Comparative Study of Particle Swarm Optimization and Genetic Algorithm for Automated LQR Gain Tuning in Quadrotor Trajectory Tracking

Temidayo Ayanda and Uchenna Charles Onyema, *Member, IEEE*

***Abstract*—Drone applications in time-critical scenarios such as urgent medical deliveries and post-disaster area mapping require robust flight controllers that ensure effective performance while minimising actuator effort. The Linear Quadratic Regulator (LQR) offers high performance and robustness; however, the absence of a systematic method for selecting its weighting matrices means performance is not guaranteed and depends on the designer's expertise. This paper addresses that gap by evaluating and comparing Particle Swarm Optimization (PSO) and the Genetic Algorithm (GA) for automated determination of optimal LQR weighting matrices for quadrotor trajectory tracking. Performance is assessed via trajectory feasibility, position error at each waypoint, and convergence speed. PSO consistently achieved a lower cost (72.83) than GA (74.94 best-case) and converged in under one second, versus 77–92 s for GA. PSO also produced smoother inter-waypoint transitions, making it the more suitable algorithm for this application.**

***Index Terms*—Genetic Algorithm, Linear Quadratic Regulator, optimal control, Particle Swarm Optimization, quadrotor, trajectory tracking, UAV.**

I. INTRODUCTION

UNMANNED Aerial Vehicles (UAVs) are a class of aircraft capable of operating without an on-board pilot. Multi-rotor platforms have gained considerable popularity owing to their versatility across applications including environmental monitoring, surveillance, and search-and-rescue. These vehicles are equipped with sensors, actuators, and embedded processors that collectively enable autonomous flight. A prominent research area is the development of control strategies that allow UAVs to accurately follow pre-programmed trajectories, with growing interest in machine learning to yield adaptive and intelligent behaviours.

There are various control strategies for quadcopters, encompassing both linear and non-linear techniques. Research has shown that LQR and PID are most widely used for the linear approach. Martins, Cardeira and Oliveira [1] designed an inner-outer control structure using LQR with integrative action for trajectory control, achieving success. Abdelhay and Zakriti [2] demonstrated similar success using a cascaded PID controller, whose simulations showed close trajectory following, though both works exhibited some underdamping and overdamping in the transient response.

In this paper, a linearised model of a quadcopter is derived about the hover equilibrium so that an LQR controller can be applied. The LQR cost function is then embedded within two meta-heuristic optimization algorithms PSO and GA to determine the optimal Q and R weighting matrices that yield the best trajectory-tracking performance.

The remainder of the paper is organised as follows: Section II presents the mathematical model and its linearisation; Section III describes the optimisation framework; Section IV presents simulation results; and Section V provides concluding remarks.

II. MATHEMATICAL MODEL

In this section, the non-linear mathematical model of the Quadcopter is presented. Refer to [3] for more information. As shown in the figure below, the mathematical model is derived using the 2 frames of reference. The body frame and the inertial frame. The origin of the body frame is the centre of the Quadcopter.

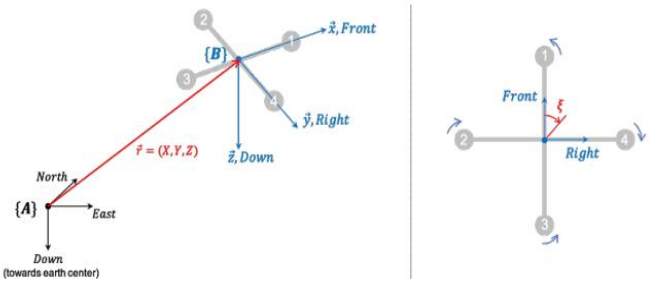

Figure 1 – Coordinate frames Quad (+) configuration

The vector containing the linear and angular position of the quadrotor in the inertial/earth frame is $[\mathrm{x\ y\ z\ \varphi\ \theta\ \psi}]^{\mathrm{T}}$. The vector containing the linear and angular position in the quadrotor's body frame is $[\mathrm{u\ v\ w\ p\ q\ r}]^{\mathrm{T}}$.

$$v = R * v_B, w = T * w_B \quad (1)$$

Manuscript submitted for review [date to be inserted by IEEE]. This work was carried out at the University of Derby. The corresponding author is U. C. Onyema. *(Corresponding author: Uchenna Charles Onyema.)*

T. Ayanda is with the Department of Engineering, University of Derby, Derby DE22 1GB, UK. (temidayo@hotmail.co.uk)

U. C. Onyema is with the Department of Engineering, University of Derby, Derby DE22 1GB, UK (e-mail: onyemauchenna@gmail.com).

Where $v = [\dot{x}\ \dot{y}\ \dot{z}]^T$, $w = [\dot{\varphi}\ \dot{\theta}\ \dot{\psi}]^T$, $v_B = [u\ v\ w]^T$ and $w_B = [p\ q\ r]^T$.

T is a matrix for angular transformations and is equal to

$$T = \begin{bmatrix} 1 & s(\varphi)t(\theta) & c(\varphi)t(\theta) \\ 0 & c(\varphi) & -s(\varphi) \\ 0 & \frac{s(\varphi)}{c(\theta)} & \frac{c(\varphi)}{c(\theta)} \end{bmatrix}$$

$t(\theta) = \tan(\theta)$. This gives an overall kinematic model of

$$\begin{cases} \dot{x} = w[s(\varphi)s(\psi) + c(\varphi)c(\psi)s(\theta)] - v[c(\varphi)s(\psi) - c(\psi)s(\varphi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\varphi)c(\psi) + s(\varphi)s(\psi)s(\theta)] - w[c(\psi)s(\varphi) - c(\varphi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\varphi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\varphi)] \\ \dot{\varphi} = p + r[c(\varphi)t(\theta)] + q[s(\varphi)t(\theta)] \\ \dot{\theta} = q[c(\varphi)] - r[s(\varphi)] \\ \dot{\psi} = r\frac{c(\varphi)}{c(\theta)} + q\frac{s(\varphi)}{c(\theta)} \end{cases} \quad (2)$$

Newton's law states that the total force acting on a quadrotor is shown by:

$$f_B = m(\omega B \wedge vB + \dot{v}B) \quad (3)$$

Where m is the quadrotor's mass, $\wedge$ is the cross product and $f_B = [f_x\ f_y\ f_z]^T$ is the total force.

The total torque applied to the quadrotor is given by Euler's equation:

$$m_B = I \cdot \dot{\omega} B + \omega B \wedge (I \cdot \omega B) \quad (4)$$

where $m_B = [m_x\ m_y\ m_z]^T$ and I is the diagonal inertia matrix

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

So, the dynamic model of the quadrotor in the body frame is:

$$\begin{cases} f_x = m(\dot{u} + qw - rv) \\ f_y = m(\dot{v} - pw + ru) \\ f_z = m(\dot{w} + pv - qu) \\ m_x = \dot{p}I_x - qrI_y + qrI_z \\ m_y = \dot{q}I_y + prI_x - prI_z \\ m_z = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad (5)$$

The external forces in the body frame f_B are given by:

$$f_B = mgR^T * \hat{e_z} - f_t\hat{e_3} + f_w \quad (6)$$

$\hat{e_z}$ is the unit vector in the inertial z axis, $\hat{e_3}$ is the unit vector in the body z axis, g is the gravitational acceleration, f_t is the total thrust generated by the rotors and $f_w = [f_{wx}\ f_{wy}\ f_{wz}]^T$ are the forces produced by wind on the quadrotor.

The external moments in the body frame m_B are given by:

$\hat{e_z}$ is the unit vector in the inertial z axis, $\hat{e_3}$ is the unit vector in the body z axis, g is the gravitational acceleration, f_t is the total thrust generated by the rotors and $f_w = [f_{wx}\ f_{wy}\ f_{wz}]^T$ are the forces produced by wind on the quadrotor.

The external moments in the body frame m_B are given by:

$$m_B = \tau B - ga + \tau w \quad (7)$$

Where g_a represents the gyroscopic moments caused by the combined rotation of the four motors and quadrotor body.

The control torques generated by the differences in rotor speeds is represented by $T_B = [T_x\ T_y T_z]^T$ and the torques produced by the wind on the quadrotors is represented by $T_w = [T_{wx}\ T_{wy} T_{wz}]^T$.

g_a is given by $g_a = \sum_{i=1}^{4} J_p(\omega B \wedge \hat{e}_3)(-1)^{i+1}\Omega_i \quad (8)$

J_p is the inertia of each rotor and Ω_i is the angular speed of rotor i. Mellinger et al. in [4] finds J_p to be too small to be effective in so the gyroscopic moments are removed. There are several aerodynamic and aeroelastic effects that disturbs a quadrotor during flight. Because of this, the force expression is substituted into the dynamic model. This gives:

$$\begin{cases} -mg[s(\theta)] + f_{wx} = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\varphi)] + f_{wy} = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\varphi)] + f_{wz} - f_t = m(\dot{w} + pv - qu) \\ \tau_x + \tau_{wx} = \dot{p}I_x - qrI_y + qrI_z \\ \tau_y + \tau_{wy} = \dot{q}I_y + prI_x - prI_z \\ \tau_z + \tau_{wz} = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad (9)$$

A. Actuator Dynamics

Here the control input is considered. The control input for the quadrotor system are the 4 rotors. As mentioned above, there are 4 ways the quadrotor can move. Each control input represents one of those 4 movements. The values of the input forces and torques are proportional to the squared speeds (Ω_i^2) of the rotors. The input forces and torques are shown below:

$$\begin{cases} f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x = bl(\Omega_3^2 - \Omega_1^2) \\ \tau_y = bl(\Omega_4^2 - \Omega_2^2) \\ \tau_z = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{cases} \quad (10)$$

l is the distance between any rotor and the centre of the drone, b is the thrust factor and d is the drag factor. f_t corresponds to thrust, t_x to roll, t_y to pitch and t_z to yaw. The actuator dynamics are substituted into the dynamic model. The dynamic model becomes:

$$\begin{cases} -mg[s(\theta)] + f_{wx} = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\varphi)] + f_{wy} = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\varphi)] + f_{wz} - b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) = m(\dot{w} + pv - qu) \\ bl(\Omega_3^2 - \Omega_1^2) + \tau_{wx} = \dot{p}I_x - qrI_y + qrI_z \\ bl(\Omega_4^2 - \Omega_2^2) + \tau_{wy} = \dot{q}I_y + prI_x - prI_z \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 + \Omega_3^2) + \tau_{wz} = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad (11)$$

B. State Space

The state vectors are organised as shown below:

$$x = [\varphi\ \theta\ \psi\ p\ q\ r\ u\ v\ w\ x\ y\ z]^T$$

The equations of dynamics are rewritten into state space form:

$$\begin{cases} \dot{\varphi} = p + r[c(\varphi)t(\theta)] + q[s(\varphi)t(\theta)] \\ \dot{\theta} = q[c(\varphi)] - r[s(\varphi)] \\ \dot{\psi} = r\,c(\varphi)c(\theta) + q\,s(\varphi)c(\theta) \\ \dot{p} = \frac{Iy-Iz}{Ix} * rq + \frac{\tau x+\tau wx}{Ix} \\ \dot{q} = \frac{Iz-Ix}{Iy} * pr + \frac{\tau y+\tau wy}{Iy} \\ \dot{r} = \frac{Ix-Iy}{Iz} * pq + \frac{\tau z+\tau wz}{Iz} \\ \dot{u} = rv - qw - g[s(\theta)] + \frac{fwx}{m} \\ \dot{v} = pw - ru + g[s(\varphi)c(\theta)] + \frac{fwy}{m} \\ \dot{w} = qu - pv + g[c(\theta)c(\varphi)] + \frac{fwz-ft}{m} \\ \dot{x} = w[s(\varphi)s(\psi) + c(\varphi)c(\psi)s(\theta)] - v[c(\varphi)s(\psi) - c(\psi)s(\varphi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\varphi)c(\psi) + s(\varphi)s(\psi)s(\theta)] - w[c(\psi)s(\varphi) - c(\varphi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\varphi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\varphi)] \end{cases} \quad (12)$$

Air resistance has been factored in equation 12 above but will be ignored moving forward in the model.

From Newton's law it can be written

$$m\dot{v} = R * f_B = mg\hat{e}_z - f_tR * \hat{e}_3 \quad (13)$$

The dynamic model of the quadrotor becomes:

$$\begin{cases} \ddot{x} = -\frac{ft}{m}[s(\varphi)s(\psi) + c(\varphi)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{ft}{m}[c(\theta)s(\psi)s(\theta) - c(\psi)s(\varphi)] \\ \ddot{z} = g - \frac{ft}{m}[c(\varphi)c(\theta)] \end{cases} \quad (14)$$

A simplification by setting $[\dot{\varphi}\ \dot{\theta}\ \dot{\psi}]$ equal to $[p\ q\ r]^T$ makes the dynamic model in the inertial frame:

$$\begin{cases} \ddot{x} = -\frac{f_t}{m}[s(\varphi)s(\psi)\ +\ c(\varphi)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{f_t}{m}[c(\varphi)s(\psi)s(\theta)\ -\ c(\psi)s(\varphi)] \\ \ddot{z} = g\ -\frac{f_t}{m}[c(\varphi)c(\theta)] \\ \ddot{\varphi} = \frac{Iy-Iz}{Ix}\dot{\theta}\dot{\psi} + \frac{\tau x}{Ix} \\ \ddot{\theta} = \frac{Iz-Ix}{Iy}\dot{\varphi}\dot{\psi} + \frac{\tau y}{Iy} \\ \ddot{\psi} = \frac{Ix-Iy}{Iz}\dot{\varphi}\dot{\theta} + \frac{\tau z}{Iz} \end{cases} \quad (15)$$

This model can be summarised as

$$\dot{x} = f(x,u)$$

where the states are $x = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \varphi\ \theta\ \psi\ p\ q\ r]^T$

$$u = [u_1\ u_2\ u_3\ u_4]^T = [f_t\ \tau x\ \tau y\ \tau z] \quad (16)$$

Now $\dot{x}$ can be expressed in terms of x and u. This is shown below:

$$\begin{bmatrix} \dot{x} = \dot{x} \\ \dot{y} = \dot{y} \\ \dot{z} = \dot{z} \\ \ddot{x} = -\frac{f_t}{m}[s(\varphi)s(\psi)\ +\ c(\varphi)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{f_t}{m}[c(\varphi)s(\psi)s(\theta)\ -\ c(\psi)s(\varphi)] \\ \ddot{z} = g\ -\frac{f_t}{m}[c(\varphi)c(\theta)] \\ \dot{\varphi} = p + q[s(\varphi)t(\theta)] + r[c(\varphi)t(\theta)] \\ \dot{\theta} = q[c(\varphi)] - r[s(\varphi)] \\ \dot{\psi} = q\big(s(\varphi)sec(\theta)\big) + r\big(c(\varphi)sec(\theta)\big) \\ \dot{p} = \frac{Iy-Iz}{Ix}qr + \frac{\tau x}{Ix} \\ \dot{q} = \frac{Iz-Ix}{Iy}pr + \frac{\tau y}{Iy} \\ \dot{r} = \frac{Ix-Iy}{Iz}pq + \frac{\tau z}{Iz} \end{bmatrix} \quad (17)$$

C. Linearization of the Quadrotor Dynamic Model

The quadcopter is linearized around an equilibrium point—the hovering position. At this point the states become:

$$\overline{x} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \psi\ 0\ 0\ 0]^T$$

and the controls become:

$$\overline{u} = [mg\ 0\ 0\ 0]^T.$$

This produces a linear model characterised by the state space equation ẋ = Ax + Bu. The A and B matrices are shown as:

$$A = \frac{\partial f(x,u)}{\partial x}|_{\substack{x=\overline{x} \\ u=\overline{u}}} \begin{bmatrix} 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&-gsin(\psi)&-gcos(\psi)&0&0&0&0 \\ 0&0&0&0&0&0&gcos(\psi)&-gsin(\psi)&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \quad (18)$$

$$B = \frac{\partial f(x,u)}{\partial u}|_{\substack{x=\overline{x} \\ u=\overline{u}}} \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ -\frac{1}{m}&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&\frac{1}{I_x}&0&0 \\ 0&0&\frac{1}{I_y}&0 \\ 0&0&0&\frac{1}{I_z} \end{bmatrix} \quad (19)$$

III. OPTIMISATION

The Linear Quadratic Regulator (LQR) provides optimally controlled feedback gains. For a linear system with the state representation:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

The cost function is:

$$J = \int_0^\infty [x^T Q x + u^T R u]\, dt$$

Where Q is a symmetrical positive definite matrix, used to weight the states. R is a symmetrical semi-positive definite matrix, used to weight the controls.

LQR is a full-state feedback controller. It is controlled using its states and a gain K:

$$u(t) = -Kx(t)$$

The gain K is:

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

P is found by solving the Algebraic Riccati Equation:

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

With the gain K, the system can be controlled. Further reading on this can be found from [5], [6] and [7]. The dynamic model was split into 4 subsystems for X, Y, Z and yaw. The yaw was considered to be 0.

X Subsystem

The state variables are $x, \dot{x}, \theta\ and\ \dot{\theta}$.

Ax is: $\begin{bmatrix} 0&1&0&0 \\ 0&0&g&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix}$

Bu is: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/Ix \end{bmatrix}$

Y Subsystem

The state variables are $y, \dot{y}, \varphi\ and\ \dot{\varphi}$.

Ax is: $\begin{bmatrix} 0&1&0&0 \\ 0&0&-g&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix}$

Bu is: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/Iy \end{bmatrix}$

Z Subsystem

The state variables are $z, \dot{z}$.

Ax is: $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

Bu is: $\begin{bmatrix} 0 \\ 1/m \end{bmatrix}$

Yaw Subsystem

The state variables are $\psi, \dot{\psi}$.

Ax is: $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

Bu is: $\begin{bmatrix} 0 \\ 1/Iz \end{bmatrix}$

Q and R matrices the same size as Ax and Bu respectively were created and put into the PSO and GA algorithms.

A. Particle Swarm Optimisation Algorithm

Particle Swarm Optimisation (PSO) is a population-based, bio-inspired metaheuristic that searches for an optimal solution by iteratively moving a swarm of candidate solutions (particles) through the search space. Each particle evaluates the objective function at its current position and updates its trajectory based on its own best-known position and the global best-known position, converging towards the global minimum. Fig. 2 shows the algorithmic flow.

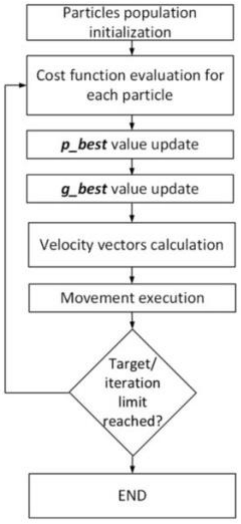

Fig. 2. PSO algorithm flowchart.

The population was the diagonal values of the Q and R matrices. The population size was set to 100 and the Max Iterations was varied. The position of each particle was updated using the formula:

$X^i(t+1) = x^i(t) + v_x^i(t+1)$

And the velocity was updated using the formula:

$V^i(t+1) = wV^i(t) + c_1 r_1 \left(pbest^i - X^i(t)\right) + c_2 r_2 (gbest - X^i(t))$

Where w, c_1, c_2 are parameters of the algorithm. $pbest^i$ is the local minimum value and $gbest^i$ is the global minimum value. The value being evaluated was the cost function resulting from the Q and R matrix values.

Steps 1: Randomly generate an initial population within the constraint range.

Step 2: Assign individuals to Q and R, compute K via LQR, and evaluate the cost.

Step 3: Store values producing minimal cost as the global best.

Step 4: Repeat Steps 2–4 until the maximum iteration count is reached; return the global best.

B. Genetic Algorithm

The Genetic Algorithm (GA) is a stochastic global optimisation method inspired by biological natural selection. A population of candidate solutions (chromosomes) evolves across successive generations: the fittest individuals are selected for reproduction, with offspring generated through crossover and mutation. Fig. 3 illustrates the algorithmic flow.

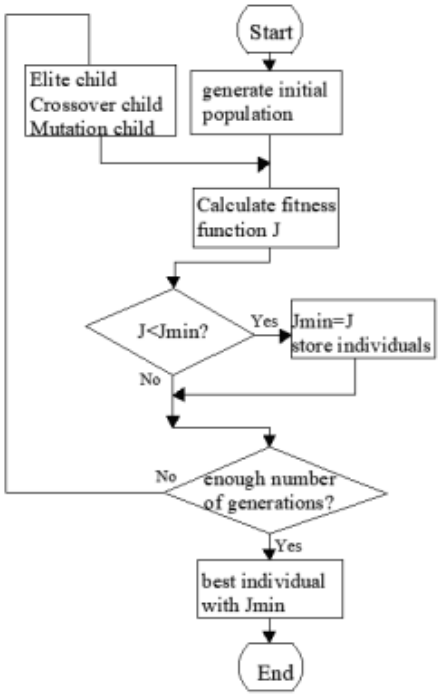

Fig. 3. GA algorithm flowchart.

The genes of the chromosome were set to the diagonal values of the Q and R matrices. The population size was set to 100 and the Max Iterations was varied. The specific steps followed are

Step 1: Generate initial population.

Step 2: Assign to Q and R, compute K via LQR, evaluate cost.

Step 3: Retain lowest-cost individuals and form new population through crossover and mutation.

Step 4: Repeat steps 2 - 4 until maximum iterations reached.

Both algorithms were evaluated across five cases.

IV. RESULTS

A. Minimum Performance When Comparing Maximum Iterations

Table I summarises the cost values and convergence times for both algorithms across the five cases.

TABLE I
PSO VS. GA COST AND CONVERGENCE TIME

Max Iterations	100	80	60	40	20
Population Size	100	100	100	100	100
PSO Cost	72.833	72.832	72.833	72.833	72.833
PSO Conv. Time (s)	0.38	0.38	0.39	0.37	0.39
GA Cost	87.76	74.94	77.83	94.37	109.72

GA Conv. Time (s)	87.76	85.41	77.83	91.93	85.83

The results show that PSO consistently achieved a lower cost than GA in every case, converging in under one second while GA required 77–92 s. GA achieved its lowest cost of 74.94 at 80 maximum iterations, close to but not matching PSO's 72.83. PSO may have converged to a local minimum, as the algorithm can be susceptible to premature convergence.

B. Feasibility of Trajectory Algorithms Provided

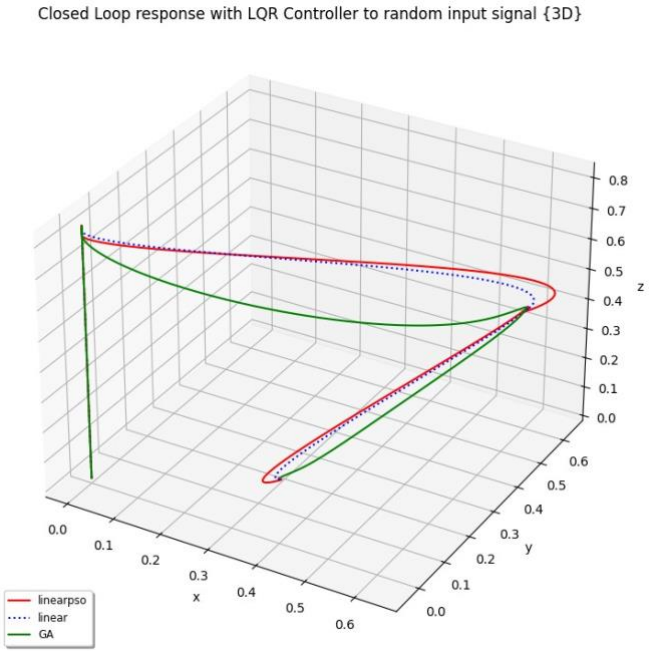

Fig. 4. Trajectory plan for both algorithms.

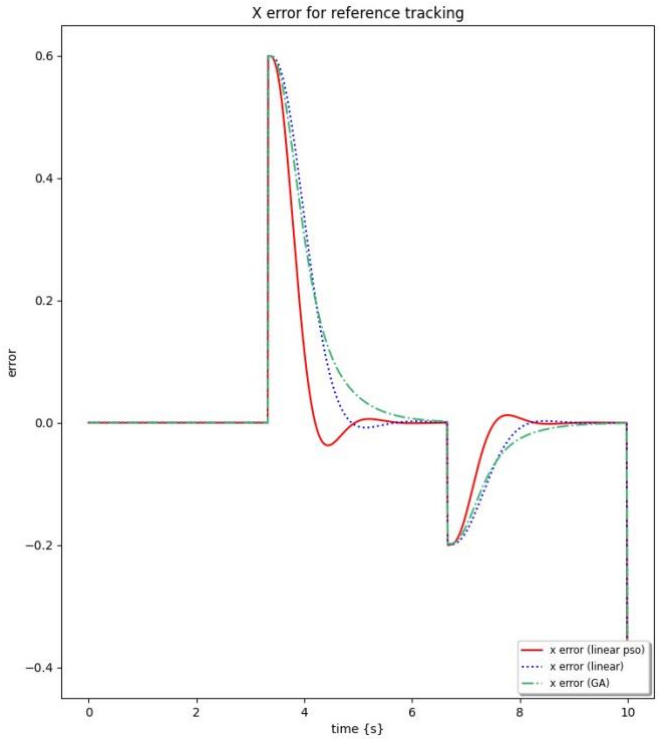

Fig. 5. Performance in position x.

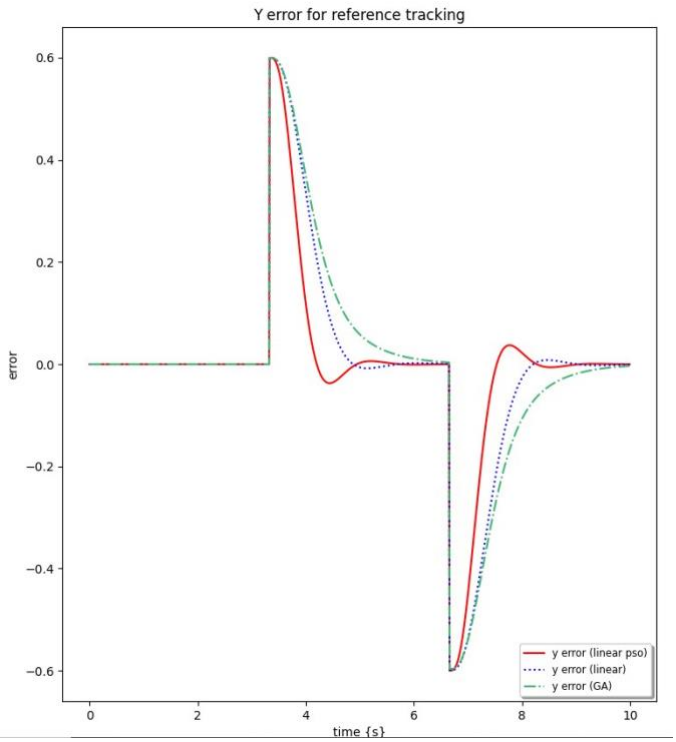

Fig. 6. Performance in position y.

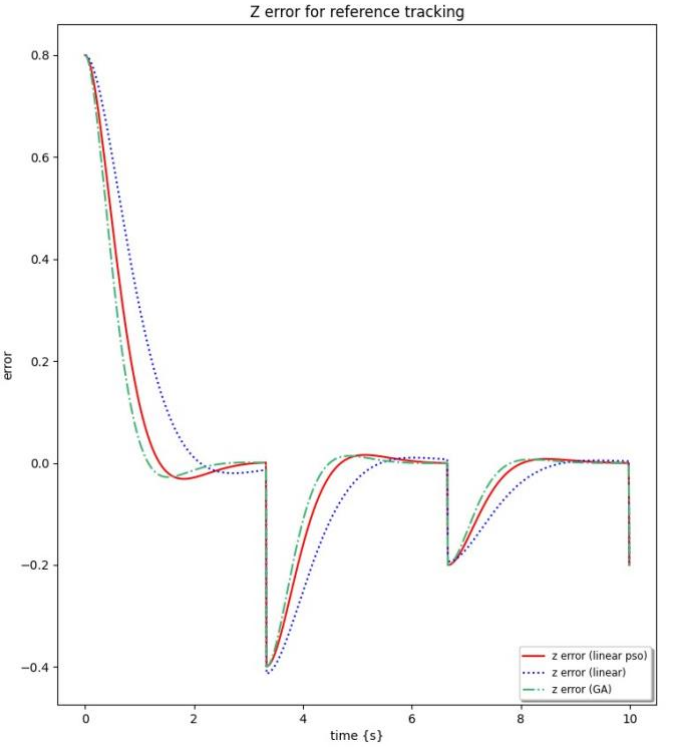

Fig. 7. Performance in position z.

Fig. 4 shows the 3-D trajectories for a three-waypoint mission. Figs. 5–7 show the x, y, and z position tracking responses allowing a detailed assessment of error and settling behavior at each waypoint. At the first waypoint (vertical ascent to 0.8 m) both algorithms exhibit z-axis overshoot before correcting; GA stabilises slightly earlier. The second leg involves substantial lateral displacement: PSO settles faster in x and y, though with increased overshoot. During the third leg PSO again outperforms GA in x/y settling, while GA consistently achieves

earlier z-axis steady-state.

The waypoint graph reveals smoother inter-waypoint transitions under PSO. The PSO trajectory exploits the quadrotor's momentum for fluid motion, whereas the GA trajectory is more abrupt and point-to-point. PSO's sub-second convergence is a meaningful practical advantage for deployment on a low-cost microcontroller-based quadrotor.

V. Conclusion

This paper compared PSO and GA for automated LQR weighting matrix selection for quadrotor trajectory tracking. PSO consistently outperformed GA: lower cost (72.83 vs. 74.94), sub-second convergence (vs. 77–92 s for GA), and smoother trajectories with faster x/y settling. GA showed marginally better z-axis settling. Future work should explore hybrid PSO-GA strategies and validate controllers on physical hardware.

References

[1] L. Martins, Cardeira, and P. Oliveira, "Linear quadratic regulator for trajectory tracking of a quadrotor," IFAC-PapersOnLine, vol. 52, no. 12, pp. 176–181, 2019, doi: 10.1016/j.ifacol.2019.11.195.

[2] S. Abdelhay and A. Zakriti, "Modeling of a quadcopter trajectory tracking system using PID controller," Procedia Manuf., vol. 32, pp. 564–571, 2019.

[3] F. Sabatino, "Quadrotor control: Modeling, nonlinear control design, and simulation," M.S. thesis, KTH Royal Inst. Technol., Stockholm, Sweden, 2015.

[4] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in Proc. IEEE/RSJ IROS, 2011, pp. 2668–2673.

[5] B. Douglas. (2019). State space, part 4: What is LQR control? [Online]. Available: https://www.youtube.com/watch?v=E_RDCFOlJx4

[6] B. Douglas. (2023). Why the Riccati equation is important for LQR control. [Online]. Available: https: //www.youtube.com/watch?v=ZktL3YjTbB4

[7] K. Alexis. (n.d.). LQR control. [Online]. Available: http://www.kostasalexis.com/lqr-control.html

[8] J. Gu and D. Fang, "Genetic algorithm based LQR control for AGV path tracking," J. Phys. Conf. Ser., vol. 1952, no. 3, 2021.

[9] P. Pawłowski and S. Konatowski, "Linear controller design with PSO for UAV trajectory tracking," Proc. SPIE, vol. 11442, 2020.

[10] S. Howimanporn et al., "Design and implementation of PSO based LQR control for inverted pendulum," in Proc. IEEE/SICE SII, 2016.

[11] A. Tam. (2021). A gentle introduction to particle swarm optimization. [Online]. Available: https://machinelearningmastery.com

[12] R. Tedrake. (2023). Underactuated robotics. [Online]. Available: https://underactuated.mit.edu/lqr.html

Uchenna, C. Onyema (Member, IEEE) is a recent PhD graduate of the University of Derby. He holds a background in Electronic Engineering and a master's degree (Distinction, 2018) from the University of Derby. He has accumulated experience collaborating with national research centers, contributing to cross-disciplinary projects at the intersection of academia and industry, teaching engineering core disciplines as an Associate Lecturer. His current research interests include advanced control systems, autonomous localization, and the application of machine/deep learning to complex engineering problems. Mr. Onyema is an active Member of IEEE and IET UK and a recipient of multiple awards for academic and research impact.

Temidayo Ayanda (Member, IET) received his BEng degree in Electrical and Electronic Engineering from the University of Derby, Derby, Uk in 2023. His research interests include Machine Learning algorithms and control systems. Since 2025, he has been at BAE Systems, where he is currently a Systems Engineer. His work focuses on Complex Electronic Hardware.