Adaptive Boundary Sine Cosine Optimizer With Population Reduction For Robustness Analysis of Finite Time Horizon Systems

Felix Biertümpfel[a], Nantiwat Pholdee[b,*], Sujin Bureerat[b], Harald Pfifer[a]

[a]*Chair of Flight Mechanics and Control, Technische Universität Dresden, 01307, Dresden, Germany*

[b]*Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Thailand, 40002*

Abstract

This paper proposes an adaptive boundary sine cosine optimizer with population reduction. It is designed specifically to calculate an upper bound on the worst-case gain of a known finite horizon Linear Time Varying (LTV) system and a perturbation. The input/output behavior of the perturbation is described by a time domain Integral Quadratic Constraint (IQC). The analysis condition is formulated as a parametric Riccati differential equation which depends on the IQC representation. A nonlinear optimization problem is posed that minimizes the upper bound on the worst-case gain over the IQC parameterization. For industrial size applications like a space launcher, the number of decision variables can grow arbitrarily large depending on the number of considered perturbations as well as the type and representation of the IQC. This is aggravated by nonlinear constraints on the decision variables imposed by the Riccati differential equation making it challenging to solve. Several established Meta-Heuristics (MHs) along with the proposed algorithm are applied to an industry size worst case analysis of a space launcher during its atmospheric ascend. Their respective performances are evaluated to emphasize the advantages of the developed optimizer. This work builds the foundation of applying MHs to IQC based robustness analysis of finite horizon LTV systems.

Keywords: Finite Horizon Linear Time Varying Systems, Meta-Heuristics, Robust Performance, Optimization With Adaptive Bounds, Sine Cosine

*Corresponding Author

1. Introduction

Meta-heuristic (MH) represents a general algorithmic framework with guidelines for the development of search algorithms. Typical algorithms designed under the meta-heuristic paradigm are, for example, evolutionary algorithms, swarm intelligent algorithms, or estimation of distribution algorithms [2, 26]. Over the last decades, these algorithms have attracted increasing attention and have been implemented on complex engineering problems. In general, these methods have been developed based on imitation of nature such as genetic evolution [18], laws of physics [34], [58], food finding of insects or animals [43, 46, 48, 44, 68, 22, 38, 32], etc. [3, 6, 19, 68, 20]. They are classified as global optimization methods due to the use of a population-based concepts and randomization in the search procedure. This offers the advantage of avoiding being trapped at a local optimum. Moreover, they require no function derivatives in an optimization run. Thus, they are easy to use and can be applied to almost any kind of functions and design variables. However, the lack of search consistency and slow convergence are still the main drawbacks of MHs. In this regard, numerous MHs have been developed and upgraded over the last two decades emphasizing mainly on increasing their exploitation and exploration capabilities. For various engineering applications, the successful use of MHs has been reported in literature [15, 54, 55, 71, 72, 73, 8, 23, 56, 59, 75]. Meanwhile, novel search methods, including the gradient-based optimizer (GBO)[3], the slime mould algorithm (SMA)[38], heap-based optimizer (HBO) [6], and Harris hawks optimization (HHO)[32], extended the range of available MHs. However, a popular class of population-based MH algorithms remains sine cosine algorithms (SCAs) first introduced by Mirjalili et al. 2016 [46]. These algorithms apply trigonometric sine and cosine functions for population updates. As they do not require predefined/specific optimization parameters settings, their application is user-friendly and straightforward. This led to their employment on numerous optimization problems [26], demonstrating their efficiency and robustness. Since its introduction, the original SCA has undergone several improvements in the form of, e.g a binary algorithm [25], adaptive parameters [24], and hybridization techniques [37, 30], with further successful applications [7, 4, 16]. However, in contrast to, e.g. genetic algorithm (GA) [50, 17, 70], differential evolution (DE) [14, 13], particle swarm optimization (PSO) [9, 57, 40], or Lévy flight based pigeon-inspired optimization (LFPIO) [19], no implementation of SCA for complex control engineering problems, such as robustness analyses, exists. Also, population reduction and adaptive boundary strategies to address large design domain problems (arising in robustness analysis) have not been implemented in SCAs, motivating further investigation.

In this paper, an adaptive boundary sine cosine optimizer with population reduction is proposed that specifically addresses the robustness analysis problem for finite time horizon systems [10]. A wide variety of control problems can be formulated as finite horizon linear time varying (LTV) systems, i.e. systems whose dynamics vary with time and that follow a specific predefined trajectory. Examples of this application include robot arms [52], swarm robots [51], terminal guidance systems [35] or the in the benchmark considered atmospheric ascend of a space launcher [10].

Of specific interest for these applications is how they perform under perturbations, e.g. uncertainties or hard nonlinearities such as saturations. Results on the input/output gain computation of "nominal" finite time horizon LTV systems based on the solution of a Riccati differential equation (RDE) [27] have been recently leveraged to include perturbations via the framework of integral quadratic constraints (IQC) [10, 62]. Specifically, IQCs are used to bound the input/output characteristic of the perturbations in the system [42]. In general, IQCs are not unique and a family of IQC parameterizations can describe the same perturbation behavior. Effectively, the analysis results for perturbed, finite time horizon LTV systems are still based on the solution of a Riccati differential equation (RDE) which is now, however, parametrized by the chosen IQCs. Depending on the chosen parameters, the performance analysis can give widely different bounds on the input/output gain of the analyzed system. Hence, it is necessary to formulate the robustness analysis in this framework as a minimization of the upper bound on the worst-case gain as a function of the IQC parameterization. This amounts to a highly complicated nonlinear optimization problem with a parameterized differential equation as nonlinear constraint. The number of decision variables scales with the number of considered perturbations in the system and complexity of the chosen IQC parameterization. In addition, the search space is not easily defined, has to be assumed non-convex and non-smooth. Furthermore, it can be, in general, arbitrary large. All these factors make the problem ideally suitable to be tackled with a MH. Although many research works applied MHs to engineering problems, none of them was applied to the LTV robustness problem [14, 13, 15, 16, 54, 55, 73, 71, 72, 8, 23, 56, 59, 75]. Thus, an evaluation of their comparative performance for this specific problem has yet to be investigated. In this paper, a thorough comparison between the proposed adaptive boundary sine cosine optimizer and established MHs is presented in Section 4 on the example of the analysis of a space launcher [10]. Specifically, the MHs used in this study include: differential evolution [63][38], adaptive differential evolution with optional external archive (JADE) [74], success-history based adaptive differential evolution (SHADE) [64], whale optimization algorithm (WOA) [47], moth-flame optimization algorithm (MFO) [43], dragonfly algorithm (DA) [45], grey wolf optimizer (GWO) [48], heap-based optimizer [6], Harris hawks optimization [32], particle swarm optimization algorithm (PSO) [22], genetic algorithm (GA) [18] , and sine cosine algorithm [46]. The considered benchmark study is of a sufficient engineering complexity, so the results of the study allow drawing conclusions of the efficiency and industrial relevance of the considered algorithm.

2. Robust Performance of Linear Time Varying Systems

Automation becomes more and more prominent in a variety of systems applications. A significant subset of these systems follow a preprogrammed trajectory leading the system from a fixed starting point to a fixed terminal point. A typical example for trajectory based operations are industry robots, which are used for automated assembly [12], materials and quality testing [33] or manufacturing [36, 49]. Another example are space launchers during atmospheric ascent as depicted in Fig. 1. The launcher has to tightly follow a predefined trajectory starting from the lift-off and ending with the burn-out of the first stage. Consequently, its nonlinear equations of motion are actually time dependent. Hence, their linearization along this specific trajectory results in a finite horizon linear time varying system. Note that the linearization is necessary as no sufficient worst case analysis condition for nonlinear systems exists. The following space launcher example illustrates how this procedure exactly works and a general linear time varying system can be derived from the system dynamics along the design trajectory.

Example 1.

During the atmospheric flight phase carried out by the first stage, the launcher's guidance is typically tracking a preprogrammed trajectory [21]. Focusing on the vertical plane (pitch motion), the launch vehicle follows a pitch program, i.e. the launcher tracks a pre-calculated time dependent pitch angle trajectory θ_d. It usually describes a so called gravity turn maneuver, i.e. the nominal angel of attack α is zero [69]. Linearizing the nonlinear equation of the pitch dynamics, with respect to the launcher fixed frame (subscript b), along the gravity turn trajectory

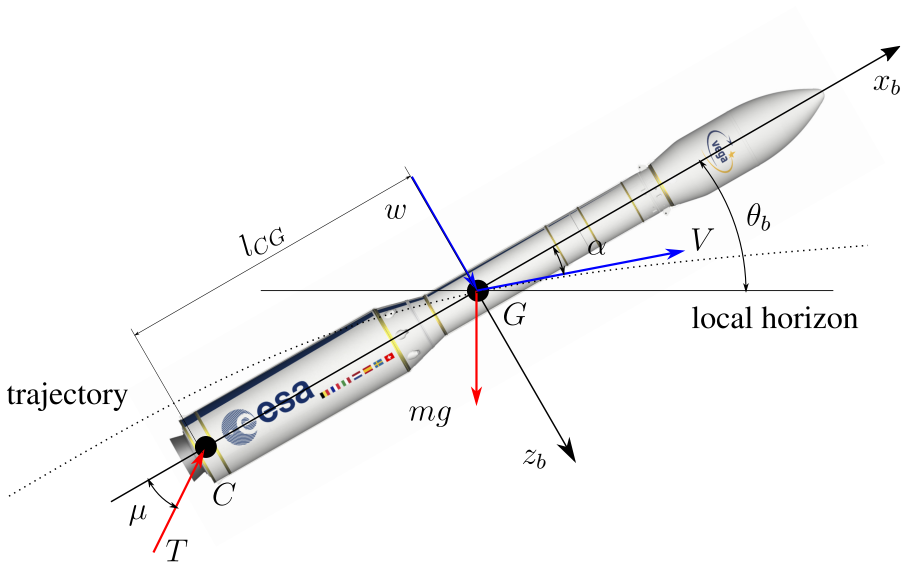

Figure 1: Vega space launcher (Source: ESA)

leads to the following linearized dynamics:

$$\begin{bmatrix} \dot{\alpha}(t) \\ \dot{\theta}_b(t) \\ \ddot{\theta}_b(t) \end{bmatrix} = \begin{bmatrix} Z_\alpha(t) & -\frac{g_0(t)}{V(t)} & 1 \\ 0 & 0 & 1 \\ \frac{M_\alpha(t)}{Jy(t)} & 0 & \frac{M_q(t)}{J_y(t)} \end{bmatrix} \begin{bmatrix} \alpha(t) \\ \theta_b(t) \\ \dot{\theta}_b(t) \end{bmatrix} + \begin{bmatrix} \frac{T(t)}{m(t)V(t)} & \frac{Z_\alpha(t)}{m(t)V(t)} \\ 0 & 0 \\ \frac{T(t)l_{CG}(t)}{J_y(t)} & \frac{M_\alpha(t)}{J_y(t)} \end{bmatrix} \begin{bmatrix} \mu(t) \\ \delta_\alpha(t) \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \alpha(t) \\ \theta_b(t) \\ \dot{\theta}_b(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha(t) \\ \theta_b(t) \\ \dot{\theta}_b(t) \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mu(t) \\ \delta_\alpha(t) \end{bmatrix}$$

where the states are the angle of attack α, the pitch angle θ_b, and the pitch rate $\dot{\theta}_b$. The angle θ_b effectively describes the offset to the desired pitch angle θ_d. In (1), the inputs are the TVC deflection μ used for attitude control and the wind disturbance which is modeled as an additional exogenous angle of attack signal δ_α. The disturbance δ_α is defined parallel to the z_b-axis of the launcher fixed frame, see Fig. 1. It is given by $\delta_\alpha \approx \frac{-w}{V}$, where V is the velocity of the space launcher.

The variables Z_α, M_α and M_q are the aerodynamic stability derivatives, see [66]. In (1), m is the total mass of the ELV, and J_y denotes the overall mass moment of inertia with respect to the launcher's center of gravity G. The thrust is denoted as T. It acts at the nozzle reference point C. The geometric variables l_{CG} is defined as the absolute distances between C and G. The gravitational acceleration g_0 is modeled according to the WGS 84 assuming a launch at the equator. In general, all the introduced variables are time dependent as a result of the predefined trajectory.

Consequently, the linear dynamics of the space launcher can be written more generally as a finite horizon linear time varying system G

$$\begin{aligned} \dot{x}_G(t) &= A_G(t)x_G(t) + B_G(t)d(t) \\ e(t) &= C_G(t)x_G(t) + D_G(t)d(t), \end{aligned} \tag{2}$$

where $x_G(t) \in \mathbb{R}^{n_x}$, $d(t) \in \mathbb{R}^{n_d}$ and $e(t) \in \mathbb{R}^{n_e}$ are the state, input and output vector, respectively. In (2), the system matrices are piecewise continuous bounded functions of time t of compatible size to the corresponding vectors, e.g. $A_G(t) \in \mathbb{R}^{n_{xG} \times n_{xG}}$. Many other applications can be written in this general form, e.g. aircraft in final approach, robots or spacecraft during the reentry and landing.

2.1. Uncertain Linear Time Varying Systems

Coming back to the example of the space launcher, its dynamics as described above are inherently unstable if it does not posses aerodynamic surfaces. Hence, the tasks of its control system are stabilization and tracking of the preprogrammed pitch angle θ_d. As a diversion from the design trajectory could result in a loss of the launcher, the pitch angle has to be tracked tightly. Therefore, the control system must perform robustly not only under external disturbance such as wind turbulence, but also in presence of uncertainty/perturbations regarding its dynamics (1). These system perturbations have to be identified. Subsequently, the feedback interconnection $F_u(G, \Delta)$ of the nominal system G and Δ as shown in Fig. 3 can be written generally as

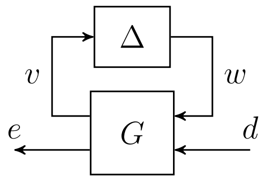

Figure 2: Feedback Interconnection LTV system G and uncertainty Δ

$$\begin{aligned}\dot{x}_G(t) &= A_G(t)\,x_G(t) + B_G(t)\,d(t)\\ e(t) &= C_G(t)\,x_G(t) + D_G(t)\,d(t)\\ w(t) &= \Delta(v),\end{aligned} \tag{3}$$

where $v \in \mathbb{R}^{n_v}$ and $w \in \mathbb{R}^{n_w}$ are the perturbation input and output vector, respectively.

2.2. Integral Quadratic Constraints

The input/output behavior of uncertainties/perturbations Δ in LTV systems can be bounded by the means of IQCs. The time domain definition of an IQC is based on a filter $\Psi \in \mathbb{RH}_\infty^{n_z \times (n_v+n_w)}$ and a $n_z \times n_z$ real, symmetric matrix M ([61]) The uncertainty Δ satisfies the IQC defined by M and Ψ if the output z of the filter Ψ fulfills the quadratic time constraint

$$\int_0^T z(t)^T M z(t)\,dt \geq 0 \tag{4}$$

for all $v \in L_2[0,T]$ and $w = \Delta(v)$ over the interval $[0,T]$. In this case the short notation $\Delta \in IQC(\Psi, M)$ is used.

2.3. LTV Robust Performance Analysis

Using the worst case analysis condition for nominal LTV systems presented in [27] as well as the time domain IQC representation of the uncertainty Δ, it is possible to propose a robust performance analysis [10, 62]. This allows for the worst case analysis of the interconnection $F_u(G_t, \Delta)$. Therefore, the IQC filter Ψ has to be introduced in said interconnection as illustrated in Fig. 3. The dynamics

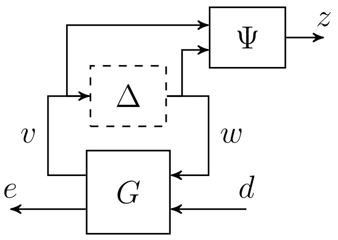

Figure 3: Feedback Interconnection LTV system G and uncertainty Δ

of the interconnection depend on an extended LTV system G_{ext} of the following form:

$$\begin{aligned}\dot{x}(t) &= A(t)\,x(t) + \begin{bmatrix} B_1(t) & B_2(t)\end{bmatrix}\begin{bmatrix} w(t)\\ d(t)\end{bmatrix}\\ \begin{bmatrix} z(t)\\ e(t)\end{bmatrix} &= \begin{bmatrix} C_1(t)\\ C_2(t)\end{bmatrix} x(t) + \begin{bmatrix} D_{11}(t) & D_{12}(t)\\ D_{21}(t) & D_{22}(t)\end{bmatrix}\begin{bmatrix} w(t)\\ d(t)\end{bmatrix}.\end{aligned} \tag{5}$$

In (5), $x(t) \in \mathbb{R}^{n_x}$ represents the state vector containing the states of G_t and Ψ, $d(t) \in \mathbb{R}^{n_d}$ the input vector and $e(t) \in \mathbb{R}^{n_e}$ the output vector. Consequently, the time domain inequality (4) enforced on the output z of Ψ is used to replace the explicit representation of the uncertainty $w = \Delta(v)$. Performance metrics for uncertain LTV systems are usually based on worst case gains, i.e. the worst case amplification from the input signal $d(t)$ to the performance outputs over all $\Delta \in IQC(\Psi, M)$. An example is the finite horizon worst case induced $L_2[0,T]$ gain [10, 62]. It is defined as

$$\|F_u(G_t,\Delta)\|_{2[0,T]} := \sup_{\Delta \in \text{IQC}(\Psi,M)} \sup_{\substack{d \in L_2[0,T] \\ d \neq 0, x(0)=0}} \frac{\|e(t)\|_{2[0,T]}}{\|d(t)\|_{2[0,T]}}, \tag{6}$$

i.e. the worst-case amplification of the finite horizon 2-norm from performance input to performance output over all valid input signals in the Lebesgue-2 space and $\Delta \in IQC(\Psi, M)$. It is most suitable to identify the maximum (signal/energy) amplification from a disturbance input to performance output. This is especially interesting to evaluate the tracking performance of launch vehicles.

An upper bound γ on the worst case gain $\|F(G_t,\Delta)\|_{2[0,T]}$ is defined by the strict BRL for IQCs given in [10] and [62]. The interested reader can find the respective Theorem in Appendix A.

2.4. Worst Case Gain Optimization Problem

The exact/concrete value of the worst case gain $\|F(G_t,\Delta)\|_{2[0,T]}$ can not be determined. Although, an upper bound γ on its value is defined by the strict BRL for IQCs given in [10] and [62]. The existence of γ is directly related to the solvability of the RDE

$$\begin{aligned}\dot{P}(t) = & Q(t,M,\gamma) + P(t)\tilde{A}(t,M,\gamma) + \tilde{A}^T(t,M,\gamma)P(t) \\ & - P(t)S(t,M,\gamma)P(t).\end{aligned} \tag{7}$$

Details as well as the complete strict BRL theorem are given in Appendix A. In other words, the input/output gain of the interconnection is bounded by γ, if for a fixed Ψ and M the RDE (7) has a stabilizing solution. As in general an infinite amount of valid IQCs exists to describe a given uncertainty, a common approach in literature is to select a fixed filter Ψ and parameterize $M(\lambda) \in \mathcal{M}$ [61, 67], where M is a function of the decision vector λ. Hence, $M(\lambda)$ is one parameterization in the feasibility set of possible parameterization $\mathcal{M}$, which imposes a certain structure and constraints on $M(\lambda)$. This procedure is emphasized by the following example.

Example 2.

Any norm bounded dynamic uncertainty Δ, with $0 < \|\Delta\|_\infty \leq b$ satisfies the $IQC(\Psi, M)$ with

$$\Psi = \begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix} \text{ and } M = \begin{bmatrix} b^2 & 0 \\ 0 & -1 \end{bmatrix}, \tag{8}$$

where D represents any stable minimum phase LTI system. In general, multiple IQCs build from valid Ψ and M exist which all individually upper bound Δ. In [53], it was shown that Δ also satisfies any conic combination build of these IQCs. This means, the outputs z_i of the respective IQC filters Ψ_i fulfill the quadratic time constraint build by the conic combination of multipliers

$$\int_0^T z_1(t)^T \lambda_1 M z_1(t) + ... + z_k(t)^T \lambda_k M z_k(t)\, dt \geq 0,\ \forall \lambda_i > 0,\ i = 1, ..., k \tag{9}$$

for all $v \in L_2[0, T]$ and $w = \Delta(v)$ over the interval $[0, T]$. Hence, the multiplier of the i^{th} IQC is are parameterized with the strict positive real number λ_i. The IQCs in (9) can be stacked in a single IQC defined by

$$\Psi = \begin{bmatrix} \Psi_1 \\ \vdots \\ \Psi_k \end{bmatrix} \text{ and } M(\lambda) = \begin{bmatrix} \lambda_1 M & & \\ & \ddots & \\ & & \lambda_k M \end{bmatrix} \tag{10}$$

Consequently, the design variable in this example would be $\lambda \in \mathbb{R}^k_+$, i.e. a vector consisting k strict positive real numbers arranged in form of $M(\lambda) \in \mathcal{M}$ describing the IQC parameterization.

The choice of Ψ and M directly effects the achievable value of γ, see [53, 67]. Hence, in order to obtain the lowest upper bound γ of the worst-case input/output gain of the uncertain interconnection an optimization over the IQC parameterization $M(\lambda)$ given a fixed Ψ must be performed. The nonlinear optimization problem minimizing γ over $M(\lambda) \in \mathcal{M}$ can be written as:

$$\min_{M(\lambda) \in \mathcal{M}} \gamma$$

such that $\forall t \in [0, T]$

$$
\begin{aligned}
&P(T) = 0 \\
&\dot{P}(t) = Q(t,\gamma,M(\lambda)) + P(t)\tilde{A}(t,\gamma,M(\lambda)) + \tilde{A}^T(t,\gamma,M(\lambda))P(t) \\
&\qquad - P(t)S(t,\gamma,M(\lambda))P(t) \\
&R(t,\gamma,M(\lambda)) < 0.
\end{aligned}
\tag{11}
$$

where all matrix functions are defined as in Theorem 1 in Appendix A. In [10], the structure of the problem is exploited to render a computationally efficient algorithm to solve (11) by splitting it in an inner and outer optimization loop. In the inner loop, a simple bisection over the scalar γ is performed for a fixed M, i.e. given M, γ is reduced until the RDE cannot longer be integrated over the full horizon $[0, T]$. Tight bounds for the bisection are readily available such that the bisection can be performed quickly. In contrast, the outer loop optimization, i.e. finding $M(\lambda) \in \mathcal{M}$ for which γ is a global minimum, is a hard, nonlinear optimization problem. It is non-smooth and non-convex and further identifying a good initial guess for it is hard. Hence, a MH appears well suitable to solve (11).

3. Adaptive boundary sine cosine optimiser with population reduction

Mostly, MHs have been developed to deal exclusively with bound constrained optimizations. The optimization problem in (11) is more complicated with respect to the design variables. In general, the elements of the design vector $\lambda \in \mathbb{R}^{n_\lambda}$ building the IQC parameterization $M(\lambda)$ are not bounded. They are only constrained by the by the structure imposed by $M(\lambda) \in \mathcal{M}$, which in fact renders certain value combinations infeasible. Additionally, the optimization problem (11) includes two nonlinear constraints. Firstly, the strict negative definiteness of R due to Theorem 1 in Appendix A. Secondly, the solvability of the RDE (7), which is directly related to the existence of the optimized upper bound γ. Hence, the direct application of an existing MH is not feasible.

Therefore, a novel adaptive boundary sine cosine algorithm with population size reduction (Ab-SCA-PR) is introduced. It is specifically designed to deal with the arbitrarily large space, but also to work robustly under the nonlinear constraints and exploit attained information on the LTV IQC analysis problem wherever possible. Its basic search procedure is based on the original sine cosine algorithm (SCA) introduced in [46]. Sine-based updates in the population iterations were already proposed and demonstrated sucessfully in [20]. Similarly to most MHs, the Ab-SCA-PR contains three main steps, namely initialization, reproduction (based on sine and cosine functions), and selection phase. It is extended with an adaptive bound technique to deal with the large search space. Additionally, a population reduction is included to avoid extensive cost function evaluations late in the search. These modifications differentiate the proposed algorithm from the original SCA implementation in [46]. These are generally computationally expensive due to the bisection. In Algorithm 1, pseudo-code of the Ab-SCA-PR's implementation is presented.

Before the algorithm is executed, the user needs to provide the maximum population size $N_{\mathrm{p,max}}$, the maximum number of population iterations $N_{\mathrm{i,max}}$, the number of decision variables n_λ, maximum number of unsuccessful reproductions $U_{\max}$, lower and upper bound of the bisection γ_{l} and γ_{u}, the vector λ_{u} with the initial upper bounds of the search space, the vector $\lambda_{\max}$ containing the maximum upper bounds of the search space, the extended LTV system G_{ext} containing the fixed IQC filter Ψ , and last the nonlinear constraint of the search space $\mathcal{M}$ providing the structure and properties of $M(\lambda)$. Note, that good estimates for the bisection's lower and upper bound are available either from the nominal systems analysis or previous optimization runs, see e.g., [10]

The main algorithm starts with generating a random initial population P. It describes a set of l solution vectors $\lambda^l \in \mathbb{R}^{n_\lambda}$ written as:

$$P = \{\lambda^1, \lambda^2, ..., \lambda^{N_{\mathrm{p,max}}}\}. \tag{12}$$

The elements of λ^l build the respective IQC paramterization $M(\lambda)$. Hence, it must be assured that $M(\lambda^l) \in \mathcal{M}$. In case of the IQC parameterization given in Example 2 the elements of λ need to be strict positive scalars. Hence, it is sufficient to define the search space for each element of λ^l as $\lambda_k^l \in (0, \lambda_{\max,k}]$, where $\lambda_{\max,k}$ is a sufficiently high upper limit. While an initial $\lambda_{\max}$ has to be specified, it will be adapted during the search if necessary. This allows for a narrowed initial search space, e.g. exploiting information from previous optimizations without confining the optimization to it. More complex parameterizations as shown e.g. in [67] can also be considered in the algorithm.

After it is assured that $M(\lambda) \in \mathcal{M}$, the minimal value of γ for each λ^l in the initial population $\gamma(M(\lambda^l))$ is calculated using a bisection constrained by the solvability of the RDE (7). Note that due to numerical reasons it is possible that for a given $M(\lambda^l)$ no finite γ exists such that the RDE is fully solvable. In this case, $\gamma(M(\lambda))$ is set to 10^{20}. As R is a function of the bisected γ and $M(\lambda^l)$, the $R < 0$ condition is evaluated in the bisection. In case $R(M(\lambda^l), \gamma)) \geq 0$, the respective γ bisection step is treated similarly to a not fully solvable RDE. Note that in most cases $R < 0$ is automatically fulfilled for a valid $M(\lambda) \in \mathcal{M}$.

Subsequently, the current best solution λ_{best} is identified. Now, the iteration

Algorithm 1 Adaptive boundary sine cosine optimiser with population reduction

```
1:  **Input:** $N_{\text{p,max}}$, $N_{\text{i,max}}$, $n_\lambda$, $U_{\max}$, $\gamma_{\text{l}}$, $\gamma_{\text{u}}$, $\lambda_{\max}$, $\lambda_u$, $G_{\text{ext}}$, $\mathcal{M}$
2:  **Output:** $\lambda_{\text{best}}$, $\gamma_{\text{best}}$
3:  Generate random initial population $P$ and build respective $M(\lambda^l) \in \mathcal{M}$
4:  Calculate $\gamma(M(\lambda^l))$ via bisection ($\gamma_{\text{l}}$/$\gamma_{\text{u}}$ fixed) constrained by solvability of
    the RDE over $[T, 0]$, for $P(T) = 0$, treat $R \geq 0$ as failed integration
5:  Find the best solution $\lambda_{\text{best}}$ and initialize $U = 0$
6:  **for** $N_i = 1$ to $N_{i,\max}$ **do**
7:      Calculate parameter $r_1$ based on (14)
8:      **for** $l = 1$ to $N_p$ **do**
9:          **for** $k = 1$ to $N_\lambda$ **do**
10:             Randomly generate the parameter $r_2$,$r_3$ and $r_4$ in the ranges of
                $[0, 2\pi]$, $[0, 2]$ and $[0, 1]$, respectively
11:             Update the $k^{th}$ element of the $l^{th}$ solution ($\lambda_l$) based on (13)
12:         **end for**
13:         Build $M(\lambda^l) \in \mathcal{M}$
14:         **if** RDE solvable for $\gamma^l_{N_i-1}$ and $M(\lambda^l)$ **then** Execute bisection with
            $\gamma_u = \gamma^l_{N_i-1}$ calculating $\gamma(M(\lambda^l))$, handle $R \geq 0$ as failed integration
15:         **else** Skip bisection, treat $\lambda^l$ as failure
16:         **end if**
17:     **end for**
18:     Find $\lambda_{\text{best,new}}$
19:     **if** $\gamma(M(\lambda_{\text{best,new}})) < M(\gamma(\lambda_{\text{best}}))$ **then** $\lambda_{\text{best}} = \lambda_{\text{best,new}}$ and set $U = 0$
20:     **else** $U = U + 1$
21:     **end if**
22:     **if** $U > U_{\max}$ **then** Update search bounds via (15), reset $U$ to 0
23:         Generate $N_{\text{add}}$ solutions in $\mathcal{M}$ using LHS for new bounds
24:         Remove all solutions located in the old bounds
25:         Apply the $k$-mean clustering technique to group the remaining
            solutions into $N_{\text{add}}$ groups and find the centroid solutions
26:         Calculate $\gamma$ values of the centroid solutions of each group via bisection
            ($\gamma_{\text{l}}$/$\gamma_{\text{u}}$ fixed) and save to current population if they are better than the
            worse solution in the population
27:     **end if**
28:     Update population size via (16) and remove the worst solutions
29: **end for**
```

starts with the reproduction process updating each design solution in the population via

$$\lambda^l_{\text{new},k} = \begin{cases} \lambda^l_{\text{old},k} + r_1 \sin(r_2) \left| r_3 \lambda_{\text{best},k} - \lambda^l_{\text{old},k} \right|, & \text{if } r_4 < 0.5 \\ \lambda^l_{\text{old},k} + r_1 \cos(r_2) \left| r_3 \lambda_{\text{best},k} - \lambda^l_{\text{old},k} \right|, & \text{otherwise} \end{cases}, \tag{13}$$

where $\lambda^l_{\text{new},k}$, $\lambda^l_{\text{old},k}$, and $\lambda_{\text{best},k}$ are the k^{th} vector element of a newly generated solution for λ^l, a current solution and the current best solution of the population, respectively. The parameters r_2, r_3, and r_4 are uniformly randomized for each iteration in the ranges of $[0, 2\pi]$, $[0, 2]$ and $[0, 1]$, respectively. The parameter r_1 is an iterative adaption using

$$r_1 = a - N_\text{i} \frac{a}{N_{\text{i,max}}}, \tag{14}$$

where N_i is the current iteration and a is a predefined constant. Again, it has to be assured that the updated $M(\lambda^l_\text{new}) \in \mathcal{M}$. As $M(\lambda^l_\text{new}) \notin \mathcal{M}$ for Example 2 is a simple boundary infraction, the respective $\lambda^l_{\text{new},k}$ are simply set to their closest boundary value.

Before the bisection for the updated $\lambda^l_{\text{new},k}$ is conducted, it is checked if the RDE (7) is solvable for $M(\lambda^l)$ and the minimal $\gamma^l_{N_\text{i}-1}$ value calculated for its respective parent . If this is not the case, the bisection can be skipped as the offspring is no improvement over its parent and can be treated as failure, i.e. it is assigned a value of 10^{20}. This avoids unnecessary evaluations of the RDE and thus, significantly reduces the computational effort. In Section 4, it will be shown that the accompanying reduction of search information does not come at the cost of search performance.

If the RDE (7) is fully solvable for $M(\lambda^l)$ and $\gamma^l_{N_\text{i}-1}$, the bisection is conducted using $\gamma^l_{N_\text{i}-1}$ as (the new) upper bound γ_u. Using this adaptive upper bound, significantly narrows the bisection interval compared to the initial population's or the one used after a boundary extension, which utilize user defined bounds. Consequently, the number of computational expensive RDE integrations is considerably reduced.

After concluding the bisections, the best solution obtained from the newly generated population is compared with its counterpart from the previous iteration. If the best solution of the new population is better, the current best solution is updated, while U is reset to zero. Here, U is a variable that counts the number of unsuccessful iterations. Otherwise, the current best solution is not changed and U is increased by one. If the value of U is higher than a predefined limit, the bounds of the search space are extended. Given the IQC parameterization in Example 2 the upper bound of each design variable is extended by

$$\lambda_{\text{u,new}_k} = \begin{cases} 10\lambda_{\text{u,old}_k} & \text{if } \lambda_{\text{u,old}_k} < \lambda_{\text{max}_k} \\ \lambda_{\text{u,old}_k} & \text{otherwise} \end{cases} \tag{15}$$

whereas the lower bound remains zero. In (15), λ_{u,old_k} and λ_{u,new_k} are the upper bound of the k^{th} element of all design variables λ^l before and after updating, respectively. The maximum allowable upper bounds of the respective elements are given by λ_{max_k}. Note, that this boundary adaption can be easily adjusted to other search spaces.

Since the bounds of the design variables have been extended, a set of additional N_{add} solutions located on/inside the extended boundary must be generated to enhance the search performance of the optimizer. In order to have the solutions well distributed throughout the extended boundary, a Latin Hypercube Sampling (LHS) technique is firstly used to create $50N_{\text{add}}$ solutions throughout the whole boundary of the design variables [41]. Then, all solutions inside the old boundary are removed, while a total of N_{add} (default is 10) solutions are created based on the distribution of the remaining solutions. Here, a k-mean clustering technique is used to group the remaining solutions into N_{add} groups, whereas the centroid of each group is assigned as one of those N_{add} solutions [5, 39]. Having evaluated their objective function values, the solutions are added to the current population. Afterwards, the worst solutions in the population are deleted to restore the correct population size.

For each iteration, after obtaining the current best solution of γ and updating the search space, the population size is reduced based on the following equation

$$N_{\text{p}_{N_\text{i}+1}} = N_{\text{p,max}} - \text{round}\,\frac{N_\text{i}\left(N_{\text{p,max}} - N_{\text{p,min}}\right)}{N_{\text{i,max}}}, \tag{16}$$

where $N_{\text{p}_{N_\text{i}+1}}$ is the population size at iteration $N_\text{i} + 1$. $N_{\text{p,max}}$ and $N_{\text{p,min}}$ are the user-defined maximum and minimum population sizes, respectively. In order words, the proposed algorithm starts with the maximum population size at the initial function evaluation and decreases the population size with increasing function evaluations. If $N_{\text{p}_{N_\text{i}+1}}$ is lower than the current population size, the worst solutions in the current population are removed to match the new population size. This allows to reduce the amount of necessary bisection evaluation at the end of the search when the region of the global minimum is likely identified. At the same time, it also biases the search continuously towards this region promising a better convergence. Subsequently, the reproduction starts again. The search process terminates as soon as the maximum number of iterations $N_{\mathrm{i,max}}$ is reached providing minimal γ_{best} and the corresponding λ_{best}.

The proposed Ab-SCA-PR is slightly more computational complex than the original SCA algorithm due to the introduced boundary extension and population reduction. This increase in complexity can be quantified using Big-O analysis. For the original SCA the overall complexity is O$(N_{\text{p,max}} \times n_\lambda \times N_{i,\text{max}})$, calculated from the initialization (O$(N_{\text{p,max}} \times n_\lambda)$), reproduction (O$(N_{\text{p,max}} \times n_\lambda \times N_{i,\text{max}})$), and selection process (O$(N_{\text{p,max}} \times N_{i,\text{max}})$). In case of the proposed Ab-SCA-PR, the additional boundary adaptation process including LHS (O$(10N_{\text{add}})$) as well as K-mean clustering (O$(10N_{\text{add}} \times N_{\text{add}})$), and the population reduction process ($O(N_{\text{p,max}} - N_{\text{p,min}})$) contribute to the computational complexity. This results in an overall complexity of O$(N_{\text{p,max}} \times n_\lambda \times N_{i,\text{max}} + N_{\text{add}}^2)$. However, this increase in complexity (i.e., the additional steps) is insignificant regarding its additional computational time compared to the time saving resulting from reducing the overall amount of expensive objective function evaluations.

4. Numerical Experiment

In the final section, the worst case optimization problem (11) as presented in a benchmark example is solved using the Ab-SCA-PR algorithm proposed in Section 3 and thirteen existing MHs. The benchmark example is taken from [10], where the robust performance of a small space launcher under wind disturbance is analyzed. The Matlab implementation of the numerical experiment as well as additionally tested solver are provided on https://github.com/tudfmr/LTViqcMHs.

In Fig. 4, the corresponding analysis interconnection is displayed. Here, the

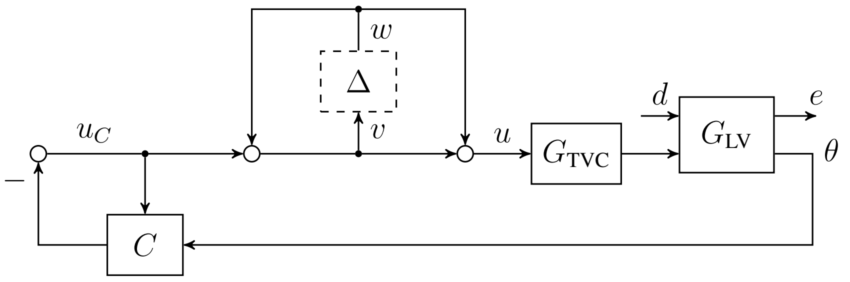

Figure 4: Analysis Interconnection

block G_{LV} represents the space launcher's LTV dynamics. These are similar to the ones introduced in Example 1 represented by equation (1). The associated numerical values of the matrix coefficients can be found in [66], which describe the Vanguard space launcher, a small expandable launch vehicle from the times of early space exploration. The launcher is controlled by a thrust vector control (TVC) represented by G_{TVC}. Its dynamics are described by

$$G_{\text{TVC}} = \frac{50}{s+50} \tag{17}$$

The respective control signals are calculated by a linear quadratic regulator (LQR) including an observer based on pitch angle feedback. The corresponding gains are calculated in accordance to [66]. In Fig. 4 the controller is represented by C. To evaluate the robustness of the system towards phase and gain perturbations/uncertainties, the interconnection is extended with a norm bounded dynamic LTI uncertainty Δ, with $\|\Delta\|_\infty \leq b$. This specific implementation of Δ as shown in Fig. 4 is chosen to mimic input LTI disk margins. These are commonly used in flight control validation, e.g. [11]. Here, the norm bound b directly corresponds to simultaneous phase and gain disturbances' maximum value. Hence, analyzing the interconnection in Fig. 4 for increasing b allows to identify the maximum tolerable disturbance. In total eight values of b in the range from 0.01 to 0.085 are evaluated. The performance of the launcher is quantified by its worst case finite time induced $L_2[0,T]$ gain γ from the wind disturbance $d = \delta_\alpha$ to the angle of attack $e = \alpha$.

This requires a conversion of the analysis interconnection in Fig. 4 into the IQC framework as described in Section 2.3. Here, the conic combination of two IQCs is chosen to cover the input/output behavior of the dynamic LTI uncertainty Δ following Example 2. The first IQC, denoted by $IQC_1(\Psi_1, M_1)$, is factorized with $\Psi_1 = I_2$ and parameterized by $M_1(\lambda_1) = \lambda_1 M$, with $M = \left[\begin{smallmatrix} b^2 & 0 \\ 0 & -1 \end{smallmatrix}\right]$. For the second IQC, denoted by $IQC_2(\Psi_2, M_2)$, the factorization is chosen as $\Psi_2 = \frac{1}{s+1} I_2$ with the parameterization $M_2 = \lambda_2 M$. The scalar factors λ_1 and λ_2 are strict positive. Subsequently, $IQC_1(\Psi_1, M_1)$ and $IQC_1(\Psi_1, M_1)$ stacked into a single IQC, as described in Example 2. This results in a single IQC with factorization $\Psi = [\Psi_1^T, \Psi_2^T]^T$ and parameterization $M(\lambda) = \left[\begin{smallmatrix} \lambda_1 M & \\ & \lambda_2 M \end{smallmatrix}\right]$. Consequently, the optimization problem (11) described in Section 2.4 is solved over the two decision variables λ_1 and λ_2 identifying the worst case gain γ. The analysis is repeated for increasing values of b. Note, that the RDE is solved using the Matlab internal solver `ODE15s` which is a variable step size solver for stiff differential equations. Over a finite time interval theoretically all signals remain bounded. Thus for any value b there should exist a finite value γ. In practice, it is possible that the finite escape times of the RDE ([1]) is always smaller than considered analysis horizon for a given b. Consequently, no γ value can be calculated in this case. Therefore, it is possible that some solvers cannot find a valid γ for all executed test cases in the presented analyses.

4.1. Solver and Benchmark Setup

A total of eighteen MHs are evaluated against the proposed Ab-SCA-PR. Each is solving the worst-case gain optimization of the space launcher in five independent runs. All solvers are initialized with a population size of 50. Solvers with fixed population sizes are terminated after 50 generations. In case of an adaptable population size, the algorithm is terminated after a total of 2500 (50×50) objective function evaluations. All optimizers except the Ab-SCA-PR will use a lower bound and upper bound of $1 \cdot 10^{-6}$ and $1 \cdot 10^{8}$, respectively for both λ_1 and λ_2. As the proposed Ab-SCA-PR algorithm utilizes self-adaptive upper bounds, the optimizer can be initialized with a significantly tighter search space. Therefore, the initial upper bound of both decision variables is set to 100. All meta-heuristic specific optimization parameter settings are listed below.

1. Differential evolution (DE) [55]: DE/best/2/bin strategy was used. A scaling factor, crossover rate and probability of choosing elements of mutant vectors are 0.5, 0.7, and 0.8 respectively
2. Adaptive differential evolution (JADE) [74]: The parameters are self-adapted during an optimization process.
3. Success-history based adaptive differential evolution (SHADE) [64]: The parameters are self-adapted during an optimization process.
4. SHADE with Linear Population Size Reduction (L-SHADE) [65] : The parameters are self-adapted during an optimization process.
5. Neuro-dynamic Differential Evolution Algorithm (L-SHADE-ND) [60]: The parameters are self-adapted during an optimization process.
6. L-Shade with Eigenvector-Based Crossover and Successful-Parent-Selecting Framework (SPS-L-SHADE-EIG) [28]: The parameters are self-adapted during an optimization process.
7. Whale optimization algorithm (WOA) [47]: The code and parameters are provided by the authors of the algorithm.
8. Moth-flame optimization algorithm (MFO) [43]: The code and parameters are provided by the authors of the algorithm.
9. Dragonfly Algorithm (DA) [45]: Default parameter settings from original code by [45] are used in this study.

10. Grey Wolf Optimiser (GWO) [48]: Default parameter settings from original code by [48] are used in this study.
11. Sinusoidal differential evolution (SinDE) [20]: Default parameter settings from original code by [20] are used in this study.
12. Particle swarm optimization algorithm (PSO) [22]: The starting inertia weight, ending inertia weight, cognitive learning factor and social learning factor are set to be 0.5, 0.01, 2.8, and 1.3, respectively.
13. Genetic algorithm (GA) [18]: The crossover and mutation probability are set to be 1 and 0.1, respectively.
14. Heap-based optimizer (HBO) [6]: The code and parameters are provided by the authors of the algorithm.
15. Harris hawks optimization (HHO) [32]: The code and parameters are provided by the authors of the algorithm.
16. Sine cosine algorithm (SCA) [46] (Algorithm 1): The constant a parameter is set to be 2.
17. Improved sine cosine algorithm with crossover scheme (ISCA) [31]: The constant a parameter is set to be 2 while crossover rate is set to be 0.3.
18. Modified sine cosine algorithm (m-SCA) [29]: The constant a parameter, crossover rate and jumping rate are set to be 2, 0.3 and 0.1, respectively.
19. Adaptive boundary sine cosine optimizer with population reduction (Ab-SCA-PR) (Algorithm 1): Used the same parameter settings as SCA.

Note that all optimizations are run with an adaptive upper bound of the bisection as well as skipping the bisection altogether if the offspring shows no improvement (see Section 3 for details). However, the lower bound γ_l remains fixed and the initial population is evaluated fully with a fixed upper bound γ_u. The corresponding values for the different evaluated b are taken from [10] and summarized in Table 1. Note that for $b = 0.01$, γ_1 equals the nominal worst case gain and γ_u is a factor of ten higher. The subsequent b use a γ_l and γ_u of 0.8 and 10 times the worst case γ of the previous norm bound, respectively.

4.2. Results and Discussion

Four criteria are applied to evaluate the search performance of the MHs. Firstly, the absolute search performance based on the lowest cost function value γ_{best}. Secondly, the mean value of the worst case gain μ_γ over the five optimization runs. It is used to measure the convergence rate and consistency of the algorithms. These are indicators for the reliability of the algorithm, which is essential for its industrial applicability. This is further emphasized by the third criterion, the number of

Table 1: Lower bound γ_l and upper bound γ_u used for the bisection for a given b based on [10]

Norm bound b	Lower bound γ_l	Upper bound γ_u
0.01	1.1527	11.527
0.03	1.8837	23.5460
0.05	2.2529	28.1610
0.06	2.8194	35.2420
0.07	3.2229	40.2860
0.075	3.7438	46.7970
0.08	4.2086	52.6070
0.085	5.3578	66.9730

successful runs n_{feas}. In case two algorithms have the same number of successful runs, the standard deviation σ_γ of the worst case gain is used to measure the search consistency instead. It should be noted that, only algorithms which can find feasible solutions in more than one optimization run are considered for the μ_γ and σ_γ value comparison.

In Figure 5, the Ab-SCA-PR's the mean value as well as the variance of γ over b is shown. It is compared to the existing optimizer with the most successful runs, namely the GWO, and the original SCA. For all cases, the proposed algorithm has a low variance σ_γ in the worst case gain. The two existing MHs cannot provide multiple solutions for all values of b. Furthermore, they show a higher variance as well as higher mean values μ_γ. This is especially visible for the SCA for $b = 0.03$ and $b = 0.07$. Hence, the proposed Ab-SCA-PR is visibly the most reliable and consistent optimizer for the LTV robustness analysis problem. This is further emphasized by evaluating the detailed results for each algorithm summarized in Table 2. Note that results for the remaining b can be found in Table B.5 in the appendix. Focusing on absolute search performance, the proposed algorithm is the best performer for norm bounds of 0.03, 0.05, and 0.085. For the lowest norm bound $b = 0.01$, it reaches the sixth place, with the LSND providing the lowest γ_{best}. Given $b = 0.07$, the proposed algorithm provides the third lowest γ. Nevertheless, the Ab-SCA-PR's γ_{best} is always equivalent to at least the second decimal of the respective best algorithm's.

With respect to search convergence, the proposed Ab-SCA-PR is the best performer for the cases of $b = 0.03$, $b = 0.05$ as well as $b = 0.085$ and is further the runner-up for 0.07. Given $b = 0.01$, the best algorithms in this category are the

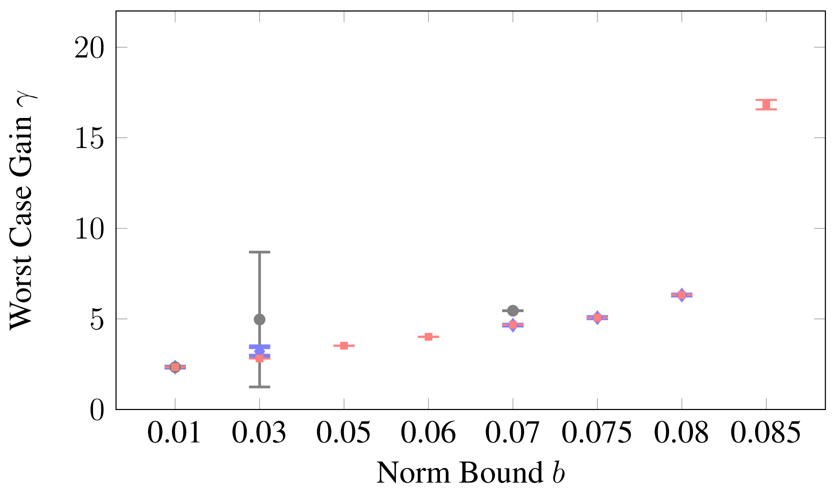

Figure 5: Mean values and variance of the top three optimizers: Ab-SCA-PR (■), GWO (♦), SCA (●)

GWO and the WOA, with the Ab-SCA-PR ranking fourth. For a norm bound of 0.07, the GWO is the best optimizer. The runner-ups for $b = 0.03$ and $b = 0.05$ are the GWO and the LSHADE-ND, respectively. The third best algorithm for $b = 0.01$ and $b = 0.03$ is the SCA, while the third best method for $b = 0.07$ is the WOA.

With respect to the search consistency, the best performer across all norm bounds is the proposed Ab-SCA-PR achieving 100% success rate. All other algorithms show a deterioration of search consistency for increasing values of the norm bound b. Given the lowest norm bound of 0.01 the overall results are still good, with five algorithms (SCA, DA, GWO, WOA, HHO and mSCA) reaching 100% success rate, but none of these algorithms achieved a standard deviation as low as the Ab-SCA-PR ($\sigma_\gamma = 0.0004$). The second lowest standard deviation was achieved by the SCA with $\sigma_\gamma = 0.007$ and the highest by the HBO with $\sigma_\gamma = 4.4961$. Two other algorithms, the LSND and the ISCA, achieved 4 successful runs, with the ISCA performing worse overall having a ten times higher standard deviation. Another algorithm concluding 2 successful runs was the DE and HBO. Finally, a total of 7 algorithms cannot identify a valid γ in any run. Increasing the norm bound to 0.03, other than the proposed algorithm, only the GWO and the SCA have more than one positive run, with four and three, respectively. Although, their standard deviations are significantly worse than the Ab-SCA-PR's 0.0013 with 0.2581 and 3.7230 achieved by the GWO and the SCA, respectively.

Apart from that, just the WOA and HHO can identify a solution at all for this norm bound As the norm bound is increased to 0.05, besides the Ab-SCA-PR only the LSND has multiple successful runs with two each with $\gamma = 3.5668$. Although, this γ value is slightly higher than the Ab-SCA-PR's attained $\mu_\gamma = 3.5213$. Apart from that, just three other algorithms (GWO, HHO and WOA) concluded successfully, leaving fourteen algorithms not delivering results. For $b = 0.07$, beyond the proposed algorithm the SCA, WOA and GWO had multiple successful runs with the first two having three and the last two successes. All of these algorithms reached a standard deviation of zero, but only the GWO calculated a γ less than the Ab-SCA-PR's mean value. Furthermore, Ab-SCA-PR still achieves a very low $\sigma_\gamma = 0.0040$. Given the maximum norm bound of $b = 0.085$, only the HHO other than the Ab-SCA-PR found a valid solution. However, it only had one successful run, with $\gamma = 20.009$ which is significantly worse than the mean value achieved by the Ab-SCA-PR ($\mu_\gamma = 16.831$). Also taking the test cases in Table B.5 into consideration, only the proposed Ab-SCA-PR could be exclusively used for all b. Consequently, the user would have to change the solvers or their settings depending on b, without indication of their suitability. This is infeasible for industrial application. Moreover, the proposed algorithm is generally superior for high values of b. This is especially important to evaluate the worst case performance. It can be concluded, that the proposed Ab-SCA-PR shows the best overall search performance. Hence, the integration of boundary adaptation as well as population reduction technique into the SCA significantly increased the algorithms suitability for LTV worst case analysis.

Table 2: Optimization results for adaptive bisection bounds (n.f. = not feasible, and N/A = not available, S = SHADE, LS = L-SHADE, LSND = L-SHADE-ND, SPS = SPS-L-SHADE-EIG, New = Ab-SCA-PR)

b		DE	JADE	S	LS	LSND	SPS	SinDE	SCA	DA	GWO	MFO	GA	PSO	HBO	HHO	WOA	ISCA	mSCA	New
0.01	γ_{best}	2.3383	n.f	6.6080	n.f.	2.3346	n.f.	n.f.	2.3349	2.3436	2.3350	n.f.	n.f.	n.f.	3.5243	2.3360	2.3350	2.3887	2.3370	2.3365
	μ_γ	2.3383	N/A	N/A	N/A	2.3406	N/A	N/A	2.3362	3.5430	2.3361	N/A	N/A	N/A	6.7036	2.3385	2.3361	2.3764	2.3377	2.3372
	σ_γ	0	N/A	N/A	N/A	0.0020	N/A	N/A	0.0007	0.6704	0.0016	N/A	N/A	N/A	4.4961	0.0015	0.0010	0.0246	0.0010	0.0004
	n_{feas}	2	0	1	0	4	0	0	5	5	5	0	0	0	2	5	5	4	5	5
0.03	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	2.8172	n/A	2.8181	n.f.	n.f.	n.f.	n.f.	2.8206	2.8177	n.f.	n.f.	2.8140
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.9667	N/A	3.2052	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2.8156
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.7230	N/A	0.2581	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0013
	n_{feas}	0	0	0	0	0	0	0	3	0	4	0	0	0	0	1	1	0	0	5
0.05	γ_{best}	n.f.	n.f	n.f.	n.f.	3.5668	n.f.	n.f.	n.f.	n.f.	8.2731	n.f.	n.f.	n.f.	n.f.	3.5233	3.5212	n.f.	n.f.	3.5204
	μ_γ	N/A	N/A	N/A	N/A	3.5668	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.5213
	σ_γ	N/A	N/A	N/A	N/A	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0005
	n_{feas}	0	0	0	0	2	0	0	0	0	1	0	0	0	0	1	1	0	0	5
0.07	γ_{best}	4.6686	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	5.4528	n.f	4.6552	n.f.	n.f.	n.f.	n.f.	n.f.	4.6818	n.f.	n.f.	4.6737
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.4528	N/A	4.6552	N/A	N/A	N/A	N/A	N/A	4.6818	N/A	N/A	4.6796
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	N/A	0	N/A	N/A	N/A	N/A	N/A	0	N/A	N/A	0.0040
	n_{feas}	1	0	0	0	0	0	0	3	0	2	0	0	0	0	0	3	0	0	5
0.085	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	20.009	n.f.	n.f.	n.f.	16.679
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	16.831
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.2606
	n_{feas}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	5

4.3. Effects of the Bisection Adaptations

The modifications of the bisection, namely an adaptive upper bound and skipped evaluations, reduce the search space exploration. Hence, their influence on the search performance is investigated in this subsection. Accordingly, the analyses of the previous section are repeated using fixed upper and lower bounds in the bisection as provided in Table 1. Furthermore, no bisections of the offspring are skipped. Hence, the optimizer is provided with more search information at the cost of significantly increased computational effort.

In Fig. 6 the mean value as well as the variance of γ over b of the three most consistent optimizers, namely the Ab-SCA, the SCA, and GWO are compared. The increase in search information shows only an insignificant improvement on the search performance of the Ab-SCA-PR compared to the preceding evaluation. Only for higher values of b the σ_γ is slightly reduced, with no visible effect on the achieved μ_γ. The marginal improvement in search performance requires significantly more computational effort. Compared to the preceding evaluations, on average twice as much time was required for the same number of function evaluations. More successful solutions, over a wider range of b are identified by the SCA. The variance of γ is reduced and the achieved mean values are closer to the Ab-SCA-PR. In the contrary, an adverse effect on the search performance of the GWO with respect to μ_γ and σ_γ is evident. It only has multiple successful runs for four values of b, which is one less than in the preceding analysis. Hence, the

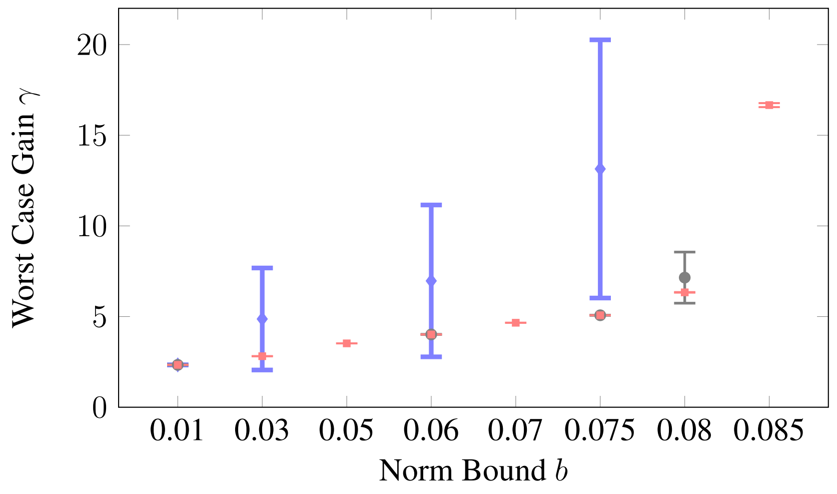

Figure 6: Mean values and variance of the top three optimizers without bisection adaptations: Ab-SCA-PR (■), GWO (♦), SCA (●)

existing optimizers are highly sensitive towards alterations of the search information. In the contrary, the proposed Ab-SCA-PR is significantly more robust. This becomes more evident comparing the detailed results for all solvers as provided in Table 3 with the preceding results in Table 2. Note that the results for the remaining b can be found in Table B.6 in the appendix. The γ_{best} over all b of the Ab-SCA-PR are reduced on average by only -0.25%. Most of the existing solvers show more improvement with respect to the absolute search performance compared to the Ab-SCA-PR, especially for higher values of b. Here, the most significant improvements showed the GWO with -58% for $b = 0.05$ and the MFO with -53% for $b = 0.08$. Nevertheless, some solvers performed significantly worse, e.g. the γ_{best} of the DE was increased by 262% for $b = 0.06$. Consequently, the extended search information and increased computational effort do not guarantee an improvement of γ_{best} for the existing solvers.

Focusing on the search convergence (μ_γ), only the proposed algorithm showed an improvement for all b. Although, it is insignificantly averaging -0.2%, with a maximum improvement of 1% for $b = 0.085$. The existing algorithms show the same indifferent behavior as for the absolute search performance. Here, the DA showed the most improvement with -34.03% for $b = 0.01$, whereas the GWO showed the most degradation with 159% for $b = 0.075$.

With respect to the search consistency the best algorithm remains the Ab-SCA-PR achieving a 100% success rate for all analyzed norm bounds. Despite the significantly increased computational effort, none of the already existing solvers showed consistent search behavior over all b. Actually, the total number of feasible runs over all optimizations dropped from 137 to 124. In general, the analyses required nearly twice as much time as in the preceding section given the same number of function evaluations.

Concluding, the Ab-SCA-PR allows to fully exploit the bisection modification without almost no cost for the performance criteria. The existing solvers perform at times significantly better without these modifications. However, they are still significantly less consistent than the proposed solver with the modifications, which is then also much faster.

Table 3: Optimization results for fixed bisection bounds (n.f. = not feasible, and N/A = not available, S = SHADE, LS = L-SHADE, LSND = L-SHADE-ND, SPS = SPS-L-SHADE-EIG, New = Ab-SCA-PR)

b		DE	JADE	S	LS	LSND	SPS	SinDE	SCA	DA	GWO	MFO	GA	PSA	HBO	HHO	WOA	ISCA	mSCA	New
0.01	γ_{best}	2.3351	n.f	6.6080	n.f.	2.3307	n.f.	n.f.	2.3348	2.3361	2.3345	2.3375	n.f.	n.f.	3.6556	2.3345	2.3343	2.3376	2.3374	2.3360
	μ_γ	2.3357	N/A	N/A	N/A	2.3463	N/A	N/A	2.3365	2.3373	2.3355	2.3376	N/A	N/A	4.8721	2.3352	2.3350	2.3958	2.4481	2.3365
	σ_γ	0.0005	N/A	N/A	N/A	0.0037	N/A	N/A	0.0014	0.0011	0.0008	0.0001	N/A	N/A	2.4331	0.0009	0.0010	0.1297	0.1565	0.0004
	n_{feas}	4	0	1	0	4	0	0	4	3	4	2	0	0	4	5	2	5	2	5
0.03	γ_{best}	2.8152	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	2.8156	2.8129	n.f.	n.f.	n.f.	n.f.	2.8140	n.f.	2.8161	3.4976	2.8130
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.8670	N/A	N/A	N/A	N/A	2.8142	N/A	N/A	N/A	2.8146
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2.8127	N/A	N/A	N/A	N/A	0.0003	N/A	N/A	N/A	0.0011
	n_{feas}	1	0	0	0	0	0	0	0	1	5	0	0	0	0	2	0	1	1	5
0.05	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	3.5216	2.5236	3.5199	n.f.	n.f.	n.f.	n.f.	3.5199	n.f.	n.f.	n.f.	3.5199
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.5213
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0013
	n_{feas}	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	5
0.07	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	4.6743	n.f	12.599	n.f.	n.f.	n.f.	n.f.	n.f.	4.6484	n.f.	n.f.	4.6570
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.6593
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0036
	n_{feas}	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	5
0.085	γ_{best}	16.210	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	16.478
	μ_γ	17.526	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	16.669
	σ_γ	0.9668	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.1111
	n_{feas}	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5

Table 4: Comparison of the best results obtained with Ab-SCA-PR in this study with the best results from the algorithm used in the previous work

b	Results [10]	Ab-SCA-PR (fixed bounds)	Ab-SCA-PR (adaptive bounds)
0.01	2.3546	2.3360	2.3365
0.03	2.8161	2.8130	2.8140
0.05	3.5242	3.5199	3.5204
0.06	4.0286	4.0071	4.0127
0.07	4.6797	4.6570	4.6737
0.075	5.2607	5.0733	5.0718
0.08	6.6973	6.2975	6.3140
0.085	n.f.	16.6624	16.6789

4.4. *Comparison to Original Benchmark*

The evaluation of the proposed Ab-SCA-PR is concluded by the comparison of the best results obtained in this study with the best results from the algorithm used in [10] shown in Table 4. When comparing the results with the previous work [10], the proposed MH (Ab-SCA-PR) gives better results than those obtained by the optimizer in [10] using the same fixed bisection intervals. The improvement becomes more significant for increasing values of b. For $b = 0.085$, the optimization in [10] cannot find a feasible solution. Note that the algorithm in [10] employed a local gradient free search and is highly dependent on an initial guess. It required a significant background research to find good initial estimates for λ_1 and λ_2. This is highly undesirable in an industrial application, as the optimizer shall be deployable with as little prior knowledge as possible. Furthermore, the applied solver in [10] can by the nature of its search strategy neither exploit adaptive bisection bounds nor the avoidance of bisections.

5. Conclusion

In this work, an adaptive boundary sine cosine optimizer with population reduction is proposed for worst case gain optimization of finite horizon LTV systems. The optimization problem is set to find an IQC parameterization in order to minimize the worst-case gain's upper bound. Several existing MHs and the proposed algorithm are used to solve the problem. The comparison results shown that the proposed algorithm is the best performer in every category. Using the reduction of γ_u during the MHs search process lead to less computational time, however, the optimum results obtained are slightly worse. For this numerically hard optimization problem, using adaptive boundary techniques and population reduction leads to significant enhancement of SCA. However, the proposed technique might be suitable for the LTV IQC problem, while its performance on other large design domain problems requires further investigation. This work forms the baseline of applying MHs to IQC based worst-case gain optimization of finite horizon LTV systems of industrial complexity. The applicability is demonstrated on the example of the robustness analysis of a space launcher under a disk margin-type uncertainty.

Acknowledgements

The authors are grateful for the financial support by the National Research Council of Thailand (NRCT5-RSA63003-06). Furthermore, the authors would like to thank the European Space Agency for financially funding this work through the Networking/Partnering Initiative contract No. 4000123233.

Appendix A. Strict Bounded Real Lemma for IQCs

The strict bounded real lemma as used to derive the optimization problem in this paper, is based on a dissipation inequality expressed as an equivalent RDE formulation. It is given in the following Theorem:

Theorem 1. *Let $F_u(G_t, \Delta)$ be well posed $\forall\, \Delta \in IQC(\Psi, M)$, then $\|F_u(G_t, \Delta)\|_{2[0,T]} < \gamma$ if there exist a continuously differentiable $P : \mathbb{R}_0^+ \to \mathbb{S}^{n_x}$ such that*

$$P(T) = 0, \tag{A.1}$$

$$\dot{P} = Q + P\tilde{A} + \tilde{A}^T P - PSP \qquad \forall t \in [0, T] \tag{A.2}$$

and

$$R = \begin{bmatrix} D_{11}^T M D_{11} + D_{21}^T D_{21} & D_{11}^T M D_{12} + D_{21}^T D_{22} \\ D_{12}^T M D_{11} + D_{22}^T D_{21} & D_{12}^T M D_{12} + D_{22}^T D_{22} - \gamma^2 I \end{bmatrix} < 0, \tag{A.3}$$

with

$$\tilde{A} = \begin{bmatrix} B_1 & B_2 \end{bmatrix} R^{-1} \begin{bmatrix} (C_1^T M D_{11} + C_2^T D_{21})^T \\ (C_1^T M D_{12} + C_2^T D_{22})^T \end{bmatrix} - A, \tag{A.4}$$

$$S = -\begin{bmatrix} B_1 & B_2 \end{bmatrix} R^{-1} \begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix}, \tag{A.5}$$

$$\begin{aligned} Q = &- C_1^T M C_1 - C_2^T C_2 \\ &+ \begin{bmatrix} (C_1^T M D_{11} + C_2^T D_{21})^T \\ (C_1^T M D_{12} + C_2^T D_{22})^T \end{bmatrix}^T R^{-1} \begin{bmatrix} (C_1^T M D_{11} + C_2^T D_{21})^T \\ (C_1^T M D_{12} + C_2^T D_{22})^T \end{bmatrix}. \end{aligned} \tag{A.6}$$

PROOF. The proof is based on the definition of a time-dependent quadratic storage function $V(x, t) = x^T P(t) x$. After a perturbation of (A.2) the resulting Riccati differential inequality can be reformulated as an equivalent linear matrix inequality by Schur's complement. The left and right multiplication of $[x^T, w^T, d^T]$ and $[x^T, w^T, d^T]^T$ respectively leads to a dissipation inequality whose integration from 0 to T for zero initial conditions provides the upper bound on γ using the nominal formulation of (6).

More details regarding the proof can be found in [62].

Appendix B. Additional Optimization Results

The following tables contain the results for the remaining analyzed norm bounds:

Table B.5: Additional optimization results for adaptive bisection bounds (n.f. = not feasible, and N/A = not available, S = SHADE, LS = L-SHADE, LSND = L-SHADE-ND, SPS = SPS-L-SHADE-EIG, New = Ab-SCA-PR)

b		DE	JADE	S	LS	LSND	SPS	SinDE	SCA	DA	GWO	MFO	GA	PSO	HBO	HHO	WOA	ISCA	mSCA	New
0.06	γ_{best}	4.0143	n.f	n.f.	n.f.	n.f.	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	n.f.	4.0396	4.0129	4.7757	n.f.	4.0127
	μ_γ	4.0143	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.0129	N/A	N/A	4.0131
	σ_γ	0.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	N/A	N/A	0.0005
	n_{feas}	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	1	0	5
0.075	γ_{best}	5.2537	n.f	n.f.	n.f.	n.f.	n.f.	n.f	n.f.	n.f	5.0696	n.f.	n.f.	n.f.	17.9339	5.1412	n.f.	n.f.	11.108	5.0733
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.0741	N/A	N/A	N/A	17.9946	5.3860	N/A	N/A	12.721	5.0735
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0091	N/A	N/A	N/A	0.0858	0.3193	N/A	N/A	0.9312	0.0001
	n_{feas}	1	0	0	0	0	0	0	0	0	4	0	0	0	2	5	0	0	3	5
0.08	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f	n.f.	n.f.	6.3167	13.442	n.f.	n.f.	n.f.	6.3071	n.f.	6.4161	n.f.	6.3140
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	6.3167	N/A	N/A	N/A	N/A	6.5871	N/A	N/A	N/A	6.3238
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	N/A	N/A	N/A	N/A	0.3960	N/A	N/A	N/A	0.0139
	n_{feas}	0	0	0	0	0	0	0	0	0	3	1	0	0	0	2	0	1	0	5

Table B.6: Additional optimization results for fixed bisection bounds (n.f. = not feasible, and N/A = not available, S = SHADE, LS = L-SHADE, LSND = L-SHADE-ND, SPS = SPS-L-SHADE-EIG, New = Ab-SCA-PR)

b		DE	JADE	S	LS	LSND	SPS	SinDE	SCA	DA	GWO	MFO	GA	PSO	HBO	HHO	WOA	ISCA	mSCA	New
0.06	γ_{best}	14.521	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	4.0151	4.0145	4.0066	n.f.	n.f.	n.f.	n.f.	4.0084	4.0070	n.f.	n.f.	4.0071
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.0151	N/A	6.9674	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	4.0105
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	N/A	4.1872	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0032
	n_{feas}	1	0	0	0	0	0	0	2	1	2	0	0	0	0	1	1	0	0	5
0.075	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	5.0705	n.f	5.0511	n.f.	n.f.	n.f.	n.f.	n.f.	5.0624	n.f.	14.359	5.0733
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.0719	N/A	13.143	N/A	N/A	N/A	N/A	N/A	5.0680	N/A	N/A	5.0735
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0020	N/A	7.1196	N/A	N/A	N/A	N/A	N/A	0.0049	N/A	N/A	0.0001
	n_{feas}	0	0	0	0	0	0	0	2	0	3	0	0	0	0	0	3	0	1	5
0.08	γ_{best}	n.f.	n.f	n.f.	n.f.	n.f.	n.f.	n.f.	6.3345	n.f.	n.f.	6.3459	n.f.	n.f.	n.f.	n.f.	6.4081	6.8475	6.3195	6.2975
	μ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	7.1484	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	7.6963	6.3193
	σ_γ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.4097	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.9471	0.0136
	n_{feas}	0	0	0	0	0	0	0	4	0	0	1	0	0	0	0	1	1	2	5

References

[1] Abou-Kandil, H., Freiling, G., Ionescu, V., Jank, G., 2003. Matrix Riccati Equations in Control and Systems Theory. Birkhäuser Basel, doi:10.1007/978-3-0348-8081-7.

[2] Agrawal, P., Abutarboush, H.F., Ganesh, T., Mohamed, A.W., 2021. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). IEEE Access 9, 26766–26791, doi:10.1109/access.2021.3056407.

[3] Ahmadianfar, I., Bozorg-Haddad, O., Chu, X., 2020. Gradient-based optimizer: A new metaheuristic optimization algorithm. Information Sciences 540, 131–159, doi:10.1016/j.ins.2020.06.037.

[4] Al-Qaness, M.A.A., Elaziz, M.A., Ewees, A.A., 2018. Oil consumption forecasting using optimized adaptive neuro-fuzzy inference system based on sine cosine algorithm. IEEE Access 6, 68394–68402, doi:10.1109/access.2018.2879965.

[5] Arthur, D., Vassilvitskii, S., 2006. k-means++: The Advantages of Careful Seeding. Technical Report 2006-13. Stanford InfoLab.

[6] Askari, Q., Saeed, M., Younas, I., 2020. Heap-based optimizer inspired by corporate rank hierarchy for global optimization. Expert Systems with Applications 161, 113702, doi:10.1016/j.eswa.2020.113702.

[7] Attia, A.F., Sehiemy, R.A.E., Hasanien, H.M., 2018. Optimal power flow solution in power systems using a novel sine-cosine algorithm. International Journal of Electrical Power & Energy Systems 99, 331–343, doi:10.1016/j.ijepes.2018.01.024.

[8] Bejarbaneh, E.Y., Bagheri, A., Bejarbaneh, B.Y., Buyamin, S., Chegini, S.N., 2019. A new adjusting technique for PID type fuzzy logic controller using PSOSCALF optimization algorithm. Applied Soft Computing 85, 105822, doi:10.1016/j.asoc.2019.105822.

[9] Bian, Q., Nener, B., Wang, X., 2018. Control parameter tuning for aircraft crosswind landing via multi-solution particle swarm optimization. Engineering Optimization 50, 1914–1925, doi:10.1080/0305215x.2018.1435646.

[10] Biertümpfel, F., Pfifer, H., 2018. Worst Case Gain Computation of Linear Time-Varying Systems over a Finite Horizon, in: 2018 IEEE Conference on Control Technology and Applications (CCTA), pp. 952–957, doi:10.1109/CCTA.2018.8511591.

[11] Blight, J.D., Lane Dailey, R., Gangsaas, D., 1994. Practical control law design for aircraft using multivariable techniques. International Journal of Control 59, 93–137, doi:10.1080/00207179408923071.

[12] Boothroyd, G., 1984. Use of robots in assembly automation. CIRP Annals 33, 475–484, doi:10.1016/s0007-8506(16)30169-x.

[13] Boughari, Y., , Ghazi, G., Boyez, R.M., Theel, F., 2017. New methodology for optimal flight control using differential evolution algorithms applied on the cessna citation x business aircraft – part 2. validation on aircraft research flight level d simulator. INCAS BULLETIN 9, 45–59, doi:10.13111/2066-8201.2017.9.2.4.

[14] Boughari, Y., Botez, R.M., Ghazi, G., Theel, F., 2016. Flight control clearance of the cessna citation x using evolutionary algorithms. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 231, 510–532, doi:10.1177/0954410016640821.

[15] Bureerat, S., Pholdee, N., 2016. Optimal truss sizing using an adaptive differential evolution algorithm. Journal of Computing in Civil Engineering 30, 04015019, doi:10.1061/(asce)cp.1943-5487.0000487.

[16] Chen, H., Jiao, S., Heidari, A.A., Wang, M., Chen, X., Zhao, X., 2019. An opposition-based sine cosine approach with local search for parameter estimation of photovoltaic models. Energy Conversion and Management 195, 927–942, doi:10.1016/j.enconman.2019.05.057.

[17] Chen, S.H., Ho, W.H., Chou, J.H., Zheng, L.A., 2011. Design of robust-stable and quadratic finite-horizon optimal active vibration controllers with low trajectory sensitivity for uncertain flexible mechanical systems using an integrative computational method. Applied Soft Computing 11, 4830–4838, doi:10.1016/j.asoc.2011.06.018.

[18] Deb, K., 1999. An introduction to genetic algorithms. Sadhana 24, 293–315, doi:10.1007/BF02823145.

[19] Dou, R., Duan, H., 2017. Lévy flight based pigeon-inspired optimization for control parameters optimization in automatic carrier landing system. Aerospace Science and Technology 61, 11–20, doi:10.1016/j.ast.2016.11.012.

[20] Draa, A., Bouzoubia, S., Boukhalfa, I., 2015. A sinusoidal differential evolution algorithm for numerical optimisation. Applied Soft Computing 27, 99–126, doi:10.1016/j.asoc.2014.11.003.

[21] Dukeman, G., Hill, A., 2008. Rapid trajectory optimization for the ARES i launch vehicle, in: AIAA Guidance, Navigation and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics, doi:10.2514/6.2008-6288.

[22] Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory, in: MHS 95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, doi:10.1109/mhs.1995.494215.

[23] ElSaid, A., El Jamiy, F., Higgins, J., Wild, B., Desell, T., 2018. Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration. Applied Soft Computing 73, 969–991, doi:10.1016/j.asoc.2018.09.013.

[24] kai Feng, Z., Liu, S., jing Niu, W., jian Li, B., chuan Wang, W., Luo, B., min Miao, S., 2020. A modified sine cosine algorithm for accurate global optimization of numerical functions and multiple hydropower reservoirs operation. Knowledge-Based Systems 208, 106461, doi:10.1016/j.knosys.2020.106461.

[25] Fernández, A., Peña, A., Valenzuela, M., Pinto, H., 2018. A binary percentile sin-cosine optimisation algorithm applied to the set covering problem, in: Advances in Intelligent Systems and Computing. Springer International Publishing, pp. 285–295, doi:10.1007/978-3-030-00211-4_25.

[26] Gabis, A.B., Meraihi, Y., Mirjalili, S., Ramdane-Cherif, A., 2021. A comprehensive survey of sine cosine algorithm: variants and applications. Artificial Intelligence Review , doi:10.1007/s10462-021-10026-y.

[27] Green, M., Limebeer, D.J.N., 1995. Linear robust control. Prentice-Hall, Inc.

[28] Guo, S.M., Tsai, J.S.H., Yang, C.C., Hsu, P.H., 2015. A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set, IEEE. pp. 1003–1010.

[29] Gupta, S., Deep, K., 2019a. A hybrid self-adaptive sine cosine algorithm with opposition based learning. Expert Systems with Applications 119, 210–230, doi:10.1016/j.eswa.2018.10.050.

[30] Gupta, S., Deep, K., 2019b. Hybrid sine cosine artificial bee colony algorithm for global optimization and image segmentation. Neural Computing and Applications 32, 9521–9543, doi:10.1007/s00521-019-04465-6.

[31] Gupta, S., Deep, K., 2019c. Improved sine cosine algorithm with crossover scheme for global optimization. Knowledge-Based Systems 165, 374–406, doi:10.1016/j.knosys.2018.12.008.

[32] Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems 97, 849–872, doi:10.1016/j.future.2019.02.028.

[33] Hessert, R., Satzger, W., Haase, A., Schafmeister, A., 2006. A new type of x-ray diffractometer with cooperating robots for residual stress analysis on large components. Materials Science Forum 524-525, 749–754, doi:10.4028/www.scientific.net/msf.524-525.749.

[34] Kaveh, A., 2017. Charged System Search Algorithm, in: Kaveh, A. (Ed.), Advances in Metaheuristic Algorithms for Optimal Design of Structures. Springer International Publishing, Cham, pp. 45–89.

[35] Kim, J.H., Whang, I.H., Kim, B.M., 2016. Finite Horizon Integrated Guidance and Control for Terminal Homing in Vertical Plane. Journal of Guidance, Control, and Dynamics 39, 1104–1112, doi:10.2514/1.G001699.

[36] Klimchik, A., Ambiehl, A., Garnier, S., Furet, B., Pashkevich, A., 2017. Efficiency evaluation of robots in machining applications using industrial performance measure. Robotics and Computer-Integrated Manufacturing 48, 12–29, doi:10.1016/j.rcim.2016.12.005.

[37] Kumar, S., Parhi, D.R., Muni, M.K., Pandey, K.K., 2020. Optimal path search and control of mobile robot using hybridized sine-cosine algorithm and ant colony optimization technique. Industrial Robot: the international journal of robotics research and application 47, 535–545, doi:10.1108/ir-12-2019-0248.

[38] Li, S., Chen, H., Wang, M., Heidari, A.A., Mirjalili, S., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems 111, 300–323, doi:10.1016/j.future.2020.03.055.

[39] MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations, in: In 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

[40] Mahmoodabadi, M.J., Rezaee Babak, N., 2020. 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, doi:10.1016/j.ast.2019.105598.

[41] McKay, M.D., Beckman, R.J., Conover, W.J., 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21, 239, doi:10.2307/1268522.

[42] Megretski, A., Rantzer, A., 1997. System analysis via integral quadratic constraints. IEEE Transactions on Automatic Control 42, 819–830, doi:10.1109/9.587335.

[43] Mirjalili, S., 2015a. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems 89, 228–249, doi:10.1016/j.knosys.2015.07.006.

[44] Mirjalili, S., 2015b. The Ant Lion Optimizer. Advances in Engineering Software 83, 80–98, doi:10.1016/j.advengsoft.2015.01.010.

[45] Mirjalili, S., 2016a. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications 27, 1053–1073, doi:10.1007/s00521-015-1920-1.

[46] Mirjalili, S., 2016b. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems 96, 120–133, doi:`10.1016/j.knosys.2015.12.022`.

[47] Mirjalili, S., Lewis, A., 2016. The Whale Optimization Algorithm. Advances in Engineering Software 95, 51–67, doi:`10.1016/j.advengsoft.2016.01.008`.

[48] Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey Wolf Optimizer. Advances in Engineering Software 69, 46–61, doi:`10.1016/j.advengsoft.2013.12.007`.

[49] Mohammad, A.E.K., Hong, J., Wang, D., 2018. Design of a force-controlled end-effector with low-inertia effect for robotic polishing using macro-mini robot approach. Robotics and Computer-Integrated Manufacturing 49, 54–65, doi:`10.1016/j.rcim.2017.05.011`.

[50] Mohanty, C.S., Khuntia, P.S., Mitra, D., 2017. Design of stable nonlinear pitch control system for a jet aircraft by using artificial intelligence. Proceedings of the National Academy of Sciences, India Section A: Physical Sciences 89, 57–66, doi:`10.1007/s40010-017-0396-z`.

[51] Morgan, D., Chung, S.J., Hadaegh, F.Y., 2015. Swarm Assignment and Trajectory Optimization Using Variable-Swarm, Distributed Auction Assignment and Model Predictive Control, in: AIAA Guidance, Navigation, and Control Conference. American Institute of Aeronautics and Astronautics. number 0 in AIAA SciTech Forum.

[52] Murray, R.M., 1994. A Mathematical Introduction to Robotic Manipulation. Taylor and Francis Inc.

[53] Pfifer, H., Seiler, P., 2016. Less conservative robustness analysis of linear parameter varying systems using integral quadratic constraints. International Journal of Robust and Nonlinear Control 26, 3580–3594, doi:`10.1002/rnc.3521`.

[54] Pholdee, N., Bureerat, S., 2014. Hybrid real-code ant colony optimisation for constrained mechanical design. International Journal of Systems Science 47, 474–491, doi:`10.1080/00207721.2014.891664`.

[55] Pholdee, N., Bureerat, S., Yıldız, A.R., 2017. Hybrid real-code population-based incremental learning and differential evolution for many-objective optimisation of an automotive floor-frame. International Journal of Vehicle Design 73, 20, doi:10.1504/ijvd.2017.082578.

[56] Qu, C., Gai, W., Zhong, M., Zhang, J., 2020. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. Applied Soft Computing 89, 106099, doi:10.1016/j.asoc.2020.106099.

[57] Rajput, V.S., Jatoth, R.K., Dhanuka, P., Bhasker, B., 2016. Hybrid DE-PSO based pitch controller design for aircraft control system, in: 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), IEEE, doi:10.1109/cimca.2016.8053258.

[58] Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Information sciences 179, 2232–2248.

[59] Ray, P.K., Mohanty, A., 2019. A robust firefly–swarm hybrid optimization for frequency control in wind/PV/FC based microgrid. Applied Soft Computing 85, 105823, doi:10.1016/j.asoc.2019.105823.

[60] Sallam, K.M., Sarker, R.A., Essam, D.L., Elsayed, S.M., 2015. Neurodynamic differential evolution algorithm and solving CEC2015 competition problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1033–1040, doi:10.1109/CEC.2015.7257003.

[61] Seiler, P., 2015. Stability Analysis With Dissipation Inequalities and Integral Quadratic Constraints. IEEE Transactions on Automatic Control 60, 1704–1709, doi:10.1109/TAC.2014.2361004.

[62] Seiler, P., Moore, R.M., Meissen, C., Arcak, M., Packard, A., 2019. Finite horizon robustness analysis of LTV systems using integral quadratic constraints. Automatica 100, 135–143, doi:10.1016/j.automatica.2018.11.009.

[63] Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11, 341–359.

[64] Tanabe, R., Fukunaga, A., 2013. Evaluating the performance of SHADE on CEC 2013 benchmark problems, IEEE. pp. 1952–1959.

[65] Tanabe, R., Fukunaga, A.S., 2014. Improving the search performance of SHADE using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658–1665, doi:10.1109/CEC.2014.6900380.

[66] Tewari, A., 2011. Automatic Control of Atmospheric and Space Flight Vehicles: Design and Analysis with MATLAB® and Simulink®. Springer Science & Business Media.

[67] Veenman, J., Scherer, C.W., Köroğlu, H., 2016. Robust stability and performance analysis based on integral quadratic constraints. European Journal of Control 31, 1–32, doi:10.1016/j.ejcon.2016.04.004.

[68] Wang, L., Xiong, Y., Li, S., Zeng, Y.R., 2019. New fruit fly optimization algorithm with joint search strategies for function optimization problems. Knowledge-Based Systems 176, 77–96, doi:10.1016/j.knosys.2019.03.028.

[69] Wiesel, W.E., 2010. Spaceflight dynamics. 3rd ed., Beavercreek, Ohio , Aphelion Press.

[70] Yang, M., Li, Y., Du, H., Li, C., He, Z., 2019. Hierarchical Multiobjective H-Infinity Robust Control Design for Wireless Power Transfer System Using Genetic Algorithm. IEEE Transactions on Control Systems Technology 27, 1753–1761, doi:10.1109/TCST.2018.2814589.

[71] Yildiz, A.R., Abderazek, H., Mirjalili, S., 2019. A comparative study of recent non-traditional methods for mechanical design optimization. Archives of Computational Methods in Engineering , doi:10.1007/s11831-019-09343-x.

[72] Yıldız, A.R., Yıldız, B.S., Sait, S.M., Li, X., 2019. The harris hawks, grasshopper and multi-verse optimization algorithms for the selection of optimal machining parameters in manufacturing operations. Materials Testing 61, 725–733, doi:10.3139/120.111377.

[73] Yıldız, B.S., Yıldız, A.R., 2018. Comparison of grey wolf, whale, water cycle, ant lion and sine-cosine algorithms for the optimization of a vehicle

engine connecting rod. Materials Testing 60, 311–315, doi:10.3139/120.111153.

[74] Zhang, J., Sanderson, A.C., 2009. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation 13, 945–958.

[75] Zhang, X., Lu, X., Jia, S., Li, X., 2018. A novel phase angle-encoded fruit fly optimization algorithm with mutation adaptation mechanism applied to UAV path planning. Applied Soft Computing 70, 371–388, doi:10.1016/j.asoc.2018.05.030.