Combine the new individuals and the current population members

    Evaluate the fitness value of each individual
    count = count + 1
end while
```

Selection methods generally tend to select the individuals with the highest fitness value (the best individuals) in the population. Thus, this technique only permits best individuals to survive for the next generation and to mate.

The cross-over step takes individuals coming from the selection step and combines these individuals to create better offspring than their parents [39]. However, it is not required to attend the crossover step. The crossover probability value (p_c) decides whether an individual is going to attend the crossover.

The aim of mutation step is to maintain genetic diversity and prevent squeezing the local minima [39]. As in the crossover phase, the mutation probability value (p_m) decides whether an individual is going to attend the mutation phase. The mutation step changes the genetic structures of chosen individuals to create new individuals.

Finally, new individuals that were created and current population members are combined in a certain proportion to bring about the new generation. With the new generation, the algorithm returns to the selection step and repeats all steps until finding the optimum solution or reaching maximum number of generations.

The Standard Tournament (ST) selection method is commonly preferred in GAs due to its advantages [7]. The GA using ST as the selection method is called the Standard Genetic Algorithm (SGA), whose pseudo-code is shown in Algorithm 1, in the literature [27].

4.2 Differential Evolution

Another well-known EAs that have been utilized in optimization problems is the DE algorithm. As like GAs, DE is also a population-based stochastic algorithm [50].

The first step of DE is creating the initial population, as in GAs. However, instead of proceeding with the selection step, it progresses with the mutation step. In this step, for each individual in the population, the algorithm composes a vector, called *donor vector*, by employing three different individual from the population that are randomly chosen. In the mutation step, the scaling factor F, which should be in [0,2) [50], is the significantly important parameter that controls the differential variation [25].

After creating the donor vector, recombination (crossover) step is commenced. Recombination step decides who is sent for the selection step; the individual or its donor vector. In a basic DE, crossover rate value CR, which should be in the range [0,1] [50], assists this step to give the decision; therefore, *trial vector* is created, see Algorithm 2.

On the other hand, in the selection step, the fitness value of each individual is compared with the fitness value of its trial vector. If the value of trial vector is equal or better than the value of the current member, then, the trial vector replaces the current member for the next generation.

Algorithm 2 Differential Evolution

```
Population size = N;
Specify CR (crossover rate) and F (scaling factor) values
Randomly create the initial population
Evaluate the fitness value of individuals in the population
count ← 1
while count ≤ The number of iteration do
  for i = 1 : N do
    Randomly choose three individuals i1, i2, i3 where i1 ≠ i2 ≠ i3
    Generate a random integer I_rand = {0, 1, 2, ..., D} where D is dimension
    % Trial Vector
    for j = 1 : D do
      if rand(0,1) ≤ CR or j == I_rand(i, j) then
        trial_i = F * |i1 − i2| + i3
      else
        trial_i = pop_i
      end if
    end for
    Replace trial_i, if it is better than pop_i
  end for
  Evaluate the fitness value of individuals in the new population
  count = count + 1
end while
```

Algorithm 3 Particle Swarm Optimization

```
Number of particles in the swarm = N;
Assign random values to each particle in the swarm
Evaluate the fitness value of particles in the swarm
Specify p_Best and g_Best values
count ← 1
while count ≤ The number of iteration do
  for i = 1 : N do
    Determine the velocity of particle i
    Determine the position of particle i
    Evaluate the fitness value of particle i, f_i
    % For a minimization problem
    if f_i ≤ p_Best then
      Assign f_i as p_Best
      if p_Best ≤ g_Best then
        Assign p_Best as g_Best
      end if
    end if
  end for
  Determine the velocity of particle i
  Determine the position of particle i
  count = count + 1
end while
```

4.3 Particle Swarm Optimization

In fact, PSO was not designed for optimization; instead, it was performed to represent the behavior of bird flock and fish swarm in nature [15]. After observing the performance of PSO, it was figured out that the algorithm actually carries out optimization.

PSO creates its initial population (*swarm*) with random solutions. Each potential solution (*particle*) in the population also includes a velocity to move around in the search space. Furthermore, two significant positions are also directly affect the movement of particles: the best position, which has already been stored as *pbest*, and the global best position, in which the swarm has ever found and stored as *gbest*. When a particle finds a better position, then, according to the position it discovered, *pbest*, *gbest* or both are improved, see Algorithm 3.

In a standard PSO, the velocity vector for the i$^{\text{th}}$ individual is calculated as in Equation 32.

$$v_i(t+1) = w.v_i(t) + c_1.r_1.(p_{Best} - x_i(t)) + c_2.r_2.(g_{Best} - x_i(t)) \tag{32}$$

where c_1 and c_2 are acceleration coefficients, w is the inertia weight, r_1 and r_2 are randomly selected real values from the range $[0,1]$, and $x_i(t)$ is the position of the i$^{\text{th}}$ individual at time t.

4.4 Grey Wolf Optimizer (GWO)

GWO was designed by inspiring the hunting strategy of grey wolves in a swarm and their subordinate-superior relations [38]. In these subordinate-superior relations, there are four main kinds of wolves: alpha, beta, omega, and delta.

The alpha wolves are the leaders of the swarm; therefore, they are responsible to make decisions such as hunting, sleeping place, time to wake, etc. On the other hand, the duty of the beta wolves is to assist the alpha wolves while they make decisions. However, delta wolves consist of scouts (warning the swarm in the case of danger), sentinels (protecting the swarm), elders (experienced wolves, used to be alpha or beta), hunters (hunting prey or providing food) and caretakers (caring weak, ill or wounded wolves). The last group of wolves are called omega which are accepted as scapegoats. These group of wolves have to obey all other groups. Based on this hierarchy, GWO was modeled.

GWO starts its procedure by randomly creating the initial population (swarm). After that, it initializes its parameters: $\overrightarrow{A}$ and $\overrightarrow{C}$. $\overrightarrow{A}$ is a vector that contains the random values greater than 1 or less than -1 to oblige the search agent to diverge from the prey. On the other hand, $\overrightarrow{C}$ is also a vector including random values in [0,2] which ensures that GWO behaves randomly throughout its operation [38].

After the initialization, grey wolves positions are determined; therefore, the wolves are identified as alpha, beta or delta based on their current position to the prey. Including of omega wolves, the positions of all other wolves are specified according to the positions of best search agents, see Algorithm 4.

Algorithm 4 Grey Wolf Optimizer (GWO)

```
Population size of grey wolves = N;
Randomly create the initial population
Initialize A and C
Randomly initialize the current positions of wolves
Evaluate the fitness value of wolves in the population
The wolf having the best fitness value is assigned as Alpha
The wolf having the second best fitness value is assigned as Beta
The wolf having the third best fitness value is assigned as Delta
count ← 1
while count ≤ The number of iteration do
   for i = 1 : N − 3 do
      Determine the position of wolf i via the positions of best search
      agents
   end for
   Update the current positions of wolves
   Evaluate the fitness value of wolves in the population
   Update the wolves: Alpha, Beta and Delta
   count = count + 1
end while
```

5 Optimization Results

In this study GA, DE, PSO, and GWO algorithms are employed to optimize the gain matrices of an LQR controller for tracking the trajectory of a quadrotor in the simulation environment.

Optimization training is carried out for total $5s$ simulation with $0.01s$ sampling intervals. During the simulation time, a step signal is selected as the desired trajectory to be tracked by the quadrotor. The desired trajectory can be specified as;

$$\begin{aligned} x_d(t) &= 1 \\ y_d(t) &= 1 \\ z_d(t) &= 1 \end{aligned} \tag{33}$$

where $[x_d, y_d, z_d]$ indicates the desired values of the position vector $[x, y, z]$ with respect to the time. Hence, the tracking errors of the quadrotor positions can be defined as;

$$\begin{aligned} e_x(t) &= x_d(t) - x(t) \\ e_y(t) &= y_d(t) - y(t) \\ e_z(t) &= z_d(t) - z(t) \end{aligned} \tag{34}$$

for x, y, and z axes, respectively.

Since the control objective is described as tracking a predefined trajectory for the quadrotor, the optimization problem on the LQR controller can be defined as selecting appropriate weighting matrices Q and R to decrease position tracking errors e_x, e_y, and e_z.

The Integral Absolute Error (IAE) which is one of the most used error-based performance indices in literature is formulated as;

$$\int_0^T |e(t)|dt \tag{35}$$

where t is time bounded as $t < T$ and $e(t)$ is the error.

Consequently, by employing the IAE on position tracking errors e_x, e_y, and e_z, a multiple objective function to be minimized can be defined as;

$$f_{sum} = c_1 \int_0^T |e_x(t)|dt + c_2 \int_0^T |e_y(t)|dt + c_3 \int_0^T |e_z(t)|dt \tag{36}$$

where c_1, c_2, and c_3 are constants and their values are selected as 1 in this study.

As stated in Section 3, the weighting matrix R has to be a positive definite symmetric matrix, and the weighting matrix Q has to be a positive semi-definite symmetric matrix to guarantee the stability of the system. Selecting the weighting matrices Q and R as diagonal matrices with all diagonal elements positive is one of the straightforward methods to ensure this requirement [17]. Therefore, the weighting matrices Q and R to be selected by the optimization algorithms can be defined as;

$$\begin{aligned} Q &= diag\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}\} \\ R &= diag\{r_1, r_2, r_3, r_4, r_5, r_6\} \end{aligned} \tag{37}$$

where $q_1, q_2, ..., q_{12}$ and $r_1, r_2, ..., r_6$ are the optimization parameters to be encoded by the SGA, DE, PSO, and GWO algorithms.

Table 1: The parameter settings of the algorithms

Algorithm	Parameter	Value
SGA	Tournament size	3
	Crossover rate	0.7
	Mutation probability	0.05
DE	Crossover rate	0.7
	Scaling factor	0.8
PSO	w	0.2
	c1	2
	c2	2
GWO	A	[-1,1]
	C	[0,2]

Table 1 shows the parameter settings of the algorithms that utilized during the tests. The parameter settings of SGA, DE, PSO and GWO were chosen based on their standard parameter settings, which are respectively in [33], [25], [15] and [37]. Moreover, the number of maximum iteration and population (swarm) size were accepted as 100. While the

code of GWO and DE were taken from [37] and [19], respectively, the code of SGA and PSO were modeled according to their pseudo-codes (Algorithm 1 and 3).

Table 2: Parameters of the quadrotor

Parameter	Value	Unit
m	2.353598	kg
g	9.81	m/s^2
J_x	0.1676	kgm^2
J_y	0.1676	kgm^2
J_z	0.29743	kgm^2

In the optimization training, the state space representation of the quadrotor presented in Equation 27 is used. In addition, the system parameters of the quadrotor used in these simulations are presented in Table 2. Block diagram of the optimization process is shown in Figure 2 where the reference input r indicates the desired values of the position vector $[x_d, y_d, z_d]$ presented in Equation 33.

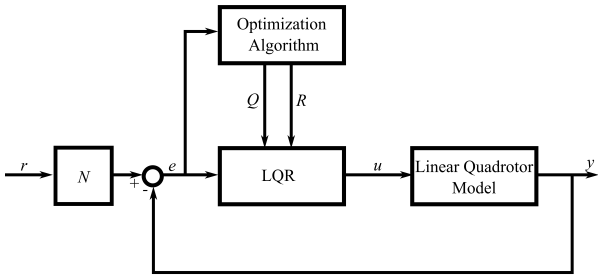


Fig. 2: Block diagram of the optimization process

After selecting the weighting matrices, the feedback gain matrix K is obtained for every optimization method by solving the Ricatti equation presented in Equation 30 for the matrix P using the selected weighting matrices Q and R and substituting the matrix P into Equation 31. Also, a precompensation gain N is added to the reference input to address the steady-state error as shown in Figure 2. The precompensation gain N can be defined as follows [5]:

$$N = -(C(A-BK)^{-1}B)^{-1} \tag{38}$$

where M^{-1} indicates the inverse of the matrix M.

Table 3: Weighting matrix Q selected by optimization algorithms

Q	SGA	DE	PSO	GWO
q_1	427.9468	363.3720	500	500
q_2	80.7950	0.1	0.1	0.1047
q_3	297.3789	382.3142	500	500
q_4	110.9966	0.1	0.1	0.1018
q_5	396.7965	500	500	500
q_6	183.9138	0.1	0.1	0.01
q_7	177.1640	94.7692	0.1	261.1998
q_8	66.5942	191.8772	500	284.0753
q_9	41.6033	0.1	490.6392	248.8252
q_{10}	245.8784	83.8978	0.1	46.0512
q_{11}	324.3094	65.6666	0.1	77.7556
q_{12}	108.5947	358.3660	0.1	185.5017

Table 4: Weighting matrix R selected by optimization algorithms

R	SGA	DE	PSO	GWO
r_1	191.7938	0.1	0.1	0.1
r_2	272.7009	0.1	0.1	0.1
r_3	9.5239	0.1	0.1	0.1
r_4	350.5630	397.2218	0.1	499.9703
r_5	10.7818	247.3427	500	119.5353
r_6	94.2075	220.1522	500	113.5608

The weighting matrices Q and R selected by SGA, DE, PSO and GWO algorithms at the end of the design process are presented in Table 3 and Table 4, respectively, where $Q = diag\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}\}$ and $R = diag\{r_1, r_2, r_3, r_4, r_5, r_6\}$. The convergence of the objective function based on the selected weighting matrices during the optimization process is plotted in Figure 3.

According to Figure 3, SGA gets stuck in local minima throughout the optimization process. While PSO and GWO find the best value at the beginning of the process, nearly at 15^{th} iteration, DE reaches the same value by improving its results during the iterations.

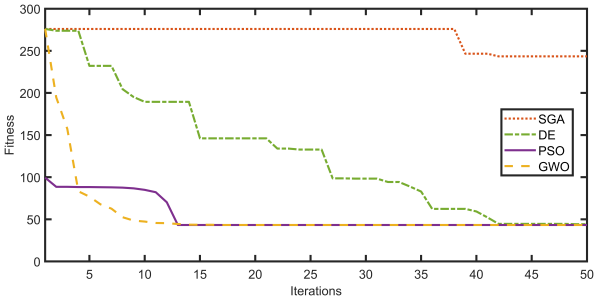


Fig. 3: Convergence of the objective function during the optimization process

Also, details of the tracking performance of designed LQR controllers for the training trajectory based on the weighting matrices selected by optimization algorithms are shown in Figures 4 (a-c). The DE, PSO and GWO algorithms manage to move the quadrotor to the desired position in both directions x and y in less than $0.5s$ and have superior performance over the SGA as shown in Figure 4 (a) and (b).

The Integral Absolute Error (IAE) indices of the tracking performances are presented in Table 5. The PSO algorithm provides the best result in the manner of IAE in the x and y positions as presented in Table 5. DE, PSO, and GWO algorithms produce exactly the same result for the change of position in the z direction as shown in Figure 4 (c) and Table 5. Although both PSO and GWO algorithms create nearly the same results, the overall tracking performance of the PSO algorithm is slightly better than the GWO algorithm as presented in Table 5.

Table 5: Tracking performance of optimization algorithms based on the IAE

	IAE	SGA	DE	PSO	GWO
Step	e_x	57.5332	7.1805	**6.6597**	6.6620
	e_y	75.8809	7.0941	**6.6597**	6.6606
	e_z	110.0331	**29.9215**	**29.9215**	**29.9215**
	total	243.4472	44.1961	**43.2410**	43.2441
Square	e_x	115.0575	14.3610	**13.3195**	13.3241
	e_y	148.9083	14.1883	**13.3195**	13.3213
	e_z	220.2195	**59.8429**	**59.8429**	**59.8429**
	total	484.1853	88.3922	**86.4819**	86.4883
Helix	e_x	422.0160	55.4325	**50.987**	51.0468
	e_y	446.7815	47.4641	**44.1945**	44.2198
	e_z	201.7553	**50.9840**	**50.9840**	**50.9840**
	total	1070.5528	153.8805	**146.1566**	146.2506

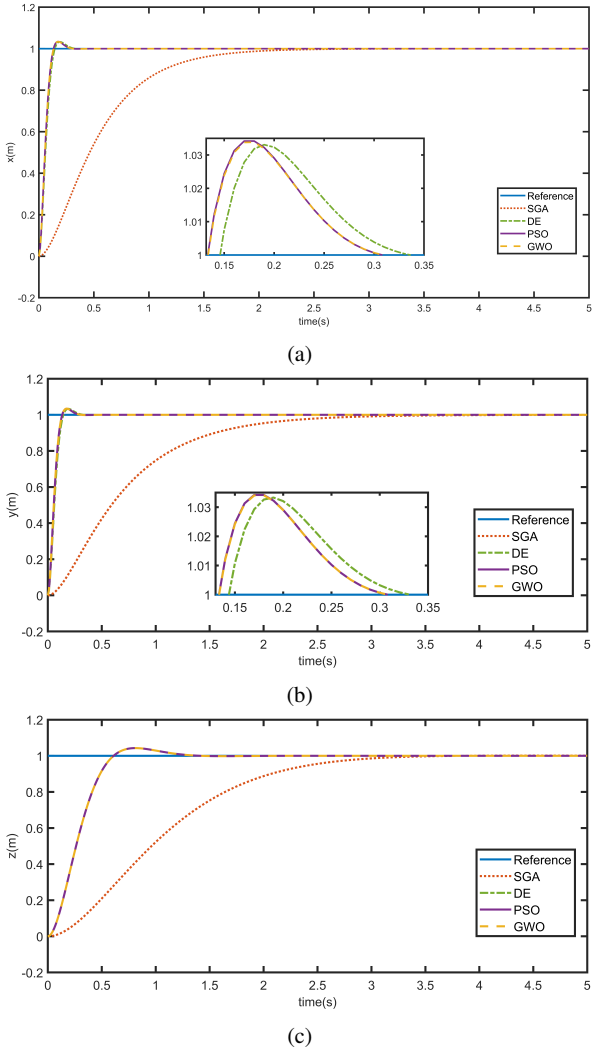


Fig. 4: Optimization results of positions x (a), y (b) and z (c) for step reference

By training the optimization algorithms in a basic step reference, it had been obtained the information of their performances. However, in order to make the results more reliable and significant, it is required to test the performances of the selected weighting matrices in more complex problems. Thus, square and helix trajectories were utilized to analyze the performances of the selected weighting matrices in trajectory tracking in two different simulations.

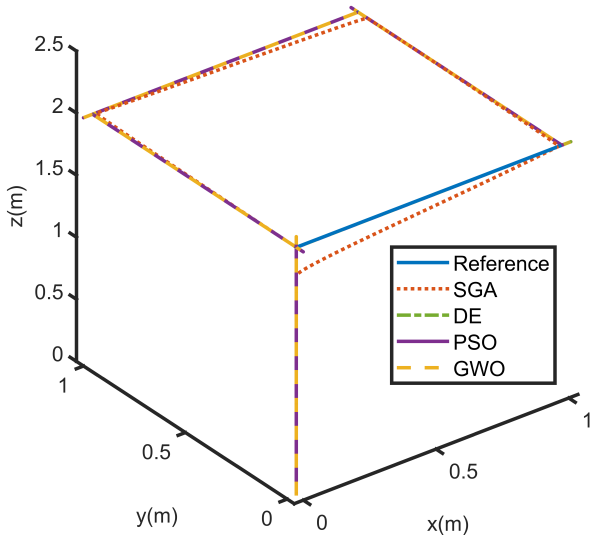
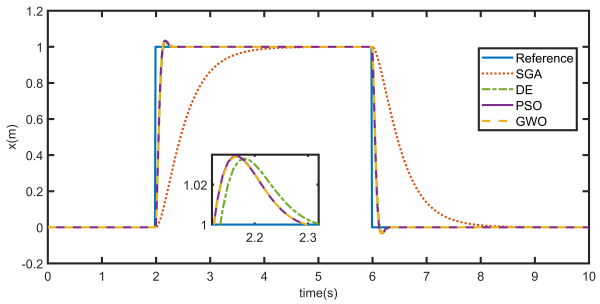
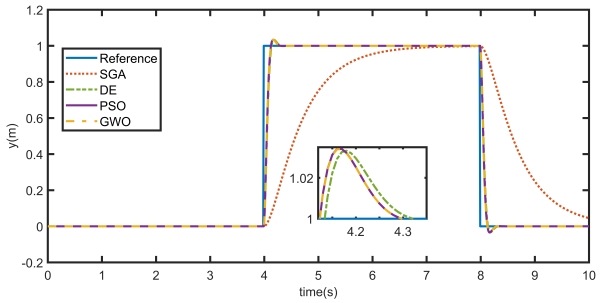


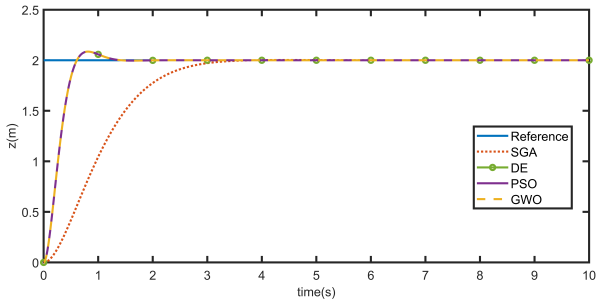
Fig. 5: The square trajectory tracking performances in simulation



(a)



(b)



(c)

Fig. 6: Positions obtained during the square trajectory tracking in simulation on the x (a), y (b), and z (c) axes

The first training trajectory is the square that is shown in Figure 5 where the initial position is selected as (0,0,0). On the other hand, the tracking performances of the optimization algorithms that choose the weighting matrices of the LQR controllers are shown in Figures 6 (a-c). According to these figures, none of the algorithms can follow the correct reference path in 2s. Nevertheless, DE, PSO, and GWO algorithms demonstrate better performance than SGA at all positions; see Figures 6 (a-c).

Based on the IAEs of the square tracking performances presented in Table 5, in the x and y positions, PSO algorithm performs better than other algorithms in the manner of IAE. Furthermore, the DE, PSO, and GWO algorithms present exactly the same result for the change of position in the z direction as in the step reference, which is shown in Table 5.

The helix trajectory where the initial position is chosen as (0,0,0) is the last complex trajectory we tested, see Figure 7. Moreover, details of the tracking performances of designed LQR controllers for the helix training trajectory based on the weighting matrices, which were selected by optimization algorithms, are shown in Figures 8 (a-c). All optimization algorithms, except SGA, manage to move the quadrotor to the desired position in all directions, as shown in Figure 8 (a-c).

According to Table 5, where the IAEs of the helix tracking performances are indicated, the PSO algorithm offers superior performance compared to other algorithms in terms of IAE, as in the square trajectory, in both x and y positions. On the other hand, the DE, PSO and GWO algorithms perform exactly the same in the z direction.

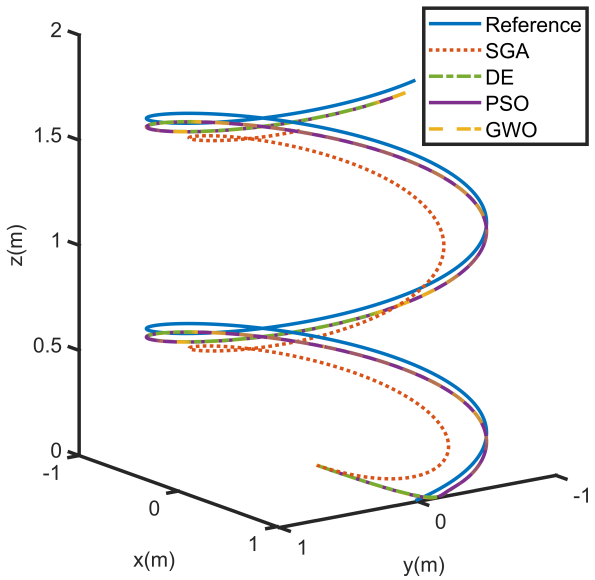


Fig. 7: The helix trajectory tracking performances in simulation

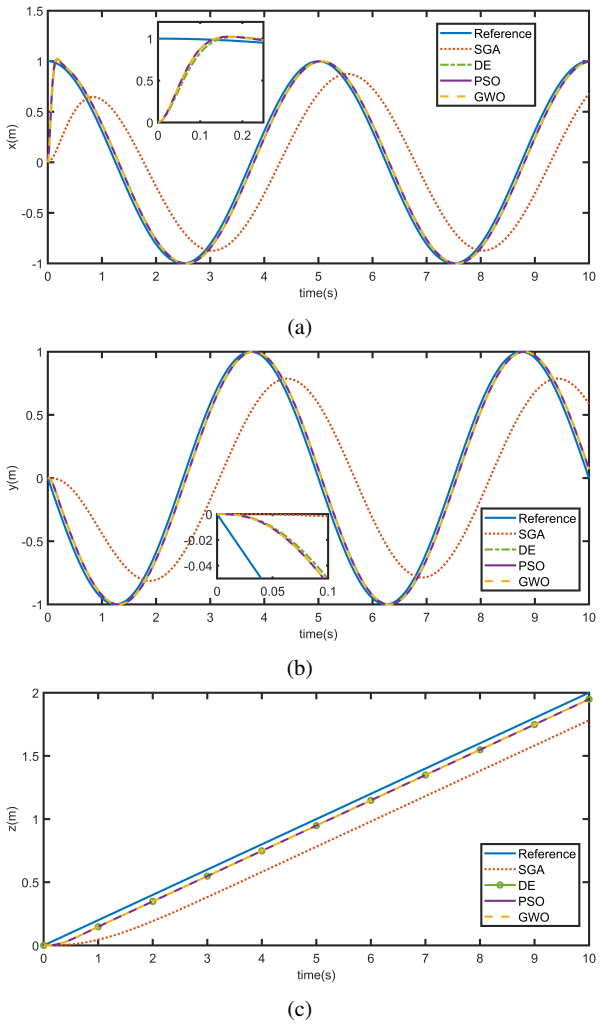


Fig. 8: Positions obtained during the helix trajectory tracking in simulation on the x (a), y (b), and z (c) axes

6 Conclusions

In this paper, SGA, DE, PSO and GWO algorithms were utilized to select weight matrices of the LQR controller that enables a quatrotor with six degrees of freedom (tri-linear tri-rotary) to move automatically in three dimensions on a predetermined trajectory in the simulation environment.

In light of the obtained results, the following remarks can be made:

- Selection of weight matrices of LQR controller is significantly important in the manner of determining the control signal gain.
- SGA does not offer adequate results in the manner of selecting weigh matrices of the LQR controller.
- PSO can be used to select the weight matrices of the LQR controller as it presents the best performances.
- Since there is no such study, it is obvious that the paper contributes to the literature in the manner of selecting weigh matrices of the LQR controller.

For future work, testing the performances of the optimization algorithms in a real-life plant can be recommended.

CRediT authorship contribution statement

Mashar C Gencal: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing. **Baris Ata:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing, Funding acquisition.

Data availability

No data was used for the research described in the paper.

Conflict of interest

The authors declare that they have no conflict of interest.

Acknowledgements This work was supported by the Research Fund of Cukurova University. Project Number: FBA-2022-14643.

References

1. Abdelhay, S., Zakriti, A.: Modeling of a quadcopter trajectory tracking system using pid controller. Procedia Manufacturing **32**, 564–571 (2019)
2. Abdolhosseini, M., Zhang, Y.M., Rabbath, C.A.: An Efficient Model Predictive Control Scheme for an Unmanned Quadrotor Helicopter. Journal of Intelligent & Robotic Systems **70**(1), 27–38 (2013)
3. Araar, O., Aouf, N.: Full linear control of a quadrotor UAV, LQ vs H;. In: 2014 UKACC International Conference on Control, pp. 133–138. IEEE, Loughborough, UK (2014)
4. Artale, V., Milazzo, C.L., Orlando, C., Ricciardello, A.: Comparison of ga and pso approaches for the direct and lqr tuning of a multirotor pd controller. Journal of Industrial & Management Optimization **13**(4), 2067 (2017)
5. Ata, B., Coban, R.: Linear Quadratic Optimal Control of an Inverted Pendulum on a Cart using Artificial Bee Colony Algorithm: An Experimental Study. Cukurova University Journal of the Faculty of Engineering and Architecture **32**(2), 109–124 (2017)
6. Basri, M.A.M., Husain, A.R., Danapalasingam, K.A.: Nonlinear control of an autonomous quadrotor unmanned aerial vehicle using backstepping controller optimized by particle swarm optimization. Journal of Engineering Science & Technology Review **8**(3) (2015)
7. Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation **4**(4), 361–394 (1996)
8. Bradtke, S.J., Ydstie, B.E., Barto, A.G.: Adaptive linear quadratic control using policy iteration. In: Proceedings of 1994 American Control Conference-ACC'94, vol. 3, pp. 3475–3479. IEEE (1994)

9. Burggräf, P., Pérez Martínez, A.R., Roth, H., Wagner, J.: Quadrotors in factory applications: design and implementation of the quadrotor's p-pid cascade control system. SN Applied Sciences **1**(7), 1–17 (2019)
10. Castillo-Zamora, J.J., Camarillo-Gomez, K.A., Perez-Soto, G.I., Rodriguez-Resendiz, J.: Comparison of pd, pid and sliding-mode position controllers for v–tail quadcopter stability. Ieee Access **6**, 38086–38096 (2018)
11. Cengiz, S.K., Ucun, L.: Optimal controller design for autonomous quadrotor landing on moving platform. Simulation Modelling Practice and Theory p. 102565 (2022)
12. Chang, K., Xia, Y., Huang, K., Ma, D.: Obstacle avoidance and active disturbance rejection control for a quadrotor. Neurocomputing **190**, 60–69 (2016)
13. Chen, H., Yang, Y., Sun, J.: Improved genetic algorithm based optimal control for a flying inverted pendulum. In: 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), pp. 1428–1432. IEEE (2019)
14. Cowling, I.D., Yakimenko, O.A., Whidborne, J.F., Cooke, A.K.: A prototype of an autonomous controller for a quadrotor uav. In: 2007 European Control Conference (ECC), pp. 4001–4008. IEEE (2007)
15. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, pp. 39–43. IEEE (1995)
16. Erkol, H.O.: Attitude controller optimization of four-rotor unmanned air vehicle. International Journal of Micro Air Vehicles **10**(1), 42–49 (2018)
17. Franklin, G.F., Powell, J.D., Emami-Naeini, A., Powell, J.D.: Feedback control of dynamic systems. Prentice Hall (2002)
18. Gao, Y., Li, R., Shi, Y., Xiao, L.: Design of path planning and tracking control of quadrotor. Journal of Industrial and Management Optimization **18**(3), 2221 (2022)
19. Gencal, M.C.: The implementation of differential evolution in matlab. URL `https://www.mathworks.com/matlabcentral/fileexchange/77617-the-implementation-of-differential-evolution-in-matlab`
20. Goldberg, D.E.: Genetic algorithms in search. Optimization, and MachineLearning (1989)
21. Günsel, S., Engin, Ş.N.: The effects of pso parameters on an lqr controlled quadrotor system gain. Tech. rep., EasyChair (2021)
22. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
23. Huang, T., Huang, D., Wang, Z., Dai, X., Shah, A.: Generic Adaptive Sliding Mode Control for a Quadrotor UAV System Subject to Severe Parametric Uncertainties and Fully Unknown External Disturbance. International Journal of Control, Automation and Systems **19**(2), 698–711 (2021)
24. İçen, M., Ateş, A., Yeroğlu, C.: Optimization of lqr weight matrix to control three degree of freedom quadcopter. In: 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), pp. 1–6. IEEE (2017)
25. Ji, S., Karlovšek, J.: Optimized differential evolution algorithm for solving dem material calibration problem. Engineering with Computers pp. 1–16 (2022)
26. Jia, Z., Yu, J., Mei, Y., Chen, Y., Shen, Y., Ai, X.: Integral backstepping sliding mode control for quadrotor helicopter under external uncertain disturbances. Aerospace Science and Technology **68**, 299–307 (2017)
27. Jiacheng, L., Lei, L.: A hybrid genetic algorithm based on information entropy and game theory. Ieee Access **8**, 36602–36611 (2020)
28. Joelianto, E., Christian, D., Samsi, A.: Swarm control of an unmanned quadrotor model with lqr weighting matrix optimization using genetic algorithm. Journal of Mechatronics, Electrical Power, and Vehicular Technology **11**(1), 1–10 (2020)
29. Joukhadar, A., Hasan, I., Alsabbagh, A., Alkouzbary, M.: Integral Lqr-Based 6dof Autonomous Quadrocopter Balancing System Control. International Journal of Advanced Research in Artificial Intelligence **4** (2015)
30. Lahmeri, M.A., Kishk, M.A., Alouini, M.S.: Artificial intelligence for uav-enabled wireless networks: A survey. IEEE Open Journal of the Communications Society **2**, 1015–1040 (2021)
31. Lee, D., Burg, T.C., Dawson, D.M., Shu, D., Xian, B., Tatlicioglu, E.: Robust tracking control of an underactuated quadrotor aerial-robot based on a parametric uncertain model. In: 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 3187–3192 (2009)
32. Li, J., Li, Y.: Dynamic analysis and pid control for a quadrotor. 2011 IEEE International Conference on Mechatronics and Automation, ICMA 2011 pp. 573–578 (2011)
33. Lobo, F.G., Goldberg, D.E.: The parameter-less genetic algorithm in practice. Information Sciences **167**(1-4), 217–232 (2004)
34. Lu, J., Yang, Y., Jin, X.: Quadrotor inverted pendulum control based on improved particle swarm optimization. In: 2021 China Automation Congress (CAC), pp. 6280–6285. IEEE (2021)
35. Mahmoodabadi, M., Babak, N.R.: Robust fuzzy linear quadratic regulator control optimized by multi-objective high exploration particle swarm optimization for a 4 degree-of-freedom quadrotor. Aerospace Science and Technology **97**, 105598 (2020)
36. Martins, L., Cardeira, C., Oliveira, P.: Linear quadratic regulator for trajectory tracking of a quadrotor. IFAC-PapersOnLine **52**(12), 176–181 (2019)
37. Mirjalili, S.: Grey wolf optimizer (gwo). URL `https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo`
38. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in engineering software **69**, 46–61 (2014)
39. Mitchell, M.: An introduction to genetic algorithms. MIT press (1998)
40. Mohanty, S., Misra, A.: 3 dof autonomous control analysis of an quadcopter using artificial neural network. In: Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough, pp. 39–57. Springer (2020)
41. Nelson, R.C.: Flight Stability and Automatic Control, 2nd ed edn. McGraw Hill, Boston (1998)
42. Ogata, K.a.: Modern control engineering. Prentice Hall, NJ (2010)
43. Okyere, E., Bousbaine, A., Poyi, G.T., Joseph, A.K., Andrade, J.M.: Lqr controller design for quad-rotor helicopters. The Journal of Engineering **2019**(17), 4003–4007 (2019)
44. Önen, Ü., Çakan, A., Ilhan, I.: Performance comparison of optimization algorithms in lqr controller design for a nonlinear system. Turkish Journal of Electrical Engineering and Computer Sciences **27**(3), 1938–1953 (2019)
45. Pawłowski, P., Konatowski, S.: Linear controller design with the use of pso algorithm for uav trajectory tracking. In: Radioelectronic Systems Conference 2019, vol. 11442, p. 114420Z. International Society for Optics and Photonics (2020)
46. Shah, J., Okasha, M., Faris, W.: Gain scheduled integral linear quadratic control for quadcopter. Int J Eng Technol **7**(4.13), 81–85 (2018)
47. Shauqee, M.N., Rajendran, P., Suhadis, N.M.: An effective proportional-double derivative-linear quadratic regulator controller for quadcopter attitude and altitude control. Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije **62**(3-4), 415–433 (2021)
48. Shauqee, M.N., Rajendran, P., Suhadis, N.M.: Proportional double derivative linear quadratic regulator controller using improvised grey wolf optimization technique to control quadcopter. Applied Sciences **11**(6), 2699 (2021)
49. Sir Elkhatem, A., Naci Engin, S.: Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control. Alexandria Engineering Journal **61**(8), 6275–6292 (2022)

50. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)
51. Sun, Y., Xian, N., Duan, H.: Linear-quadratic regulator controller design for quadrotor based on pigeon-inspired optimization. Aircraft Engineering and Aerospace Technology (2016)
52. Wang, P., Man, Z., Cao, Z., Zheng, J., Zhao, Y.: Dynamics modelling and linear control of quadcopter. In: 2016 International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 498–503 (2016). ISSN: 2325-0690
53. Xiu, C., Liu, F., Xu, G.: General model and improved global sliding mode control of the four-rotor aircraft. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering **232**(4), 383–389 (2018)
54. Yang, Y., Yan, Y.: Attitude regulation for unmanned quadrotors using adaptive fuzzy gain-scheduling sliding mode control. Aerospace Science and Technology **54**, 208–217 (2016)
55. Yildirim, A.S., Berker, E., Kayakesen, M.E.: System level test automation in uav development. In: 2018 IEEE AUTOTESTCON, pp. 1–6. IEEE (2018)
56. Zhang, X., Li, X., Wang, K., Lu, Y.: A survey of modelling and identification of quadrotor robot. Abstract and Applied Analysis **2014**, 1–16 (2014)