

Swarm Intelligence tuning of PID Position Controllers for a dual-axis Solar tracker: A Statistical Comparison of Particle Swarm Optimization, the firefly Algorithm and Cuckoo Search

Tauqeer Ahmad¹, Zaigham Abbas², Muhammad Zeeshan¹, and Syed Ali Hussain²

¹ University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, Pakistan tauqeer.ahmed443@gmail.com mzeeshanshafi2001@gmail.com

² Azerbaijan Universities of Architecture and Construction, Baku, Azerbaijan zaighamji@gmail.com syedali6160@gmail.com

Abstract. Accurate sun tracking boosts the energy yield of photovoltaic (PV) systems; in a dual-axis tracker, the accuracy is governed by the position loops driving the azimuth and tilt DC motors. In this study, we formulate the tuning of the proportional, integral and derivative gains of these loops as a bounded optimization problem, and compare three swarm-intelligence algorithms on it: particle swarm optimisation (PSO), the firefly algorithm (FFA) and cuckoo search (CSA). The plant is a permanent-magnet DC servo with an 800:1 gear reducer, modelled in MATLAB/Simulink, and we aim for the weighted overshoot, steady-state-error, rise-time and settling-time criterion of Gaing with a weighting factor of 0.5. Each algorithm was run 100 independent times. The fixed-gain loop settled in 3.95 s with a 2.20 s rise time. The three algorithms found the same best objective value of 0.25294, but their reliability varied: the mean (standard deviation) objective was 0.26565 (0.04152) for PSO, 0.25555 (0.00694) for FFA and 0.25299 (0.0000526) for CSA. Welch tests on the run statistics reveal that CSA achieves a lower mean objective than both PSO ($p = 0.003$) and FFA ($p < 0.001$). Using the mean tuned gains, the closed loop reached the set-point in around 0.3 s without visible overshoot. The results suggest that, for this class of servo-tuning problem, the strength of cuckoo search lies in run-to-run consistency, rather than a better optimum, which is relevant when the tuning is repeated on embedded hardware without supervision.

Keywords: Solar tracking . Dual-axis tracker . PID tuning . Particle Swarm Optimization . Firefly Algorithm . Cuckoo Search

1 Introduction

Photovoltaic generation has become one of the cheapest sources of new electricity, and the cost per unit of energy continues to fall as module efficiency improves [1]. The energy a PV panel accumulates depends on the angle of incidence of the sun's rays with respect to the panel normal, which varies continuously over the day, and over the year. Single-axis and dual-axis trackers reduce this angle, and reported gains over fixed, optimally

tilted panels are roughly between 20% and 40% depending on climate and latitude [2–4]. In south-eastern Turkey, for example, a two-axis tracker collected 34.6% more daily energy than a fixed south-facing panel [5]. In high-irradiance regions like Pakistan, tracking can be an inexpensive way to raise the output of installed capacity. Trackers are classified by structure (single or dual axis) and by control strategy (open loop, closed loop or hybrid) [2,6]. Single-axis designs are cheap but cannot follow the seasonal change of the solar elevation, whereas dual-axis designs follow both azimuth and elevation at a higher one-off cost [4,7]. Whatever the strategy, the motion is generated by DC-motor servo drives, and the tracking accuracy is ultimately limited by how well their position loops are tuned. Rubio et al. [8] used PI controllers tuned by classical rules on a two-axis tracker; controllers are simple, but classical rules assume a process model rarely exact and give no guarantee of good transient behaviour under the slow, high-inertia dynamics of a panel drive. The proportional–integral–derivative (PID) controller remains the dominant industrial controller, and more than 400 tuning rules have been catalogued [9,10]. Surveys nevertheless report that a large share of industrial loops run poorly tuned or in manual mode [11]. Population-based metaheuristics avoid the need for an exact model because they evaluate candidate gains directly on a simulation, and have been applied to PID tuning with ant colony optimization [12], PSO [13], the firefly algorithm [14] and others, usually with favourable comparisons against Ziegler–Nichols and similar rules. Multi-objective formulations have also been reviewed [15]. In earlier work, the first author compared PSO, genetic algorithms and simulated annealing, with and without Nelder–Mead refinement, for the speed control of a DC motor [16]. Two gaps motivate the present study. First, most metaheuristic PID studies address speed control or single-axis tracking, and position control of a dual-axis tracker is understudied. Secondly, many comparisons report a single best run per algorithm, which says little about how reliably an algorithm reaches a good controller when run once on an embedded target. Because metaheuristics are stochastic, and no algorithm is best on all problems [17], conclusions should be based on repeated runs and statistical tests [18]. The contributions of this paper can be summarized as follows.

- 1) A bounded three-variable optimization problem, based on a time-domain criterion combining overshoot, steady-state error, rise time, and settling time, is formulated for the PID tuning of the azimuth and tilt position drives of a dual-axis tracker.

- 2) The Particle Swarm Optimization (PSO), Firefly Algorithm (FFA) and Cuckoo Search Algorithm (CSA) are adapted to the problem with a common decision vector and tested in 100 independent runs each.

- 3) The algorithms are compared in terms of the distribution of the objective, the transient characteristics of the resulting loops, and convergence properties using Welch tests on the run statistics, with discussion of the practical significance of the differences.

Section 2 contains related work, Section 3 describes the tracker and motor models, Section 4 defines the optimisation problem and algorithms, Section 5 describes the experimental protocol, Section 6 reports results, Section 7 discusses and Section 8 concludes.

2 Related work

This study analyzes Particle Swarm Optimization, the Firefly Algorithm, and Cuckoo Search to tune PID position controllers in dual-axis solar trackers. It uses statistical analysis to uncover efficient optimization options for increasing solar tracking accuracy and control performance.

Lee et al. [6] and Mousazadeh et al. [2] have surveyed tracking principles, classifying trackers as passive, time-based (astronomical) or active (sensor-based). Passive trackers exploit the thermal expansion of a low-boiling-point fluid to unbalance the frame [19], and require no electronics but are slow and inaccurate. Time-based trackers calculate the solar position based on the date, time and location and command the drives according to this, while active trackers use light sensors to close the loop on the actual sun position [20]. Hybrid schemes use an astronomical model for coarse positioning and sensor feedback for correction; the strategy of Rubio et al. [8] switches between an open-loop solar model and closed-loop error correction. Yao et al. [7] have built a multipurpose dual-axis tracker with two tracking strategies, while Nsengiyumva et al. [4] have reviewed recent advances and remaining challenges. Maximum power point tracking (MPPT), as reviewed in [21], is complementary to mechanical tracking: it optimizes the electrical operating point, while the tracker optimizes the irradiance received. Gaing [13] has presented a PSO-based PID design for an automatic voltage regulator, coupled with the time-domain objective used in this paper, and has demonstrated improved step responses over a genetic algorithm. Varol and Bingul [12] have tuned PID controllers with ant colony optimization and have reported faster settling and smaller overshoot than Ziegler–Nichols, internal model control or iterative feedback tuning. Kumanan and Nagaraj [14] have applied the firefly algorithm to a flow process. Martins [22] has exhibited how ITAE-optimal PID tuning can be performed using the MATLAB optimization toolbox. Cuckoo search, developed by Yang and Deb [2,24], combines Lévy-flight exploration and a nest-abandonment move and has exhibited great performance on engineering design problems; the heavy-tailed step distribution of Lévy flights is known to be beneficial when escaping local minima [25]. Reynoso-Meza et al. [15] argue that controller tuning is inherently multi-objective, supporting the use of composite criteria such as that of Gaing [13]. Comparative studies in this area frequently report the best result out of several runs. Derrac et al. [18] suggest reporting the distribution of results in many independent runs, coupled with statistical tests, which is what we have done here.

Overall, the work emphasizes the value of swarm intelligence algorithms in optimizing PID controllers for dual-axis solar tracking systems. Its comparative findings constitute the foundation for selecting appropriate optimization approaches to improve tracking performance and control precision.

3 System Description and Modeling

3.1 Dual-axis tracker Architecture

The tracker (Fig. 1) has two independent position loops, one for the azimuth angle and one for the tilt angle. In each loop a reference-generation block produces the desired

angle from the solar-position equations, corrected by light-sensor feedback, and a PID controller drives a DC motor through a gear reducer. Load disturbances, such as wind torque on the panel, enter at the motor shaft. To limit energy consumption, the reference is updated every 30 min. Because the two axes use identical motors and gear trains and their dynamics are decoupled at the slow speeds involved, the same plant model and the same tuning procedure apply to both loops; the results reported below therefore hold for each axis.

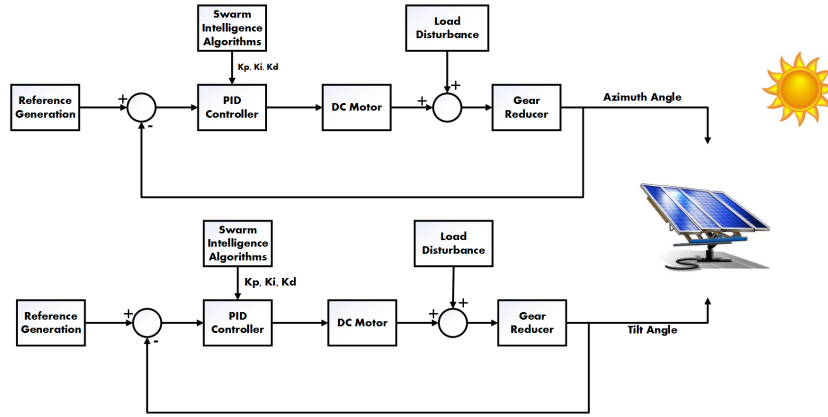


Fig. 1. Block diagram of the dual-axis sun-tracking system. Each axis has an independent PID position loop whose gains K_p , K_i and K_d are supplied by the swarm-intelligence tuner.

3.2 DC motor model

Each axis is driven by a permanent-magnet DC motor (Fig. 2). With armature voltage V_a , armature current i , rotor angle θ , armature resistance R , armature inductance L , torque constant K_e , back-EMF constant K_t , rotor inertia J , viscous friction coefficient b and load torque T_L , the electrical and mechanical equations are

$$L di/dt = V_a - Ri - K_t d\theta/dt \quad (1)$$

$$J(d^2\theta/dt^2) = K_e i - b d\theta/dt - T_L \quad (2)$$

Taking Laplace transforms with zero initial conditions and $T_L = 0$ gives the transfer function from armature voltage to rotor angle

$$\Theta(s)/V_a(s) = K_e / (s[(Ls + R)(Js + b) + K_e K_t]) \quad (3)$$

The parameter values are listed in Table 1. The rotor angle is reduced to the panel angle through an 800:1 gear train, and in the reference (fixed-gain) configuration the angular error is amplified by an error detector (gain 0.001), an operational amplifier (gain 104) and a unity-gain servo amplifier before reaching the motor (Fig. 3).

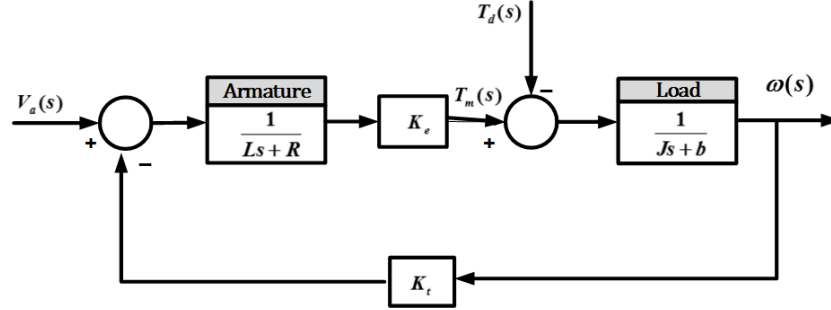


Fig. 2. Block diagram of the armature-controlled DC motor.

Table 1. Parameters of the DC servo motor used for both axes.

Symbol	Description	Value
K_e	Torque constant	0.01125 N m A ⁻¹
K_t	Back-EMF constant	0.0125 V s rad ⁻¹
R	Armature resistance	6.25
L	Armature inductance	5.6
J	Rotor moment of inertia	1×10^{-6} kg m ²
b	Viscous friction coefficient	1×10^{-6} N m s rad ⁻¹
N	Gear ratio	800:1

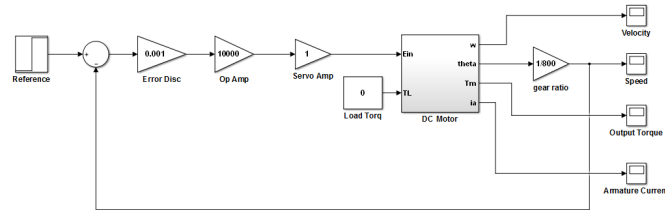


Fig. 3. Simulink model of one tracker axis in its fixed-gain configuration. In the tuned configuration the PID controller acts on the angular error ahead of the servo amplifier.

4 PID tuning as an optimization Problem

4.1 Controller and decision vector

The ideal (non-interacting) PID structure is used:

$$C(s) = K_p + K_i/s + K_d s \quad (4)$$

Each candidate solution of every algorithm is the vector $x = [K_p, K_i, K_d]^T$, bounded component-wise by $x_{lmin} \leq x_l \leq x_{lmax}$, $l = 1, 2, 3$. The bounds were $K_p \in [\min, 10]$, $K_i \in [\min, \max]$ and $K_d \in [104, \max]$.

4.2 Objective function

Integral criteria such as IAE, ISE and ITAE weight the whole error history but don't let the designer trade overshoot against speed directly. Following Gaing [13], each candidate is scored on four step-response quantities, the maximum overshoot M_p , the steady-state error E_{ss} , the rise time t_r and the settling time t_s :

$$\min_x J(x) = (1 - e^{\lambda})(M_p + E_{ss}) + e^{\lambda}(t_s - t_r) \quad (5)$$

The weighting factor λ shifts the emphasis: $\lambda > 0.7$ penalises overshoot and steady-state error more heavily, while $\lambda < 0.7$ favours fast rise and settling [13]. A balanced value $\lambda = 0.5$ is used throughout. Each evaluation of J requires one closed-loop step simulation of the Simulink model.

4.3 Particle swarm optimization

PSO [26] moves a swarm of n particles through the search space. Particle k has position x_k and velocity v_k ; positions are initialized uniformly within the bounds,

$$x(k, l)^0 = x_l^{min} + r(x_l^{max} - x_l^{min}), k = 1, \dots, n, l = 1, 2, 3 \quad (6)$$

where $r \in U(0, 1)$. At iteration t the velocity and position are updated by

$$v(k, l)^t = v(k, l)^{t-1} + c_1 r_1 (p(k, l)^t - x(k, l)^{t-1}) + c_2 r_2 (g_l^t - x(k, l)^{t-1}) \quad (7)$$

$$x(k, l)^t = x(k, l)^{t-1} + v(k, l)^t \quad (8)$$

where w is the inertia weight [27], c_1 and c_2 are the cognitive and social coefficients, $r_1, r_2 \in U(0, 1)$, p_k is the best position found by particle k and g the best position found by the swarm. Positions leaving the bounds are clipped. The algorithm stops when the iteration limit is reached or the change in the best objective falls below the tolerance.

Firefly algorithm In the firefly algorithm [28,29] each firefly is a candidate x_i , and its brightness is determined by $J(x_i)$. A firefly moves towards every brighter firefly with an attractiveness that decays with distance,

$$A(r_{ij}) = A_0 e^{(-r_{ij}^2)}, r_{ij} = x_i - x_j \quad (9)$$

$$x_i = x_i + A_0 e^{(-r_{ij}^2)}(x_j - x_i) + (r - 0.5) \quad (10)$$

where A_0 is the attractiveness at zero distance, γ is the light-absorption coefficient, scales the random step and $r \sim U(0, 1)$. Because attraction weakens with distance, the population tends to split into sub-swarms that each converge on a local optimum, which gives the algorithm a built-in multimodal search.

Cuckoo Search Cuckoo search [24,25] models the brood parasitism of cuckoos. Each nest holds one solution z_i . New solutions are generated by Lévy flights,

$$z_i^{(t+1)} = z_i^t + \alpha \text{Lévy}() \quad (11)$$

where the step lengths follow a heavy-tailed Lévy distribution generated with Mantegna's algorithm [30]. A randomly chosen nest j is replaced if the new solution is better. A fraction p_a of the worst nests is then abandoned and rebuilt by a biased random walk between two randomly permuted nests,

$$z_i^{(t+1)} = z_i^t + r H(p_a - \text{rand}) (z_{(1(i))}^t - z_{(2(i))}^t) \quad (12)$$

where H is the Heaviside function, $r \sim U(0, 1)$, and 1, 2 are random permutations of the nest indices. The few parameters to set (number of nests and p_a) make cuckoo search easy to configure.

5 Experimental Setup

5.1 Simulation Environment

All simulations were carried out using MATLAB R2014a with Simulink in a personal computer that has an Intel Core i3-4005U processor with 1.70 GHz. The same simulation environment and evaluation procedure were employed for the Particle Swarm Optimization (PSO), Firefly Algorithm (FFA), and Cuckoo Search Algorithm (CSA) to enable a fair comparison between them in tuning the PID position controller for the dual-axis solar tracking system. The three algorithms were assessed using their respective parameter configurations as summarized in Table 2. This configuration entails the population size and a maximum number of iterations as well as the algorithm dependent control parameters and numerical tolerances. The PSO algorithm was implemented with a population size of 100 and a maximum number of iterations of 200 while FFA was executed with 50 fireflies and 100 iterations. On the other hand, CSA was carried out with a population size of 100, but the maximum iteration number has to be verified from the implemented setting before the table.

5.2 Optimization Algorithm Configuration

The parameter settings were selected to define the search behavior of each optimization algorithm. In PSO, the cognitive coefficient (c_1) was set to 0.5. On the other hand, FFA involved the randomization parameter (α), attractiveness parameter (A_0) and light-absorption coefficient (γ) which were set to 0.75, 0.75 and 1.0 respectively. In CSA, the discovery rate was set to 0.75. Parameters that were not applicable were marked with dash as in Table 2. A constraint tolerance of (10^{-6}) was specified for all the three algorithms to maintain a fair numerical feasibility criterion. Function tolerance was also set to (10^{-6}) for PSO and FFA and a corresponding CSA function tolerance should be verified. Tolerances define numerical stopping or feasibility criteria; they do not, by themselves, define the accuracy or superiority of an optimization algorithm.

Table 2. Parameter settings of the optimization Algorithms.

Parameter	PSO	FFA	CSA
Population size (particles / fireflies / nests)	100	50	100
Maximum iterations	200	100	50
Cognitive coefficient c_1	0.5	—	—
Randomisation parameter	—	0.75	—
Attractiveness A_0	—	0.75	—
Absorption coefficient	—	1.0	—
Discovery rate p_a	—	—	0.75
Constraint tolerance	106	106	106
Function tolerance	106	106	106

5.3 Independent Runs and Data Collection

Each algorithm was run 100 times independently using different random seeds to take care of the stochasticity of the optimization algorithms. Independent runs provide a distribution of optimization outcomes, as opposed to searching for a single optimization run that may depend on the initial population and the random search decisions during the search process. For each run, the best PID gain vector identified by the algorithm and its corresponding objective-function value were recorded. The associated time domain performance measures namely settling time ((t_s)), rise time ((t_r)), and maximum percentage overshoot ((M_p)) were also gathered. These were to assess the optimization objective as well as the response of the controller. For each algorithm, the minimum, maximum, mean, variance, and standard deviation were calculated across the 100 runs. The minimum and maximum describe the observed range of outcomes while the mean is the average performance. Variance and standard deviation quantify the scatter of the results and provide information about the optimization process. A small standard deviation implies that there is less scatter around the mean objective value which does not necessarily mean a better mean objective value.

6 Results and Discussion

6.1 Fixed-gain loop

The step response of one axis in its fixed-gain configuration is shown in Fig. 4. and summarized in Table 3. The response is overdamped with rise and settling time of 2.20 and 3.95 s, respectively. This sluggishness is caused by the large reflected inertia of the panel via the 800:1 gear train and suggests that a compensator is necessary to improve settling and reject disturbance torques.

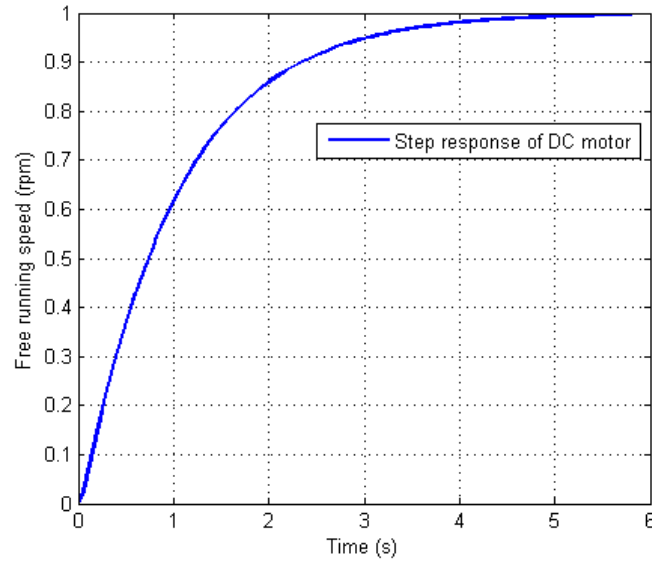


Fig. 4. Normalized step response of one tracker axis with the fixed-gain amplifier (no PID compensation).

Table 3. Step-response characteristics of the fixed-gain loop.

Quantity	Value
Rise time t_r (s)	2.199
Settling time t_s (s)	3.953
Maximum overshoot M_p (%)	0
Steady-state error E_{ss} (%)	0.5

6.2 Run-to-run distribution of the objective

Fig. 5 displays the final objective of each of the 100 runs. PSO consistently finds the best objective, but occasionally gets trapped in mediocre solutions with objective as high as 0.47, producing a handful of large spikes. FFA never gets worse than 0.288 but wanders off the optimum more frequently than PSO, and CSA remains within 0.0004 of the optimum in all runs; note that the y-scale of Fig. 5(c) is about 500 times smaller than the one of Fig. 5(a).

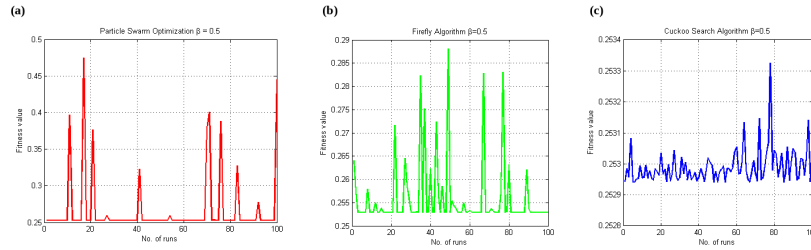


Fig. 5. Final objective value in each of 100 independent runs: (a) PSO, (b) FFA, (c) CSA ($\beta = 0.5$). Note the different vertical scales.

Table 4. Statistics of the tuned gains and objective value over 100 independent runs.

Algorithm	Quantity	Min	Max	Mean	Std. dev.
PSO	Kp	3.3152	10.0000	9.5130	1.4438
	Ki	0.0352	8.1810	7.4920	1.8110
	Kd	0.000100	0.007728	0.000223	0.000866
	J	0.252941	0.474865	0.265647	0.041524
FFA	Kp	6.7627	10.0000	9.7208	0.7232
	Ki	0.0051	8.2169	7.4405	2.2214
	Kd	0.000100	0.000100	0.000100	0
	J	0.252941	0.288091	0.255549	0.006944
CSA	Kp	9.99998	10.0000	10.0000	0.0000015
	Ki	8.0531	8.1225	8.1138	0.0099
	Kd	0.000100	0.000100	0.000100	0
	J	0.252941	0.253326	0.252986	0.0000526

These trends are quantified in Table 4, which lists the distributions of the objective function for the three methods. All algorithms produce solutions with the same minimum objective of 0.252941 (up to six decimal places), so that none finds a consistently better controller than the others. However, the three methods differ considerably in their reliability; the mean objective decreases from 0.26565 (PSO) to 0.25555 (FFA) to 0.25299 (CSA) and their standard deviation decreases sharply from 0.0415 to 0.0069 to 0.0000526. Thus, the spread of CSA is about 790 times smaller than the spread of PSO

and 130 times smaller than the spread of FFA. Similar trends are seen in the distribution of the gains; CSA produces gain values closer to their theoretical optimum, $K_p = 10.000$ and $K_i = 8.11$, in nearly every run while PSO and FFA occasionally find solutions with extreme K_p , K_i far from these values.

In every algorithm the best K_p is at its maximum allowable value of 10 and the best K_d is at its minimum allowable value of 104. Thus, the best solution to the optimization problem as posed corresponds to a PI controller with a vanishingly small derivative component. This result is not surprising given the inertia-dominated character of the plant dynamics and suggests that the optimization problem benefits from a wider range of K_p around the optimum (Section 7).

6.3 Statistical Comparison

Table 5 reports the results of the pairwise Welch tests on the mean objective. CSA has a lower mean than PSO ($t = 3.05$, $p = 0.003$) and FFA ($t = 3.69$, $p < 0.001$), and FFA has a lower mean than PSO ($t = 2.40$, $p = 0.018$). The differences are all significant at the 5% level after Holm correction for three comparisons (*the adjusted* $p = 0.006, 0.001$ and 0.018), and in fact are fairly large, albeit with small sample size; the effect sizes are in the range of $d = 0.34 - 0.52$ according to Cohen's convention, because the differences are mostly driven by a few bad runs rather than a systematic shift of the distributions. Because the objective is bounded below and its distribution is highly skewed, a rank-

Table 5. Pairwise Welch tests on the mean objective value (100 runs per algorithm).

Comparison	Mean difference	t	p	Holm-adjusted p	Cohen's d
PSO CSA	0.012661	3.05	0.0029	0.0059	0.43
FFA CSA	0.002563	3.69	0.0004	0.0011	0.52
PSO FFA	0.010098	2.40	0.0182	0.0182	0.34

based non-parametric test on the raw data (Wilcoxon rank-sum) would be a more reliable assessment; the conclusion however is likely to be similar given the large differences between the variances.

6.4 Transient metrics

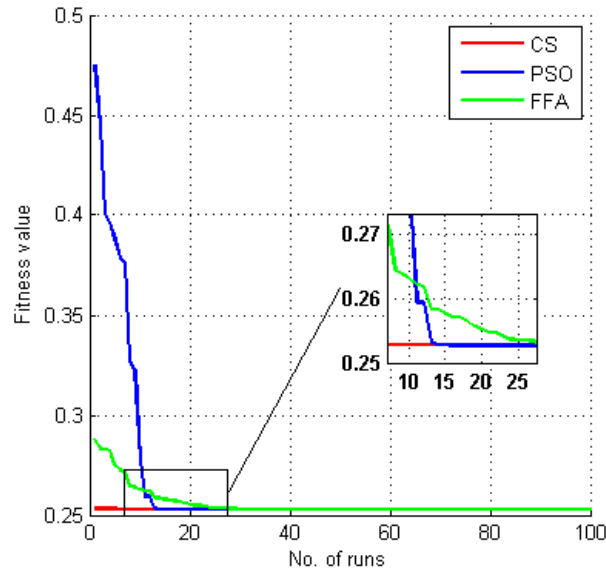
The transient response metrics for the tuned loops are collected in Table 6 for all three algorithms. The best runs of all three algorithms have rise time of 0.16-0.19 s and settling time of 0.24-2.3 s, an order of magnitude better than the fixed-gain loop. The means are inflated by a few runs with poor performance; PSO has the lowest mean rise time (0.19 s) and overshoot (0.18%), while FFA and CSA have a few runs with large overshoot or excessively long rise and settling time. For FFA, the mean rise time (53.3 s) is unexpectedly higher than the mean settling time (7.8 s), which is impossible for a second-order system; in fact, these runs did not converge within the simulation horizon. In general, the results are consistent with those of Table 4: while the controllers found in the best runs are similar, CSA has a better chance to avoid poor runs.

Table 6. Transient metrics of the tuned loops over 100 runs.

Algorithm	Metric	Min	Max	Mean	Std. dev.
PSO	$ts(s)$	0.2531	21.8513	0.4789	1.9812
	$tr(s)$	0.1604	1.0098	0.1915	0.1083
	$M_p(\%)$	0	0.1084	0.0018	0.0122
FFA	$ts(s)$	2.3067	209.0702	7.7615	27.1238
	$tr(s)$	0.1933	362.9471	53.2572	102.6571
	$M_p(\%)$	0	23.6498	3.8603	4.1450
CSA	$ts(s)$	0.2361	27.0122	1.2928	3.3243
	$tr(s)$	0.1555	3.5677	0.3593	0.4807
	$M_p(\%)$	0	90.9051	4.6510	13.1787

6.5 Convergence Behavior

The convergence of the algorithms in representative runs is displayed in Fig. 6. CSA starts near the optimum and converges steadily to it. PSO has a poor initialization (final objective above 0.47) but drops rapidly in the first ten to fifteen iterations and converges to the same optimum. FFA starts closer to the optimum than PSO but converges to it slower. The acceleration of CSA relative to PSO in the initial iterations is explained by the Lévy-flight distribution of its steps, which have a long tail extending to large values, thus exploring the box in the very first iterations.

**Fig. 6.** Convergence of the best objective value for CSA, PSO and FFA. The inset magnifies iterations 10–25.

6.6 Closed-loop Response

The step response of the three controllers corresponding to the mean gains of Table 4 are compared in Fig. 7. All three loops reach the desired pointing within approximately 0.3 s, compared to 3.95 s of the fixed-gain loop, with no visible overshoot. The inset displays the responses on a smaller y-scale; they are identical up to the third decimal place and differ only in the speed of response. In practice, all three tuned loops are indistinguishable in performance; the choice between them depends on the reliability of their respective tuning algorithms. The principal result is that CSA reaches the best known controller

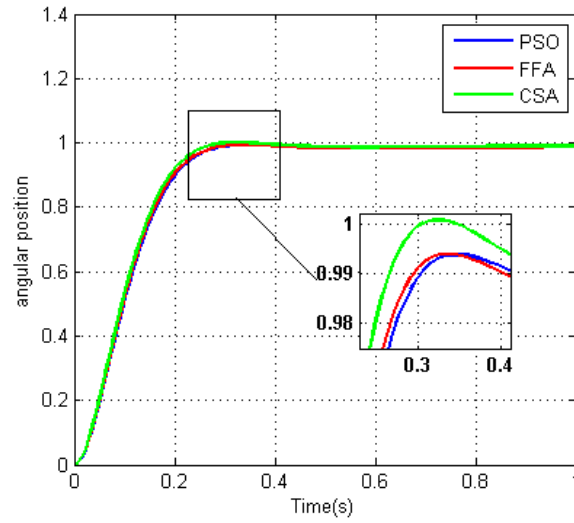


Fig. 7. Step response of the PID-compensated tracker axis using the mean gains of each algorithm. The inset magnifies the region near the set-point.

in essentially every run, while PSO and FFA do so in most but not all runs. For an offline design carried out once by an engineer, this difference is minor, because one can run PSO several times and keep the best result. It is important when tuning is repeated automatically, for example when an embedded tracker re-identifies its plant after mechanical wear or a change of panel load and retunes itself without supervision. In that setting a single unreliable run produces a poorly tuned loop, and the algorithm with the narrowest run-to-run spread is preferable. Two features of CSA plausibly explain its behaviour. The Lévy-flight step mixes many small moves with occasional long jumps, so the population samples the whole box early and is less likely to collapse prematurely onto a sub-optimal region. The abandonment step then continually replaces the worst nests, which acts as a restart mechanism. PSO, by contrast, can stall when the swarm contracts around a local best before the boundary optimum is found, which matches the isolated high-objective runs in Fig. 5(a). FFA's distance-dependent attraction divides the

population into sub-swarms, which limits large failures but leaves more runs slightly away from the optimum. Boundary optimum. All algorithms place K_p at its upper limit and K_d at its lower limit. A boundary optimum suggests that the search box, not the plant, determines the solution in those coordinates. Widening the K_p range should lower the objective further, but larger proportional gains increase motor current and actuator saturation, which the linear model does not capture. A constrained formulation that bounds peak current or control effort would make the choice of range explicit rather than arbitrary. The algorithms used different population sizes and iteration limits (Table 2), so their budgets of objective evaluations were not equal. A fixed budget of function evaluations per run would allow convergence speed to be compared directly. This does not affect the reliability conclusion, since PSO had the largest budget and still showed the widest spread. The results are simulation-based and use a linear motor model without backlash, Coulomb friction, saturation or sensor noise. Only one objective weighting ($= 0.5$) was examined. Both axes were treated with the same plant model, so axis-specific effects such as the different gravitational loading of the tilt axis are not represented. Finally, consistent with the no-free-lunch theorem, the superiority of CSA on this problem does not by itself imply superiority on other engineering problems.

7 Conclusion

The problem of PID tuning of the azimuth and tilt loops of a dual-axial solar tracker was posed as a bounded optimization problem and tuned using particle swarm optimization, the firefly algorithm and cuckoo search each run 100 times. The tuned loops reach the set-point in approximately 0.3 s compared with 3.95 s for the fixed-gain loop. The three algorithms found the same best controller, however, cuckoo search found the best objective in all but one run. In addition, while using a much smaller standard deviation per run, cuckoo search had a mean objective significantly lower than PSO and FFA. The firefly algorithm was more reliable than PSO, however, since cuckoo search also had fewer parameters to be set, cuckoo search is the method most appropriate for the automatic and repeated tuning of embedded tracker controllers.

References

1. G.K. Singh, Solar power generation by PV (photovoltaic) technology: a review, *Energy* 53 (2013) 1–13.
2. H. Mousazadeh, A. Keyhani, A. Javadi, H. Mobli, K. Abrinia, A. Sharifi, A review of principle and sun-tracking methods for maximizing solar systems output, *Renew. Sustain. Energy Rev.* 13 (2009) 1800–1818.
3. E.F. Camacho, M. Berenguel, F.R. Rubio, D. Martínez, *Control of Solar Energy Systems*, Springer, London, 2012.
4. W. Nsengiyumva, S.G. Chen, L. Hu, X. Chen, Recent advancements and challenges in solar tracking systems (STS): a review, *Renew. Sustain. Energy Rev.* 81 (2018) 250–279.
5. M. Kacira, M. Simsek, Y. Babur, S. Demirkol, Determining optimum tilt angles and orientations of photovoltaic panels in Sanliurfa, Turkey, *Renew. Energy* 29 (2004) 1265–1275.
6. C.Y. Lee, P.C. Chou, C.M. Chiang, C.F. Lin, Sun tracking systems: a review, *Sensors* 9 (2009) 3875–3890.

7. Y. Yao, Y. Hu, S. Gao, G. Yang, J. Du, A multipurpose dual-axis solar tracker with two tracking strategies, *Renew. Energy* 72 (2014) 88–98.
8. F.R. Rubio, M.G. Ortega, F. Gordillo, M. López-Martínez, Application of new control strategy for sun tracking, *Energy Convers. Manag.* 48 (2007) 2174–2184.
9. A. O'Dwyer, *Handbook of PI and PID Controller Tuning Rules*, 3rd ed., Imperial College Press, London, 2009.
10. K.J. Åström, T. Hägglund, The future of PID control, *Control Eng. Pract.* 9 (2001) 1163–1175.
11. D.B. Ender, Process control performance: not as good as you think, *Control Eng.* 40 (10) (1993) 180–190.
12. H.A. Varol, Z. Bingul, A new PID tuning technique using ant algorithm, in: *Proc. 2004 American Control Conference*, Boston, 2004, pp. 2154–2159.
13. Z.-L. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, *IEEE Trans. Energy Convers.* 19 (2004) 384–391.
14. D. Kumanan, B. Nagaraj, Tuning of proportional integral derivative controller based on firefly algorithm, *Syst. Sci. Control Eng.* 1 (2013) 52–56.
15. G. Reynoso-Meza, X. Blasco, J. Sanchis, M. Martínez, Controller tuning using evolutionary multi-objective optimisation: current trends and applications, *Control Eng. Pract.* 28 (2014) 58–73.
16. M.M. Sabir, J.A. Khan, Optimal design of PID controller for the speed control of DC motor by using metaheuristic techniques, *Adv. Artif. Neural Syst.* 2014 (2014) 126317.
17. D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.* 1 (1997) 67–82.
18. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.* 1 (2011) 3–18.
19. M.J. Clifford, D. Eastwood, Design of a novel passive solar tracker, *Sol. Energy* 77 (2004) 269–280.
20. I. Stamatescu, I. Făgărășan, G. Stamatescu, N. Arghira, S.S. Iliescu, Design and implementation of a solar-tracking algorithm, *Procedia Eng.* 69 (2014) 500–507.
21. A.R. Reisi, M.H. Moradi, S. Jamasb, Classification and comparison of maximum power point tracking techniques for photovoltaic system: a review, *Renew. Sustain. Energy Rev.* 19 (2013) 433–443.
22. F.G. Martins, Tuning PID controllers using the ITAE criterion, *Int. J. Eng. Educ.* 21 (2005) 867–873.
23. X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: *Proc. World Congress on Nature Biologically Inspired Computing (NaBIC 2009)*, IEEE, 2009, pp. 210–214.
24. X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, *Int. J. Math. Model. Numer. Optim.* 1 (2010) 330–343.
25. I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, *J. Comput. Phys.* 226 (2007) 1830–1844.
26. J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proc. IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.
27. Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: *Proc. IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.
28. X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Frome, 2008.
29. X.-S. Yang, Firefly algorithms for multimodal optimization, in: *Stochastic Algorithms: Foundations and Applications (SAGA 2009)*, LNCS 5792, Springer, 2009, pp. 169–178.
30. R.N. Mantegna, Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes, *Phys. Rev. E* 49 (1994) 4677–4683.