Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Swarm Optimized Simple Adaptive Controller for Spacecraft Proximity Operations Trajectory Tracking

Andriy Predmyrskyy * **Steve Ulrich** **

* *Carleton University,Ottawa, ON K1S 5B6 (e-mail: AndriyPredmyrskyy@cmail.carleton.ca).*
** *Carleton University,Ottawa, ON K1S 5B6 (e-mail: SteveUlrich@cunet.carleton.ca)*

Abstract: Adaptive control design allows for the management of systems with time varying or unknown dynamics. Despite their versatility, few well defined design techniques exist for some classes of adaptive controller. Without analytical techniques it is difficult to prove the efficacy of an adaptive controller design. One solution to this issue is the application of parametric search techniques to adaptive controller design. This paper explores the application of differential evolution on the simple adaptive control law formulation and compares its solution to one found using particle swarm optimization. Afterwards, variations on these techniques, namely the selection particle swarm optimization and self-adaptive differential evolution, are implemented and their results compared. The final swarm-optimized controller is compared to a classical Linear Quadratic Regulator (LQR) controller, and a manually designed simple adaptive controller for precision trajectory tracking control of spacecraft proximity operations. Parametric search techniques are able to determine controller parameters that produce a superior control response. Swarm-optimization techniques determine controllers with parameters drastically different from manually designed efforts.

Keywords: Adaptive Control, Parameter Optimization, Satellite control, Optimal Search Techniques, Model Reference, Spacecraft.

1. INTRODUCTION

Space activities hold high scientific and economic value,and often occur in the vicinity of other objects or spacecraft. Debris in low earth orbit are becoming a threat to the safety of future space missions (Palla (2016)), and will require active debris de-orbiting techniques. On-orbit servicing will dramatically increase their lifespan of missions (Sellmaier (2010)). Asteroid resource extraction may become a market as space capabilities increase(Brophy (2012)). Each of these orbital activities requires accurate trajectory tracking in close proximity to a target while conserving fuel. Spacecraft proximity operations will become a growing part of space-based activities as the need to autonomously perform tasks around a target in orbit increases.

One particular area of interest includes operations around uncooperative targets. Due to the nonlinearity of the kinematics and dynamics equations governing rendezvous and docking motion, nonlinear control methodologies have been developed to improve trajectory tracking performance. Some of these techniques, including feedback linearization or gain scheduling, were inspired by linear control theories, while others, such as sliding mode control or state-dependent Riccati equation-based control were motivated by Lyapunov's stability theory of nonlinear systems. These model based methods and are not suited for rendezvous and docking with uncooperative target, however. Once a rigid connection is established between the target and chaser spacecraft, the combined system has a drastically altered total mass, centre of mass location, and moments of inertia from the original which cannot be readily determined beforehand. Adaptive controllers offer an algorithmic approach to ensure stability even under large uncertainties and disturbances.

Simple Adaptive Control (SAC) is a direct adaptive control technique that has been recommended for spacecraft proximity operations (Ulrich (2014)) which makes use of an ideal model and gain updates to make the control response approach the ideal response (Barkana (2013)) and has been demonstrated to track trajectories during spacecraft proximity operations to a high degree of accuracy under parametric uncertainty and unknown external perturbations (Ulrich (2007)). Despite these successes, however, it is still unknown how to determine a "well-tuned" SAC. In particular the requirement for accuracy and low control activations within spacecraft proximity operations puts particular emphasis on the selection of optimal controller parameters.

In many cases the choice of SAC parameters reverts to the "guess-and-check" method until values with sufficient performance are found. Ulrich (2007) ensures the accuracy and robustness of determined SAC gains using a Monte-Carlo simulation, and Takagi (2017) show that the swarm-based Differential Evolution (DE) optimization technique

can be used to determine control parameters that improve the quadratic cost of a second order single-input single-output system controlled by a SAC.

Several swarm-based optimization techniques are compared here to determine what improvements can be made to the design of a SAC for the spacecraft proximity problem. A two-dimensional Newtonian motion simulation is created to simulate control of a chaser spacecraft around a target. A Linear Quadratic Regulator (LQR) controller is developed to track a trajectory around the target spacecraft, which is then compared to a manually designed SAC for the same problem. A linear quadratic cost function is used in tandem with swarm-based optimization methods to determine new control parameters for a SAC. Differential evolution (Takagi (2017)) and particle swarm optimization (Ouyang (2015)) are tested, along with strategy-adaptive differential evolution (Qin (2009)) and selection particle swarm optimization (Angeline (1998)) which have been identified as well-performing swarm-optimization techniques (Wahab (2015)).

Section 2 covers background concepts in SAC design, while Sec. 3 presents several swarm-optimization parametric search techniques. Section 4 describes the implementation of parametric search techniques and controller simulation in MATLAB & Simulink and highlights the results. Finally, Sec. 5 briefly discusses the results, as well as ways forward.

2. SIMPLE ADAPTIVE CONTROL

A SAC is composed of the adaptive controller and the ideal model (Barkana (2013)). The number of system inputs is equal to the number of outputs l, and the ideal model is of order n. The error between the ideal model and the adaptive controller is defined as

$$e_y = y_{mdl} - y \tag{1}$$

where $e_y \in \mathbb{R}^l$ is the model reference error, y is the measured system output and y_{mdl} the output of the model.

The total control response u is formed from three control terms,

$$u = K_e e_y + K_u U_m + K_x X_m \tag{2}$$

where $K_e \in \mathbb{R}^{l\times l}$ is a matrix of error gains, $K_u \in \mathbb{R}^{l\times l}$ is the matrix of command gains multiplied by $U_m \in \mathbb{R}^l$ the matrix of ideal model inputs, and $K_x \in \mathbb{R}^{l\times n}$ is the matrix of model states gains multiplied by $X_m \in \mathbb{R}^n$ the matrix of ideal model states. The gain matrices K_e, K_u, and K_x are adapted following

$$\dot{K_e}(t) = e_y(t)e_y^T(t)\Gamma_{ei} \tag{3}$$
$$\dot{K_u}(t) = e_y(t)U_m^T(t)\Gamma_{ui} \tag{4}$$
$$\dot{K_x}(t) = e_y(t)X_m^T(t)\Gamma_{xi} \tag{5}$$

where $\Gamma_{ei} \in \mathbb{R}^{l\times l}$, $\Gamma_{ui} \in \mathbb{R}^{l\times l}$, and $\Gamma_{xi} \in \mathbb{R}^{n\times n}$ are called the adaptation parameters and are chosen by the designer. Equations (3) through (5) adapt similarly to an integral gain, and a proportional gain analogue is used to improve convergence following

$$K_{ep}(t) = e_y(t)e_y^T(t)\Gamma_{ep} \tag{6}$$
$$K_{up}(t) = e_y(t)U_m^T(t)\Gamma_{up} \tag{7}$$
$$K_{xp}(t) = e_y(t)X_m^T(t)\Gamma_{xp} \tag{8}$$

In Eqs. (6) through (8) $K_{ep} \in \mathbb{R}^{l\times l}$, $K_{up} \in \mathbb{R}^{l\times l}$, and $K_{xp} \in \mathbb{R}^{n\times n}$ are the proportional gains for error, command, and model states, respectively, with $\Gamma_{ep} \in \mathbb{R}^{l\times l}$, $\Gamma_{up} \in \mathbb{R}^{l\times l}$, and $\Gamma_{xp} \in \mathbb{R}^{n\times n}$ being new learning parameters to be designed. The final gains are given by

$$K_e(t) = K_{ep}(t) + \int_0^t \dot{K_e}(t)dt \tag{9}$$
$$K_u(t) = K_{up}(t) + \int_0^t \dot{K_u}(t)dt \tag{10}$$
$$K_x(t) = K_{xp}(t) + \int_0^t \dot{K_x}(t)dt \tag{11}$$

The matrix product $e_y(t)e_y^T(t)$ is positive definite and causes issues in real systems due to noise, system lag, or other factors that cause $\dot{K_e}$ to stay positive unless perfect tracking is achieved. A degradation parameter σ is added to allow for negative $\dot{K_e}$ values, producing the new adaptation equation

$$\dot{K_e}(t) = e_y(t)e_y(t)^T\Gamma_{ei} - \sigma K_e \tag{12}$$

It is common for Γ matrices to be symmetric and diagonal, or of the form $\Gamma = I_l \times \gamma$ for some constant γ and identity matrix of size $l \times l$. There are seven variables to be chosen by the designer; Γ_{ei}, Γ_{ui}, Γ_{xi}, Γ_{ep}, Γ_{up}, Γ_{xp}, and σ.

3. OPTIMIZATION TECHNIQUES

Particle Swarm Optimization (PSO), Differential Evolution (DE), Selection Particle Swarm Optimization (SPSO), and Strategy-adaptive Differential Evolution (SaDE) are considered for a search of SAC designs. The DE, PSO, SaDE and SPSO search techniques all incorporate multiple agents testing points in a cost function. Agents within each algorithm hold individual states which are then updated to produce a new test position. Information on the cost of the test locations is incorporated into the next set of points tested by the algorithm.

Position is used to refer to a single combination of parameters within the search space as an analogy. The subscript i is used to refer to a parameter specific to an individual agent within the swarm, while the subscript d is used to refer to a given dimension within one of the agent's values.

3.1 Optimization Objective

The objective of optimization will be to minimize the value J of a linear quadratic cost function over the simulation period. The cost to be minimized is given by the equation

$$J(t) = \int_0^t x^T(t)Qx(t) + u^T(t)Ru(t)\,dt \tag{13}$$

where x is the vector of system states or errors, and u is the control action made by the controller. The design parameters Q and R are weight matrices that define the importance of minimizing x or u. The cost function described in Eq. (13) is used to determine the fitness of the any given controller design for all four of the search techniques considered. The simplicity of the cost function, as well as the ability to compare cost values with a simple LQR controller for the system is desirable and will clarify characteristics of SAC design.

3.2 Particle Swarm Optimization:

To begin a PSO search (Ouyang (2015)) with designed velocity dampening factor ω, agent optima attraction ϕ_p, and global optima attraction ϕ_g:

(1) Randomly assign an initial position x for each agent in the search space.
(2) Set the best position p for each agent to x.
(3) Determine the cost for each agent's position, called j, and set the swarm's minimum cost g to the lowest cost (g =min(j)), with an associated best position q.
(4) Determine the initial velocity v of each agent as a random vector.
(5) Repeat iteratively until the completion criteria are met, for each agent i:
  (a) Pick random values r_p and r_g between 0 and 1.
  (b) Update the agent's velocity as:
$$v_i' = \omega v_i + \phi_p r_p(p_i - x_i) + \phi_g r_g(q - x_i) \quad (14)$$
  (c) Update the agent's position following:
$$x_i' = x_i + v_i \quad (15)$$
  (d) Check the cost of the new position x_i'.
  (e) If the new cost is lower than the agent's best cost j_i, make the agent's new best position p_i equal to the current position.
  (f) If the new cost is lower than the swarm's best cost g, make the swarm's best position q the current agent's position and make the new best cost g equal to the current agent's cost j_i.

3.3 Differential Evolution:

To perform a DE search (Takagi (2017)) with designed crossover rate δ and scaling factor F:

(1) Initialize each agent with a random position p within the search space.
(2) Determine the cost of each agent j at position p.
(3) Determine which agent's position has the lowest cost, marking it as q, and saving the lowest cost as g.
(4) Repeat iteratively until the completion criteria are met, for each agent i:
  (a) Pick two random integers from 1 to the swarm population that are not identical and not i, calling them a_1 and a_2.
  (b) Produce a mutated vector M from the a_1^{th} and a_2^{th} agent following:
$$M = q + F(p_{a_1} - p_{a_2}) \quad (16)$$
  (c) Choose an integer b at random from 1 to the dimensionality of the problem.
  (d) Create a trial vector by checking the following for each dimension d using the design parameter δ:
    (i) Pick a random value ρ from 0 to 1.
    (ii) The trial vector u is defined for each dimension d as:
$$u_d = \begin{cases} M_d, & d = b \text{ or } \rho < \delta \\ p_d, & \text{otherwise} \end{cases} \quad (17)$$
  (e) Determine the current cost of the trial vector u.
  (f) If the cost of u is lower than the current position cost j_i, the agent's best known position p_i becomes equal to the trial vector u.
  (g) If the cost of u is lower than the best global cost g, the global best known position q becomes the trial vector u.

3.4 Selection Particle Swarm Optimization:

To begin an SPSO search (Angeline (1998)):

(1) Initialize similarly to PSO steps (1) through (4)
(2) Iteratively until the completion criteria are met:
  (a) Sort the population by cost and mark the half of the agents with the highest cost for selection. Replace the current positions x of each agent marked for selection randomly with one of the agents not marked for selection. Replace the current velocity v of each agent marked for selection randomly with one of the agents not marked for selection. Keep the best known position p of each agent unchanged.
  (b) Proceed with the optimization identically to the iterative steps in the PSO search from step (5a) through (5f).

3.5 Strategy-Adaptive Differential Evolution:

Many mutation strategies exist for DE. A pool of mutation strategies is kept by SaDE and used to improve the search. Strategies that are more likely to result in success are more likely to be chosen in future. A success is recorded as any time a strategy produces a trial vector that decreases the best cost of an agent, with the total success of the k^{th} strategy during the current iteration m being denoted by $s_{k,m}$. When the cost does not decreased it is recorded as a failure, denoted by $f_{k,m}$.

To begin a SaDE search with K trial vector generation function strategies, and learning period LP (Qin (2009)):

Initialize similarly to DE steps (1) through (3), then repeat iteratively, calling the iteration number n, until the completion criteria are met:

(1) If the current iteration number n is greater than the designed learning period LP then:
  (a) Calculate the success fraction S of each strategy throughout the learning period following:
$$S_{k,m} = \frac{\Sigma_{t=m-LP}^{m} s_{k,t}}{\Sigma_{t=m-LP}^{m} s_{k,t} + \Sigma_{t=m-LP}^{t} f_{k,t}} + \epsilon \quad (18)$$
  Values for $s_{k,t}$ and $f_{k,t}$ are recorded later. A small number ϵ is added for numerical stability.
  (b) Determine the probability of choosing the k^{th} strategy $P_{k,m}$ following
$$P_{k,m} = \frac{S_{k,m}}{\Sigma_{t=1}^{K} S_{t,m}} \quad (19)$$
  (c) Determine the new crossover ratio median δ_m as the average of crossover ratios that resulted in successful trial vectors over the last LP iterations.
(2) If $n < LP$, set the probability for each trial vector generation function $P_{k,m}$ to be equal, such that each function has an equal likelihood of being chosen.
(3) For each agent i:
  (a) Use probabilities $P_{k,m}$ to randomly choose a strategy k.
  (b) Choose F_i and G_i as normal random values with a standard deviation of 0.3 and median of 0.5.

(c) Choose δ_i as a normally distributed random value with median of δ_m and standard deviation of 0.1, ensuring $0 < \delta_i < 1$.
(d) Use the k^{th} strategy to create the trial vector u_i.
(e) Determine the current cost of the trial vector u_i.
(f) If the cost of u_i is lower than j_i, set j_i to the current cost, and set $p_i = u_i$. Increase the number of successes $s_{k,i}$ for the chosen strategy k by one and add the crossover ratio δ_i used to the list of successful crossover ratios. If the cost of u_i was not lower than the current cost j_i, increase the number of failures $f_{k,i}$ for the strategy k by one.
(g) If the cost of the trial vector u_i is lower than the best global cost g, the global best known position q becomes the trial vector u_i.

The four strategies used are outline in Eqs. (20) through (23). When used in parallel the methods require five random distinct integers a_1, a_2, a_3, a_4, a_5 with values from 1 to the swarm population that correspond to indices of members of the population that are not the agent being considered. These strategies also require an integer b between 1 and the dimensionality of the problem, a random value ρ between 0 and 1, the previously determined F_i, G_i, and δ_i values, the value of any i^{th} agent's position in the d^{th} dimension $p_{i,d}$ and the best known position in that dimension q_d. The strategies and their names are listed below.

(1) DE/rand/1/bin: For each dimension d:

$$u_{i,d} = \begin{cases} p_{a_1,d} + F(p_{a_2,d} - p_{a_3,d}), \rho < \delta_i \text{ or } d = b \\ p_{i,d}, \text{otherwise} \end{cases} \tag{20}$$

(2) DE/rand-to-best/2/bin: For each dimension d:

$$u_{i,d} = \begin{cases} p_{i,d} & +F(q_d - p_{i,d}) + F(p_{a_1,d} - p_{a_2,d}) \\ & +F(p_{a_3,d} - p_{a_4,d}), \rho < \delta_i \text{ or } d = b \\ p_{i,d} & , \text{otherwise} \end{cases} \tag{21}$$

(3) DE/rand/2/bin: For each dimension d:

$$u_{i,d} = \begin{cases} p_{a_1,d} & +F(p_{a_2,d} - p_{a_3,d}) \\ & +F(p_{a_4,d} - p_{a_5,d}), \rho < \delta_i \text{ or } d = b \\ p_{i,d} & , \text{otherwise} \end{cases} \tag{22}$$

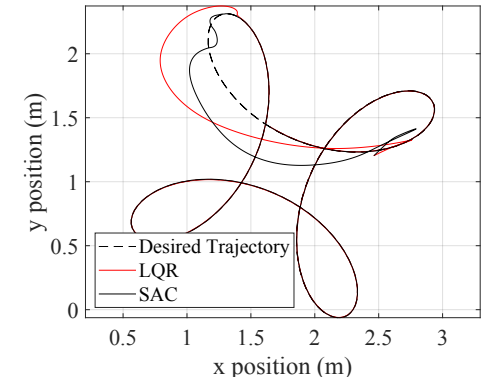
(4) DE/current-to-rand/1:

$$u_i = p_i + G(p_{a_1} - p_i) + F(p_{a_2} - p_{a_3}) \tag{23}$$

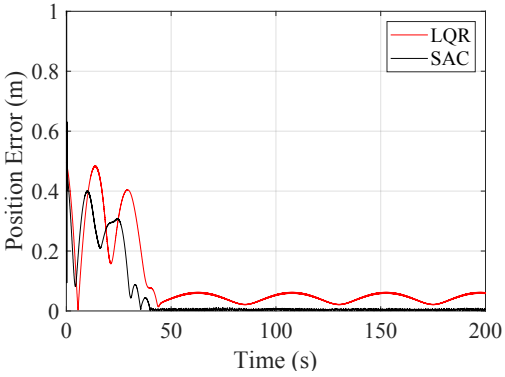
4. NUMERICAL SIMULATIONS

The local spacecraft tracking problem was used to test the validity of parametric search techniques on SAC parameter selection. A chaser spacecraft is made to track a trajectory relative to a target spacecraft within the same orbital plane. This scenario is representative of several rendezvous and proximity operations that occur in space.

Simulations were performed for position control of a 2-DOF spacecraft with a mass of 16.95 kg, and thrusters with a maximum force of 0.425 N allocated by a pulse-width-modulation (PWM) scheme. No disturbances or measurement noise were present in simulation. The non-linear saturation and PWM allocated thrust cannot be considered by the LQR design, but affect both the SAC



(a) Trajectory



(b) Position Error

Fig. 1: LQR and Manually Tuned SAC Performance

and LQR controller responses. The SAC controls each axis independently to create a two-input two-output controller. Rotation was not considered. A standard 2^{nd} order state-space model was used for the continuous position dynamics of the spacecraft with saturation limits on the thrust output. The linear state-space model for the system is given by the equations and states

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 1/16.95 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \vec{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{24}$$

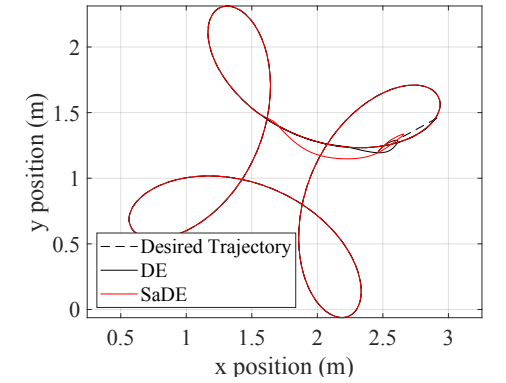
which are repeated for the x and y directions. Stability can be guaranteed for the linear system using SAC since it is almost strictly positive real (Barkana (2013)).

A LQR and a manually designed SAC are made to follow a commanded time varying position in space. The response of the adaptive controller is then optimized using parametric search techniques in order to yield an optimized controller. A Simulink file was used to run the simulation necessary for cost function calls during optimization. The cost of each SAC controller response to a step input was used for optimization, while a sinusoidal command was used for validation given by the equations

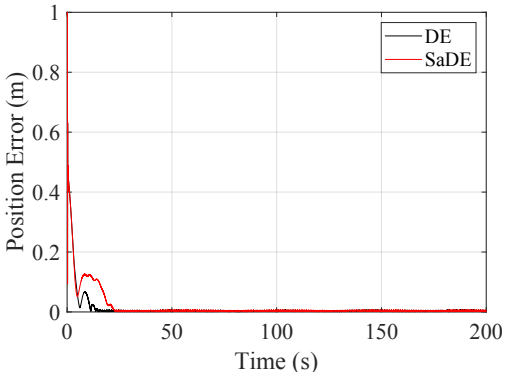
$$x_{cmd}(t) = R\cos(\omega t) + H\sin(K\omega t) \tag{25}$$
$$y_{cmd}(t) = R\sin(\omega t) + H\cos(K\omega t) \tag{26}$$

with values of 3 for K, 1.4286 for R, 1 for H, and 0.035 for ω to produce a track command resembling a four lobed leaf visible in Fig. 2a.

A LQR minimizes the cost function demonstrated in Eq. (13) in linear systems and is developed as a compar-

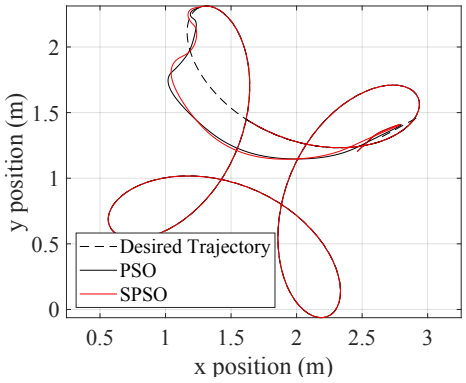


(a) Trajectory

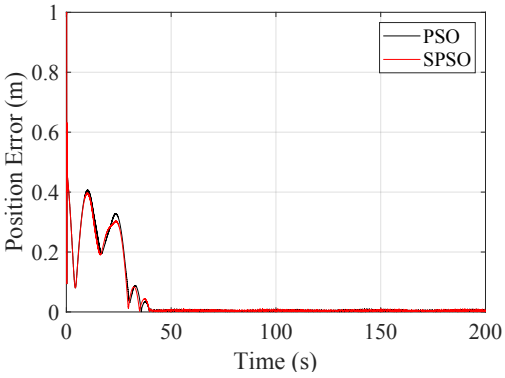


(b) Position Error

Fig. 2: DE and SaDE Optimized SAC Performance



(a) Trajectory



(b) Position Error

Fig. 3: PSO and SPSO Optimized SAC Performance

ison for the SAC responses. The weight matrices chosen were

$$Q = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{27}$$

For the linear system approximation and states $\vec{x}$, LQR formulations determined an optimal gain matrix K for each axis of

$$K = [31.6 \ 45.5] \tag{28}$$

The SAC controller ideal model is composed of a second order transfer function with a damping ratio ζ of 1 and a large natural frequency ω_n of 40 rad/s in order to allow for apt comparison of the SAC and LQR, since the SAC follows the reference model, and the LQR tracks the command. The same ideal model was used for the development of all SAC controllers. Since a second order model was used, two Γ values are used for the state learning parameter values. The final gamma matrices are in the form $\Gamma_{eP} = \text{diag}(I_2\Gamma_{xp1}, I_2\Gamma_{xp2})$ and $\Gamma_{ei} = \text{diag}(I_2\Gamma_{xi1}, I_2\Gamma_{xi2})$.

Manual tuning of a SAC was performed using the familiar "guess-and-check" technique. Each controller parameter would be varied while until no increase in performance could be found. Optimization of the controller was then performed using DE, SaDE, PSO, and SPSO algorithms.

A script incorporating the search techniques presented in Sec. 3 was implemented in MATLAB, with Simulink calls to run the spacecraft proximity operation simulation and determine a cost. During operation each search algorithm determined a combination of controller design parameters

Table 1: SAC Parameters

Variable	Designed SAC	DE	PSO	SaDE	SPSO
Γ_{ep}	100	4.394×10^5	7.007×10^4	4.938×10^5	1.225×10^5
Γ_{ei}	1×10^5	0	1.277×10^5	0	4.513×10^5
Γ_{up}	100	614.5712	1.582×10^5	0	5.345×10^5
Γ_{ui}	1	0	0	0	0
Γ_{xp1}	100	0	2.079×10^4	867.9	2.580×10^5
Γ_{xp2}	100	1.000×10^6	1.672×10^4	4.851×10^5	3.833×10^5
Γ_{xi1}	1	0	0	0	0
Γ_{xi2}	1	1.000×10^6	5.827×10^4	2.421×10^5	9.998×10^5
σ	0.4	0.93	0.21	0.54	0.02
Cost	3338.6	1283.5	3337.6	1405.9	3184.0

to test. The controller response to a step input over 100 seconds was used to determine the cost.

All four algorithms were run for 100 iterations with 100 agents. Values of $\omega = 0.2$, $\phi_g = 0.1$, and $\phi_r = 1.0$ were chosen for the PSO search. The DE search was performed using values of 0.9 and 0.8 for the crossover rate and scaling factor, respectively. For SPSO, 50% of the agents were used to provide updated positions of the other 50% of agents. In the SaDE search the four trial vector generation functions described in Sec. 3.5 were used in strategy adaptation.

The designed SAC controller gains and their optimized counterparts are demonstrated in Table 1, along with the final cost of each controller. The results for the designed LQR and SAC controllers are shown in Figs. 1 through 3.

The designed LQR controller achieved similar transient behaviour as the manually designed SAC, but was unable to reach zero error.

It can be seen that all optimization methods were able to find parameters that improve the response when compared with a manually designed SAC. The DE and SaDE searches were able to determine significantly lower cost controllers than the PSO and SPSO searches.

While the designed controller contains mostly feedback error adaptation, The DE determined controller contains mostly state adaptation. The DE controller converges on the model response very quickly, which is reflected in the cost. Optimization significantly improved the controller cost. It is likely that the absence of disturbances and measurement noise caused the optimization to ignore error adaptation Γ_{ei} since adequate control could be achieved with only state adaptation.

The SPSO search produced very similar results to the PSO search, using high error and state adaptations to improve the tracking of the hand-designed controller. The use of selection in SPSO likely increased convergence by moving high cost agents closer to the lower cost agents and more quickly refining the determined control parameters (Angeline (1998)). The faster convergence of PSO and SPSO techniques compared to DE and SaDE may have contributed to the higher final cost of their designs, due to lower exploration of the design space.

Generally it was found that convergence and exploration behaviour of the techniques matched those presented in Wahab (2015), with SPSO having the fastest convergence time, at the expense of exploration, followed by PSO, SaDE, and finally DE which more thoroughly explored the search space at the expense of convergence. The explicit attraction of PSO to the determined minimum increases convergence when compared to DE, with selection pressure further increasing convergence in SPSO. The mutations present in DE encourage exploration but reduce convergence behaviour, and the use of multiple strategies in SaDE increase the convergence behaviour of DE somewhat.

The simulation used in this survey did not include any measurement noise or disturbances. It is possible that due to the lack of complicating factors in simulation that some of the determined controller parameters may not be useful in practice. For example, although the DE controller has the lowest cost, it does not make use of error adaptation at all, and may have a higher cost response than a controller including these terms when implemented in hardware. Similarly, all controllers used very large values for many of the adaptation terms, which may cause instability if the simulated system response is faster than the true system response.

Finally, since several parameter configurations were able to produce similar costs for the controller, it is likely that even for quadratic cost functions there exists a complex cost landscape for parameter selection in SACs. Wahab (2015) suggests that DE has the best performance of swarm-based techniques for multimodal cost functions, which may be the case for many SAC designs.

5. CONCLUSIONS

Several swarm-optimization techniques were used to determine if an optimal adaptive controller could be found for the local spacecraft trajectory tracking problem. Optimized controllers were compared to a manually designed effort, as well as to a standard LQR controller. It was found that swarm-optimization techniques were successfully able to determine adaptive control parameters that lowered the cost of a linear quadratic cost function. The designed controllers are shown to increase performance when compared with manually designed efforts.

Future work in this field may include research into the structure of cost functions used in conjunction with adaptive controllers. The standard quadratic cost function may be difficult to optimize for the SAC architecture. The two distinct response types found by the four particle swarm algorithms presented here suggest that several local minima exist despite the simple cost function used. In future, techniques to minimize the search space or determine characteristics of the cost function applied may greatly improve search speeds and results or may suggest other avenues of research.

REFERENCES

P. Angeline, "Using Selection to Improve Particle Swarm Optimization," *IEEE World Conference on Computational Intelligence*, Anchorage, USA, 1998.

I. Barkana, "On Robustness and Perfect Tracking with Simple Adaptive Control in Nonlinear Systems," *20^{th} IFAC World Congress*, Vol. 50 No. 1 pp. 4258-4263 2017.

I. Barkana, "Simple Adaptive Control: The Optimal Model Reference - Short tutorial," *11^{th} IFAC International Workshop on Adaptation and Learning in Control and Signal Processing*, Caen, France, 2013.

J. R. Brophy et. al., "Asteroid Retrieval Feasibility," *2012 IEEE Aerospace Conference*, Montana, USA, 2012.

P. Ouyang, and V. Pano, "Comparative Study of DE, PSO and GA for Position Domain PID Controller Tuning," *Algorithms*, Vol. 8 No. 3 pp. 697-711, 2015.

C. Palla, J. Kingston, "Forecast Analysis on Satellites that need De-orbit Technologies: Future Scenarios for Passive De-orbit Devices," *CEAS Space Journal*, Vol. 8 No. 3 pp. 191-200, 2016.

A. Qin, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*,Vol. 13 No. 2 pp. 398-417, 2009.

F. Sellmaier et. al., "On-Orbit Servicing Missions: Challenges and Solutions for Spacecraft Operations," *SpaceOps 2010 Conference*, Alabama, USA, 2010.

T. Takagi, M. Ito, and I. Mizumoto, "Parameter optimization of simple adaptive control via differential evolution," *2017 6^{th} International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, Taipei, Taiwan, 2017.

M. Wahab et al., "A Comprehensive Review of Swarm Optimization Algorithms," *Plos One* Vol. 10 No. 5 , 2015.

S. Ulrich et. al., "Simple Adaptive Control for Spacecraft Proximity Operations," *AIAA Guidance, Navigation, and Control Conference*, Maryland, USA, 2014.

S. Ulrich, and J. de Lafontaine, "Autonomous Atmospheric Entry on Mars: Performance Improvement Using a Novel Adaptive Control Algorithm," *The Journal of the Astronautical Sciences*, Vol. 55 No. 4 pp. 431-449, 2007.