DOI: 10.13009/EUCASS2022-7355
9TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)

Bayesian Optimization with Temporal Logic Constraints for Automated GNC Tuning

R. Polonio[⋆]*, C. Ardura*[⋆]*, V. Preda*[⋆⋆] *and L. Hewing*[⋆†]
[⋆]*SENER Aerospace*
Tres Cantos, Madrid, Spain
[⋆⋆]*European Space Agency*
ESTEC, Noordwijk, Netherlands
[rafael.polonio | carlos.ardura | lukas.hewing]@aeroespacial.sener · valentin.preda@esa.int
[†]Corresponding author

Abstract

Virtually every controller and component involves a number of parameters whose correct tuning can be critical to ensure satisfactory performance. Because the manual tuning of controllers is tedious and time-consuming, resulting control and GNC systems often operate at reduced performance. In this paper, we propose the automatic tuning of GNC parameters based on Bayesian Optimization (BO) constrained with Temporal Logic (TL) constraints. We validate the methodology proposed against two study cases: the tuning of a classical control problem in form of an inverted pendulum, and the retuning of the GNC subsystem for an autonomous parafoil system in the final approach phase just prior to pinpoint landing. Our results show how temporal logic constrained BO can be efficiently used to improve system performance, explore parameter interdependencies and provide valuable insights to support the tuning of GNC systems to fulfill complex requirements.

1. Introduction

Automated tuning of control systems (auto-tuning) has been a long-standing problem in control and continues to be an active research field. Classical automatic tuning methods typically target simple SISO structures such as PID control [1]. For complex control systems the relationship between parameterization and achieved performance (potentially with respect to high-level goals or requirements) often becomes prohibitively hard to express concisely and techniques such as global derivative-free optimization algorithms have been considered for their automatic optimization e.g. particle swarm [2] in [3], ant colony [4] in [5] and genetic algorithms [6] in [7]. Even though they have obtained promising results, they often require a vast number of simulations or experiments in order to find good optimizers. In these settings, data efficiency quickly becomes a dominating issue, in particular if experiments on real hardware are involved.

This motivates the use of data-efficient techniques such as Bayesian Optimization (BO) which aims to reduce the number of simulations required to obtain the optimum value. The explicit consideration of noisy function evaluation, as well as extension for constrained optimization, further add to the attractiveness of BO for controller tuning. In the context of controller learning, BO can be understood as aiming at optimizing a closed-loop performance metric cost $J(x)$ as a mapping of the controller parameters x which is evaluated over a sequence of finite-time experiments and has gathered significant attention in recent years [8] [9] [10] [11].

Given the flexibility of this framework, it is then attractive to use the approach to tune with respect to high-level requirements, which we express via temporal logic expressions. Temporal logic provides a framework combining familiar (Boolean) logical operations with temporal modalities to describe a rich set of specifications, based e.g. on the output signals of a system evolving over time. This allows for the precise expressions of requirements and subsequent formal manipulation, enabling the development of automated and systematic design and verification methods. We employ quantitative semantics of such expression to formulate the requirements as constraints in our black-box optimization [12], arriving at an algorithm that efficiently explores the parameter regions to maximize the performance while satisfying temporal logic requirements in the form of constraints.

The ideas described above have been translated into a toolbox called *GNCTuner* that has been developed by SENER. The *GNCTuner toolbox* is a global optimization toolbox based on BO where it is possible to express constraints in the form of TL expressions. This toolbox has been specifically designed for the tuning of GNC parameters since *a)*

DOI: 10.13009/EUCASS2022-7355

given the Gaussian nature of the BO algorithm, it performs correctly in stochastic environments, and *b)* TL allows to capture and manage efficiently the requirements of the mission.

This paper is organized as follows. Section 1 provides an introduction to BO and TL. Section 2 then presents the *GNCTuner toolbox* description while Section 3 contains the simulation studies carried out to test the performance of the toolbox. Finally, in Section 4 we draw some conclusions based on the results obtained with the *GNCTuner toolbox*.

1.1 Bayesian Optimization

Bayesian Optimization (BO) describes a family of ML-based optimization techniques minimizing an unknown objective function having access only to noisy observations thereof [13]. In this contribution, we consider the case of *constrained* and *contextual* BO which aims to solve problems of the form

$$\begin{aligned} \underset{x \in \mathbb{D} \subset \mathbb{R}^d}{\text{minimize}} \quad & \mathbb{E}(f(x,p,w)) \\ \text{subject to} \quad & \mathbb{E}(c_j(x,p,w)) \leq 0\,,\, j = 0,\ldots,n_c\,, \end{aligned} \tag{1}$$

where the *optimization variable* x typically has reasonably low dimension d and the *context variable* $p \in \mathbb{R}^{d_p}$ parameterizes the problem. The only access to objective and constraint functions is through noisy evaluations $y = f(x,p,w)$, $y_c = c_j(x,p,w)$, respectively, in which w describes the random noise and is typically independently distributed between evaluations. The goal of the optimization is then to find $x^*(p)$ minimizing the objective function while satisfying constraints in expectation in an effective manner, i.e. reducing the number of function evaluations.

The main components that constitute the BO framework to achieve that goal are

(a) a *surrogate model* of the objective and constraint functions, typically presented by Gaussian process (GP)[1]; and

(b) an *acquisition function* $\alpha_i : \mathbb{R}^{d+d_p} \to \mathbb{R}$, where i is the current optimization step, on the basis of which the BO algorithm selects a new probing location to effectively navigate the *exploration-exploitation tradeoff*.

Surrogate Model GP regression is a kernel-based nonlinear stochastic regression technique that provides the interpretation of a distribution over functions and is used as the standard surrogate model in BO. A Gaussian process

$$f(x) \sim \mathcal{GP}(\mu(x), k(x,x'))$$

is completely specified by its prior mean and covariance function

$$\begin{aligned} \mu(x) &= \mathbb{E}(f(x)) \\ k(x,x') &= \operatorname{cov}(f(x), f(x')) \end{aligned}$$

based on which the *posterior* distribution of the function f (after collection of data $\{x_i, y_i\}$) can be computed. The stochastic Bayesian interpretation – and in particular the fact that function evaluations of f follow a Gaussian distribution – is a distinguishing feature of Bayesian optimization setting it apart from competing optimization schemes.

In GP regression, it is generally assumed that the observations are function evaluations perturbed by additive Gaussian noise with fixed intensity $y_i = f(x_i) + w_i$, where $w_i \sim \mathcal{N}(0, \sigma_n^2)$. In this case, the inference can be carried out in closed form by Gaussian conditioning, while (hyperparameter) training, e.g. the tuning of parameters affecting the kernel function k, is often done through marginal likelihood maximization, see e.g. [15] for a comprehensive overview. Practically, GP regression is often very effective also if these assumptions are violated – as is the case in scenarios we consider in this paper. Extensions have been developed, dealing e.g. with spatially varying noise characteristics [16], mixture and multi-modal models [17, 18, 19], or heavy-tailed distributions, aiding robustness to outliers [20], which we do, however, not consider here.

Acquisition Function The acquisition function α_i acts as a surrogate cost function that is optimized to enable efficient exploration and thereby define the following probing location

$$x_{i+1} = \underset{x \in \mathbb{D}}{\arg\min} \quad \alpha_i(x)\,, \tag{2}$$

where i is the current optimization step and x_{i+1} the new location at which objective and constraint functions are to be evaluated. The acquisition function α_i is then much easier to optimize than the original f since it is based on the

[1] Note that alternatives exist and the algorithm is generally adaptable to any Bayesian surrogate model [14]

DOI: 10.13009/EUCASS2022-7355

GP approximation and does not require expensive (simulation) experiments associated with the evaluation of $f(x)$. A multitude of choices of different acquisition functions exists. In this contribution, we focus on variants of the *expected improvement* (EI) acquisition function [21]. In particular, we consider the so-called *plug-in* expected improvement [22] which can be expressed as

$$\alpha_i^{\mathrm{PEI}}(x) = (\mu_i(x) - \mu_i^*)\Phi\left(\frac{\mu_i^* - \mu_i(x))}{\sigma_i(x)}\right) - \sigma_i(x)\varphi\left(\frac{\mu_i^* - \mu_i(x))}{\sigma_i(x)}\right), \tag{3}$$

where φ and Φ are the density and cumulative distribution function of the standard normal distribution and $\mu_i(x)$, $\sigma_i(x)$ the mean and standard deviation at x of the current GP surrogate model. The current estimated optimal value, represented by μ_i^* is then selected as the best posterior mean of the objective function among previous evaluation points

$$\mu_i^* = \min_{1\le k\le i}(\mu_i(x_k) - \beta\sigma_i(x_k)) \tag{4}$$

where $\beta \in \mathbb{R}$ is a parameter that defines the quantile used to establish the current minimum. Throughout this paper, we use $\beta = 2$.

Constrained Bayesian Optimization In constrained BO, the (unknown) feasible region of the problem is defined by

$$\left\{x \,\middle|\, c_j(x) \le 0,\ j = 1, \ldots, n_c\right\}$$

in which n_c is the number of constraints present. We consider an approach following largely [12] and [23] which assumes access to (noisy) real-valued evaluations of the constraint functions c_j, allowing for the training of surrogate constraint models $c_j \sim \mathcal{GP}$ directly analogous to the objective function f. The EI acquisition function can then be adjusted to the constrained case by weighing the expected improvement with the probability of constrained satisfaction

$$\alpha_i^{\mathrm{wPEI}} = \alpha_i^{\mathrm{PEI}}(x)\prod_{j=1}^{n_c} p_{j,i}(x), \tag{5}$$

where $p_{j,i}(x)$ is the probability of constraint satisfaction under the surrogate model $c_{j,i}$. Due to the fact that under the GP surrogate model $c_{j,i}(x)$ follows a Gaussian distribution, this probability can be expressed in a straightforward way as $p_{j,i}(x) = \Phi\left(\frac{-\mu_{j,i}(x)}{\sigma_{j,i}(x)}\right)$ where $\mu_{j,i}(x)$, $\sigma_{j,i}(x)$ are posterior mean and standard deviation of the surrogate model of constraint j and optimization step i.

Contextual Bayesian Optimization *Contextual* BO addresses the optimization under available *side-information* (or *context* information) which may change the minimum in the considered optimization problem. This results in a parametric optimization problem where the parameter $p \in \mathbb{R}^{d_p}$ is observed and may vary during the optimization. At each optimization step i the BO then aims to find the parameterized optimizer $x_i^*(p_i)$. It is assumed that the context parameter p influences the location of the minimum in a similar fashion to x, which motivates the natural solution of extending the GP surrogate models by the considered context [24], i.e. for the objective function

$$f \sim \mathcal{GP}(\mu, k),\ \mu : \mathbb{R}^{d+d_p} \to \mathbb{R},\ k : \mathbb{R}^{d+d_p} \times \mathbb{R}^{d+d_p} \to \mathbb{R},$$

i.e. with $\tilde{x} := [x^\mathsf{T}, p^\mathsf{T}] \in \mathbb{R}^{d+d_p}$. Under these considerations, the previous exposition is readily adapted to the contextual case, in which now part of the variables p are fixed at each step, i.e. cannot be chosen in the optimization, whereas x is optimized over as before.

With these main ingredients, a high-level pseudo-algorithm for BO is stated in Algorithm 1. For simplicity, we formulate this in the unconstrained and non-contextual case. There, the *stopping condition* is typically a fixed number of optimization steps, but can also arbitrary conditions on e.g. the achieved (estimated) precision, and the initial training size n_0 can be equal to 0.

1.2 Temporal Logic

Temporal logic (TL) refers to a number of formal languages that allow describing complex properties (i.e., behaviors) of the (temporal) evolution of a system. As such, it has seen important applications in the formal verification of software systems [25]. More recently, variations are researched for the management and verification of temporal objectives in mechatronic & robotic systems [26], as well as for synthesis of controllers subject to TL specifications, see e.g. [27] [28] and references therein. There exists a vast variety of different TL languages, of which we focus on

DOI: 10.13009/EUCASS2022-7355

Algorithm 1 Bayesian optimization pseudo-algorithm (unconstrained, without context)

Place a Gaussian process prior on f
Observe f at n_0 points from an initial experimental design and set $i = n_0$
while stopping criterion not met **do**
    Update the posterior probability distribution on f using all available data $\mathcal{D} = \{(x_k, y_k)\}_{k=0}^{i}$
    (optional) (Re)-optimize hyperparameters
    $x_{i+1} = \operatorname{argmin} \alpha_i(x)$
    Carry out experiment at and observe $y_{i+1} = f(x_{i+1})$,
    (Re)-compute optimizer x_{i+1}^* as the point with the smallest posterior mean.
    Increment i
end while
Return the solution x_i^*

Signal Temporal Logic (STL) specializing in continuous-valued signals over real-time [29] as is commonly the output of dynamic systems considered in GNC. We give here a simplified introduction and refer to e.g. [30, 31].

Given a *trace* $s_i(t) : \mathbb{R} \to \mathbb{R}^n$ of a dynamic system, comprising for instance the state and input trajectories, STL can be used to formulate *propositions* ϕ whose satisfaction relation is expressed as $(s, t) \vDash \phi$, indicating that the trace s satisfies ϕ starting from position t. This is done starting from *atomic propositions* a whose satisfaction relation is directly defined through a condition $\mu(s(t)) > 0$ which can be combined into complex expressions using the following syntax

$$\phi \ ::= \ \text{true} \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[a,b]} \, \phi_2$$

Here, the negation ("not", $\neg$) and conjunction ("and", $\wedge$) correspond to the typical propositional logic, whereas the until operator ($U_{[a,b]}$) with interval $[a, b]$ introduces temporal characteristics. For convenience, derived operators, such as *disjunction* ("or", $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$), *implication* ('$\phi_1 \Rightarrow \phi_2 := \neg\phi_1 \vee (\phi_1 \wedge \phi_2)$, *eventually* ($\Diamond_{[a,b]}\phi := \text{true} U_{[a,b]}\phi$) or *always* ($\Box_{[a,b]}\phi := \neg\Diamond_{[a,b]}\neg\phi$) are additionally used.

A trace s then satisfies ϕ iff $(s, 0) \vDash \phi$, which is defined inductively as [32]:

$$(s, t) \vDash \text{true} \tag{6a}$$
$$(s, t) \vDash a \qquad \text{iff } \mu(s(t)) > 0 \tag{6b}$$
$$(s, t) \vDash \neg\phi \qquad \text{iff } (s, t) \nvDash \phi \tag{6c}$$
$$(s, t) \vDash \phi_1 \wedge \phi_2 \qquad \text{iff } (s, t) \vDash \phi_1 \text{ and } (s, t) \vDash \phi_2 \tag{6d}$$
$$(s, t) \vDash \phi_1 U_{[a,b]} \phi_2 \qquad \text{iff } \exists t' \in t + [a, b] \text{ s.t. } (s, t') \vDash \phi_2 \text{ and } \forall t'' \in [t, t'] \, (s, t'') \vDash \phi_1 \tag{6e}$$

Spatial Robustness STL and related languages allow the expression of robustness measures as a real-valued quantity that describes the distance (in a suitable sense) of a given signal to the satisfaction threshold of a given proposition, often called the robustness degree [33]. Here, positive values typically represent satisfaction of the constraint, therefore the greater the robustness degree of a signal with respect to an STL formula, the greater the margin of satisfaction. An example of this is the spatial robustness degree, which can be defined recursively as [32]

$$\begin{aligned}
\rho(\text{true}, s, t) &= \infty \\
\rho(\mu, s, t) &= \mu(s(t)) \\
\rho(\neg\phi, s, t) &= -\rho(\phi, s, t) \\
\rho(\phi_1 \vee \phi_2, s, t) &= \max(\rho(\phi_1, s, t), \rho(\phi_2, s, t)) \\
\rho(\phi_1 U_{a,b} \phi_2, s, t) &= \max_{t' \in t + [a,b]} (\min(\rho(\phi_2, s, t'), \min_{t'' \in [t, t')} \rho(\phi_1, s, t''))
\end{aligned}$$

2. Temporal Logic Constrained Bayesian Optimization

The following section presents an overview of the *GncTuner Toolbox* under development at SENER Aeroespacial, aiming to enable GNC autotuning subject to temporal logic constraints.

2.1 Overview

The *GncTuner Toolbox* is developed with the goal of automatically tuning control systems – in particular GNC software – making use of noisy simulations in the style of Monte Carlo simulations. The outcomes of these simulations are

DOI: 10.13009/EUCASS2022-7355

then considered as functions, mapping a chosen parameterization of the control system x that is to be optimized (and potential additional parameters p), as well as stochastic elements w due to the Monte Carlo nature, to a simulation trace s. That is, given a simulation shot indexed by i we have

$$s_i = f_{\text{sim}}(x_i, p_i, w_i)\,,$$

where w_i are i.i.d. between simulation shots and the parameterization x_i can be chosen and is to be optimized.

In order to formulate an optimization problem, these simulation traces are then evaluated with respect to their achieved performance using a suitable scalar cost function $f(s)$, or, similarly, with respect to the satisfaction of constraints using constraint function $c_j(s) \leq 0$ for all $j = 1, \ldots, n_c$ where n_c are the number of constraints. Within the *GncTuner Toolbox*, we put particular emphasis on constraints expressed via temporal logic, see Section 2.3. Finally, in order to express a sound optimization problem, the influence of uncertainty and noise w need to be taken into account, resulting in an optimization of expected cost and constraint satisfaction

$$\begin{aligned} \underset{x \in \mathbb{R}^d}{\text{minimize}} \quad & \mathbb{E}_w\left((f \circ f_{\text{sim}})(x, p, w)\right) \\ \text{subject to} \quad & \mathbb{E}_w\Big((c_j \circ f_{\text{sim}})(x, p, w)\Big) \leq 0\,, \text{ for all } j = 1, \ldots, n_c \end{aligned} \tag{7}$$

An optimization of this problem class poses numerous challenges, for instance:

1. While we can draw samples of w and generate simulation traces to evaluate cost and constraint functions, the expected values are not directly available.
2. Derivative information, even for a single simulation trace, are not available, i.e. we are limited to simple evaluation of the functions via simulation.
3. The simulations are typically computationally expensive.
4. The parameters x we optimize over are usually continuous-valued vectors $x \in \mathbb{R}^d$.
5. The uncertainty and noise w can affect the simulation in complex and ill-behaved ways.

The first three challenges leave us in the domain of (derivative-free) stochastic optimization, with direct links to e.g. reinforcement learning and multi-armed bandits [34]. Depending on the particular problem class, there exist a wealth of solution approaches; for the case of continuous vector-valued optimization variables (challenge 4), BO has been particularly successful and currently forms the backbone of the *GncTuner toolbox*. While the formulation in (7) remains highly general, the careful choice and design of suitable cost and constraint functions f, c_j can be essential to arrive at a practically solvable problem. In particular, whenever possible, care should be taken to avoid e.g. non-continuities and facilitate the regression tasks that underlie surrogate-based optimization techniques such as BO.

2.2 Toolbox structure

The overall toolbox structure and software architecture of the *GncTuner Toolbox* is illustrated in Figure 1. It is implemented in MATLAB following an object-oriented approach with a central `GncTuner` class. This class then contains two central components

1. **`SimulationInterface`:** Manages the simulation environments associated with the considered GNC or control system tuning problem. The simulations are organized via scenario folders following a standardized interface to enable data transfer between simulation and tuning algorithm. Current examples include MATLAB, Simulink, and Python-based simulators, the latter being integrated via the established MATLAB-Python interface.
2. **`Optimizer`:** An optimizer object, implementing the actual optimization algorithm for tuning the GNC software. Currently, this supports the associated *BO Toolbox* developed in this contribution.

In addition to that, open external software packages are used, namely the GPML toolbox[2] as the basis for the Gaussian process regression on the BO Toolbox, as well as TALIRO[3] for the formulation and evaluation of temporal logic constraints.

[2] `www.gaussianprocess.org`
[3] `sites.google.com/a/asu.edu/s-taliro`

DOI: 10.13009/EUCASS2022-7355

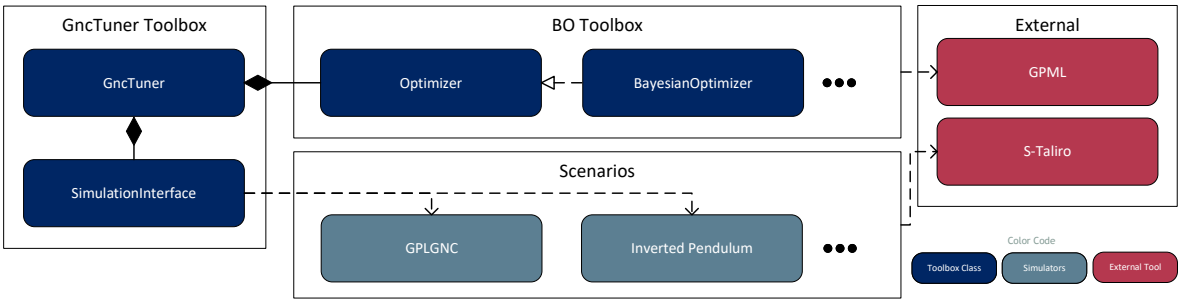


Figure 1: Overview of GncTuner Toolbox software architecture

2.3 Temporal Logic Constraints

In order to express temporal logic constraints, we make use of the open tool TALIRO (TemporAl LogIc RObustness) [35]. While the TALIRO suite encompasses a large collection of powerful tools, we make exclusive use of the sub-tool DP-TALIRO, which allows the evaluation of the *robustness degree* of a simulation trace with respect to a defined temporal logic specification in an effective manner, using a dynamic programming algorithm. For a general discussion of temporal logic and robustness degree, we refer to documents [33] and [29] respectively.

Using this robustness degree allows us to formulate temporal logic constraints of the $c_j(s) \leq 0$, which directly expresses that the simulation trace s satisfies the temporal logic constraint if (the negative of) its robustness degree is lower than 0. Additionally, using the robustness degree allows formulating a 'distance' or metric on how close the trace is to a violation, or satisfaction, of the temporal logic constraint. Under the assumption of a sufficiently smooth map from optimization parameter x to the robustness degree, this can then greatly facilitate the search and establishment of feasible parameter combinations, i.e. the parameterizations x that satisfy the considered TL requirements.

As has been noted in [36], in particular the logical operations in a TL expression can lead to difficult signals for the underlying regression[4]. In our setting, this can be partly alleviated since a) the noise in the simulations is expected to smooth the discontinuities in the individual signals, that is, we aim to learn the expected value of the robustness degree, not the robustness degree itself, and b) `AND` operations on the primary level of the TL expression can be handled by defining separate constraints, evading the problem. Nevertheless, if discontinuous derivatives or nearly discontinuous derivatives are to be expected, a careful choice of an adequate kernel function may be necessary.

3. Simulation Studies

In the following, we present two simulation studies demonstrating the developed tool. The first is a toy example of the swing-up of an inverted pendulum which is provided by the open AI gym[5] (`Pendulum-v1`). As a second example, we investigate the tuning of high-level parameters of a GNC algorithm for autonomous landing under parafoil, which is inspired by Space Rider GNC developed at SENER Aerospace [37].

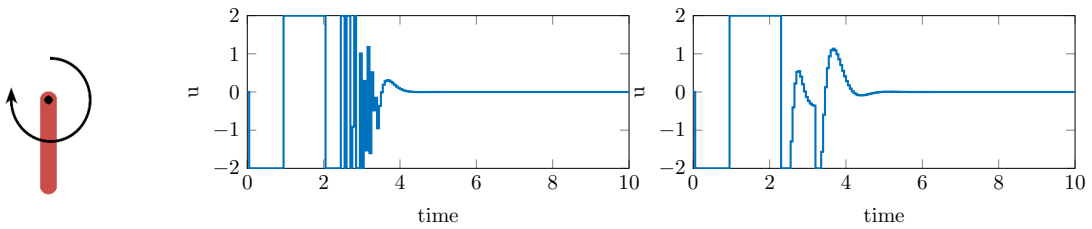
3.1 Inverted Pendulum: Swing-up Optimization

In this example case, we are going to test the proposed methodology for the tuning of a non-linear controller (4-dimensional parameter) while imposing time-dependent requirements via STL. The controller consists of an energy-based swing-up controller, which then switches to PD control around the upper equilibrium of the pendulum. The tuning parameters consist of the PD gains, as well as a gain of the energy controller and the mode switch condition between the two. In addition to that, we impose an STL constraint to combat chattering behavior which occurs under an unconstrained optimization. Figure 2 presents an illustration.

[4]For instance, `AND` and `OR` operations essentially correspond to a 'min' and 'max' operation on the two signals, leading to discontinuous derivatives.

[5]`gym.openai.com`

DOI: 10.13009/EUCASS2022-7355



(a) Simulation Screenshot (b) Input trajectory (unconstrained) (c) Input trajectory (TL-constrained)

Figure 2: Illustration of the inverted pendulum simulation environment (`OpenAI Gym`) and exemplary control inputs after unconstrained and constrained controller tuning.

3.1.1 Problem Description

System dynamics The dynamics of the inverted pendulum are given by

$$I\ddot{\theta} = gl_c \sin(\theta) + u\,, \tag{8}$$

where θ is the pendulum's angle with respect to its upper equilibrium, I is the inertia of the system, l_c is the center of mass, here $l_c = l/2$ where l is the total length of the rod, and u is the applied torque. For this example we will consider that the initial state is uncertain and is distributed as:

$$x(0) = [\theta(0),\, \dot{\theta}(0)]^\mathsf{T} \sim \mathcal{N}\left([\pi,\, 0]^\mathsf{T},\, [\sigma_w^2, 0]^\mathsf{T}\right)\,,$$

where σ_w is the standard deviation of the initial pendulum angle and constitutes the only considered source of noise in this case. We consider here $\sigma_w = 1$. The input to the system is the applied torque at the base of the rod, which is limited to $u \in [-2, 2]$.

Simulation Environment The system is simulated via a Matlab script calling the `OpenAI Gym` Python environment of the inverted pendulum. The system is discretized with an Euler forward method with a sampling time of 0.05 seconds and the total simulation time is 10 seconds. The setup corresponds to the unmodified `OpenAI Gym` continuous pendulum problem (`Pendulum-v1`).

Controller description The controller considered is an energy-based swing-up controller, switching to a linear around the (upper) equilibrium [38]. The energy of the system can be expressed as

$$E(\theta, \dot{\theta}) = E_{\text{kin}}(\dot{\theta}) + E_{\text{pot}}(\theta) = \frac{1}{2} I\dot{\theta}^2 + l_c g m \cos(\theta) \tag{9}$$

where the desired reference energy corresponds to the equilibrium ($\dot{\theta} = 0$, $\theta = 0$) leading to $E_{\text{ref}} = mgl_c$. The main concept behind this controller is, therefore, to reach the upper state as soon as possible by increasing the energy of the system and, when near the equilibrium point, to stabilize the system. As a switching criterion, we consider the 2-norm of the system state, such that the control action can be described as:

$$u(\theta, \dot{\theta}) = \begin{cases} -(K_\text{k}\theta + K_\text{d}\dot{\theta}) & \text{if} \quad \sqrt{\theta^2 + \dot{\theta}^2} < \epsilon \\ -K_\text{su} \cdot (E(\theta, \dot{\theta}) - E_\text{ref}) \cdot \text{sign}(\dot{\theta}) & \text{if} \quad \sqrt{\theta^2 + \dot{\theta}^2} \geq \epsilon \end{cases} \tag{10}$$

Here the considered *tunable parameters* are the proportional and derivative gain K_k and K_d of the stabilizing controller, as well as the proportional gain of the energy-based controller K_su and the threshold to switch between controllers ϵ_θ. With this, our tuneable parameter vector is

$$x = [K_\text{su},\, \epsilon, K_\text{k}, K_\text{d}]^\mathsf{T}\,.$$

Objective function As objective function for this example we consider the "swing-up time", which we establish as the maximum time at which $|\theta| > 0.01$, i.e the maximum time where the pendulum position is not sufficiently close to the upper equilibrium. Therefore, the objective function can be expressed as

$$f(\theta(t)) = \max t \text{ s.t. } |\theta(t)| > 0.01 \tag{11}$$

Given that the step for the discretization considered by the openAI environment is 0.05 seconds, the objective function is mildly discontinuous and varies in steps of 0.05. Additionally, due to the maximum simulation time, it is saturated at 10 seconds.

DOI: 10.13009/EUCASS2022-7355

Constraints In the unconstrained setting, optimal controllers were found to display chattering behavior in particular at the instances of controller switching, but also due to high gains in the energy controller, i.e switching rapidly from maximum to minimum action in a short period of time. Figure 2b shows an example. In order to avoid this behavior, we investigate additional requirements expressed as STL. In particular, we consider a constrained expressed in natural language as:

> *After a change in control of maximum amplitude (e.g. from maximum to minimum control action) no such change of maximum amplitude is allowed for the following 0.2 seconds.*

This is expressed in STL as:

$$\phi = \square(\text{bang} \Rightarrow \square_{[0,0.2]}(\neg\text{bang}))\,, \tag{12}$$

where "bang" is an atomic proposition that is true whenever the control input changes by a magnitude of 4 which constitutes the maximum possible change. For the definition of the constraint function, we make use of the robustness degree of the previously defined STL formula for which we add a heuristically chosen safety margin accounting for the variability due to noise.

$$c(s) = -\rho(\phi, s) + 1.5 \tag{13}$$

3.2 Design Consideration

Domain Considering that we have little apriori knowledge about the true minimizer or the domain where the minimum is likely to be, we will consider a reasonably big domain:

$$\begin{aligned} K_{\text{su}} &\in [0.1, 100], \\ \epsilon &\in [0.1, 2], \\ K_{\text{k}} &\in [0.1, 100], \\ K_{\text{d}} &\in [0.1, 100]. \end{aligned}$$

To facilitate the search, we furthermore use a logarithmic transformation of the decision variables, i.e. we optimize over $log(x)$. This gives account to the assumption that the effects of changes in the controller parameters on the swing-up time behave logarithmically, e.g. we assume that a change in controller gain from 0.5 to 1 has an effect in the same order of magnitude as one from 50 to 100.

Approximation of true cost and constraint value In order to evaluate the performance of our algorithm, we would like to evaluate the *true* expected value of cost and constraint functions given a parameterization x. Unfortunately, these are not analytically available so that we have to approximate them. To investigate the performance of our algorithm we will therefore compute the approximated true cost and the true constraint as an empirical average, i.e. by simulating the same parameterization 30 times and taking the mean

$$\mathbb{E}_w\left((f \circ f_{\text{sim}})(x, w)\right) \approx \frac{i}{30}\sum_{i=1}^{30}(f \circ f_{\text{sim}})(x, w_i)\,, \tag{14a}$$

$$\mathbb{E}_w\left((c \circ f_{\text{sim}})(x, w)\right) \approx \frac{i}{30}\sum_{i=1}^{30}(c \circ f_{\text{sim}})(x, w_i)\,. \tag{14b}$$

Note that in particular this step increases the computational effort for the evaluation of the algorithms in this example.

3.2.1 Results

To evaluate the effectiveness of the proposed scheme we compare three cases, namely

TL-constrained BO Considering the weighted PEI acquisition function as presented in(5) and temporal logic constraint (13).

Unconstrained BO The same setup without the TL constraint, making use of the PEI acquisition function (4)

Random Sampling Using Latin-hypercube sampling to generate probing locations. The current minimizer is then defined by the minimum cost sample which simultaneously satisfies the TL-constraint, i.e. given sample points $\{(x_k, y_k, y_{c,k}\}_{k=1}^{i}$ we define the optimizer as $x_i^* = x_{i^*}$ with $i^* = \text{argmin}_{1\leq k\leq i}\, y_k$ such that $y_{c,k} < 0$.

DOI: 10.13009/EUCASS2022-7355

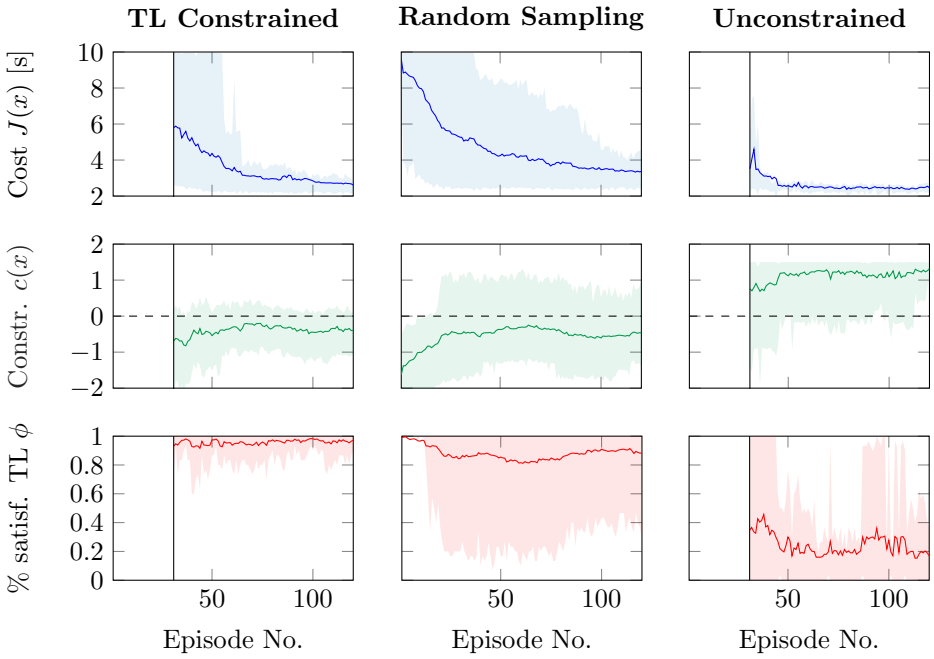


Figure 3: Inverted pendulum optimization results. *Top row:* mean and 90% quantile of the objective function. *Middle row:* mean and 90% quantile of constraint function. *Bottom row:* mean and 90% quantile of the fraction of samples satisfy the TL constraint

For all GPs, we consider a squared-exponential kernel with automatic relevance determination (ARD-SE kernel), meaning that the kernel length-scales are automatically tuned via hyperparameter optimization. We consider 30 initial data points for pre-training the GPs using Latin-hypercube sampling and carry out hyperparameter optimization after every 5 optimization steps by optimizing the log-marginal likelihood through a BFGS algorithm.

To generate the results, we run each considered case 30 times for different noise realization to evaluate the achieved *true* cost and constraint values along (14). Fig 3 shows the results of these experiments, displaying the mean (over the 30 runs) as well as the 90% quantile of the objective function (14a) and constraint function (14b). Additionally, we show the mean and quantile of the fraction of samples in (14b) that satisfy the original temporal logic constraint, i.e. approximating the probability of satisfying the TL constraint with the current optimizer.

We observe that the unconstrained BO rapidly minimizes the necessary time to swing-up the pendulum, although this clearly comes at the expense of constraint satisfaction, that is the solution generally displays chattering behavior, which can be quantified by the high likelihood of violating the temporal logic expression (12). The constraint optimization, on the other hand, converges somewhat slower and to a slightly higher minimum swing-up time, as was to be expected by introducing the additional constraint. It can, however, be observed that throughout the optimization the expected value of the constraint function is kept below 0 and that as a result, the current optimizers provide a high chance of satisfying the original TL constraint. Finally, the proposed random sampling scheme shows both slower convergence with respect to the objective value and decreased reliability with respect to constraint satisfaction.

Overall, the results show that the *GncTuner* can be effectively used to ease the controller design process for dynamical systems as it has allowed expressing a complex high-level and time-dependent requirement in a mathematical and verifiable form. The underlying BO algorithm has shown good success when optimizing the 4-dimensional controller parameters, enhancing the performance of the overall system significantly while satisfying the TL constraints with high probability.

3.3 Autonomous Parafoil Landing: Final Approach Optimization

As a second scenario consider the terminal descent and landing (D&L) phase of an atmospheric re-entry scenario, in particular, a flight under a guided parafoil for pinpoint landing. Such a pinpoint landing scenario is considered, for instance, in the ongoing European Space Agency (ESA) Space Rider mission for which SENER is developing the GNC [37]. In the following we investigate the automatic tuning for a similar GNC algorithm using a related benchmark simulation environment–we will collectively refer to this as a generic parafoil landing GNC (GPLGNC).

The guidance strategy for the vehicle is based on the selection of two waypoints, which define regions where the

DOI: 10.13009/EUCASS2022-7355

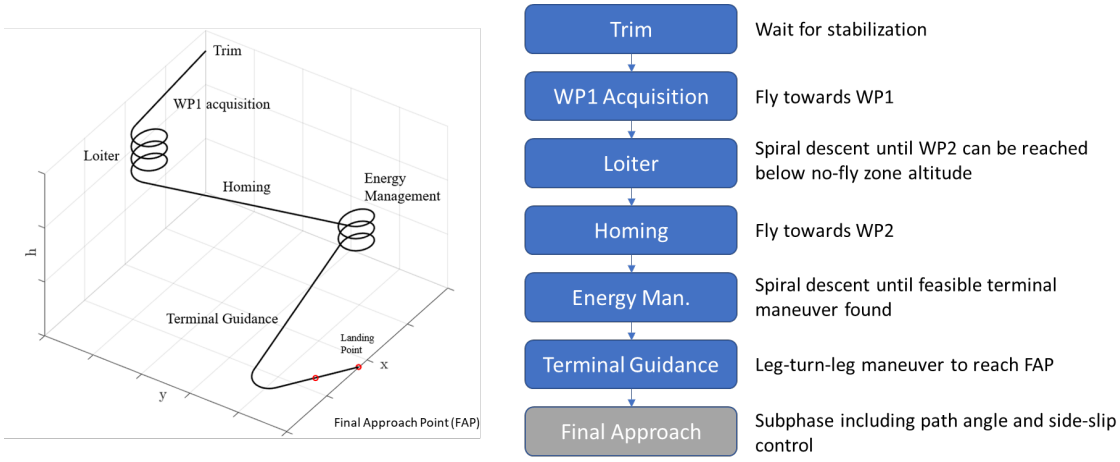


Figure 4: GPLGNC trajectory legs. Credits: [37]

system can safely lose (potential) energy, i.e. height, after which a final maneuver is executed in the *terminal guidance* phase. Figure 4 presents an overview. To account for the dynamic capabilities of the system and the drift induced by wind effects, guidance is done in so-called wind-corrected coordinates, i.e. by subtracting nominal effects of wind that are assumed known due to measurements previous to the landing operation. The waypoints are then computed (offline) on-ground based on the wind conditions, aerodynamic coefficients, and MCI properties of the system and then uploaded to the re-entry vehicle. For further information on the considered guidance strategy, we refer to [37].

For the use of the presented GNC autotuner, we will consider the final phase of the D&L scenario, namely the *final approach*. This corresponds roughly to the final 100 m of the descent, including a "flare" maneuver to temporarily reduce the vertical velocity and ensure a soft touchdown. The objective is, therefore, to improve the vehicle's landing accuracy, while satisfying the velocity constraint at the moment of touchdown. This is to be optimized considering uncertainties corresponding to a full Monte Carlo campaign, including perturbations or aerodynamic properties, mass and inertia as well as uncertainties regarding the wind knowledge and resulting disturbances.

3.3.1 Problem Description

The final phase of the D&L is initiated when passing the so-called *final approach point* (FAP), which then activates a flight path angle control law, optional the lateral velocity correction as well as a final flare maneuver[6] just before landing. There are a number of high-level parameters affecting this phase; the following presents a quick overview of the major ones we consider here for tuning

FAP Height describes the height at which the mode change into the final approach is initiated. Default: 100 m

FAP Angle describes the reference flight path angle (in wind compensated coordinates) defining the final approach. Default: $-\text{atan}(1/3.6) = 0.271$ rad

Flare Distance describes an offset in the ground wind direction from the landing point at which the final approach aims. This is meant to compensate longitudinal effects of the flare. Default: 0 m.

With these three, the final approach point can be fully reconstructed in space, with its horizontal distance in ground wind direction and wind-compensated coordinates given by

$$\text{FAP Angle} + \tan^{-1}(\text{FAP Angle} \cdot \text{FAP Height}) \quad \text{(default value: 360 m)}\,.$$

As we suspect that the *FAP angle*, i.e. the flight path angle reference during the final descent has an influence on the vertical velocity at touchdown, we simultaneously wish to optimize the so-called flare-trigger height and formulate a touchdown velocity constraint for the optimization. Finally, we wish to investigate whether the optimal configuration is influenced by the encountered nominal wind speeds at the landing site. Since this value is known beforehand, but cannot be freely chosen, we include it as a context parameter. With this, we can summarize optimization variables and

[6]The flare is a maneuver consisting in an abrupt stroke of the parafoil to reduce the vertical velocity triggered meters above the landing point.

DOI: 10.13009/EUCASS2022-7355

context parameters as

$$x = \begin{bmatrix} \text{FAP Height} \\ \text{FAP Angle} \\ \text{Flare Distance} \\ \text{Flare Height} \end{bmatrix}, \quad p = \text{wind speed}\,.$$

Simulations In order to simulate the scenario described, an in-house simulation environment in Simulink will be used.

Objective function As the main performance metric for a pinpoint landing we consider the landing accuracy (i.e the distance from the desired landing point to the actual landing point):

$$f(s) = \|p_{\text{LP}} - p(s, t_{\text{landing}})\|\,,$$

where p_{LP} is the position of the desired landing point and $p(s, t_{\text{landing}})$ is the position extracted from simulation trace s at the time instant of landing t_{landing}, which we define as the instant at which the vehicle height drops below the landing point.

Constraint function We consider a requirement to keep the vertical velocity at impact below 3m/s, which we can express as a temporal logic expression in a natural way, e.g

$$\phi = \square\left(a_{\text{ground\_contact}} \Rightarrow a_{\text{perp\_vel}\leq 3}\right)\,. \tag{15}$$

With this, we formulate the constraint function

$$c(s) = -\rho(\phi(s)) + \epsilon\,,$$

in which we choose the margin ϵ as the 3-σ variance of touchdown velocities from previous simulations – effectively aiming to enforce that the expected value of the touchdown velocity remains at a 3-σ distance to the requirement.

3.3.2 Results

We pre-train the GP models with 150 simulations, again chosen via Latin hypercube sampling, and then optimize for another 350 steps for a total of 500 carried out simulations. Note that for the 5-dimensional parameters space we consider this corresponds to about 3.5 data points per dimension. Again, we consider an ARD-SE kernel and optimize hyperparameters every 10 optimization steps. Due to the increased computational complexity of this examples, we show results here of a single optimization as opposed to the analysis we carried out for the pendulum example.

The final result of the optimization of 350 steps is shown in Figure 5 illustrating the posterior GPs of both objective and constraint functions, as well as the final acquisition function values. In terms of the constraints, we can see that the flare trigger altitude has the biggest influence, but also the approach angle has some importance. The influence of the other parameters appears largely mild. The results suggest that steep approaches are not possible as they would violate the touchdown velocity constraint.

With respect to the objective, we observe most clearly that very shallow approach angles lead to a significant deterioration of landing performance, with more nuanced effects for larger angles. The flare height has next to no influence on the landing performance, but the flare distance parameter shows some influence, with typical minima around 10 m. We see that the identified flare height lies around 10 m, with a slight increase for higher wind speeds. In terms of the flare distance, a generally positive value of around 10 m is recommended by the tuning algorithm, which appears to vary with the identified ground wind speed. Overall, the effect of the wind speed as the context parameter appears very mild, suggesting that a GNC tuning irrespective of the nominal wind speeds at the landing points is likely a suitable solution. The height of the FAP is fairly consistently optimized towards the minimum value of 50 m, which may point to a benefit of continuing the periodic update and recalculation of the terminal guidance trajectory towards the very end of the maneuver – in the final approach phase, the heading rate is primarily kept to 0.

Based on these insights, we suggest a slight retuning of the default GNC values, namely by setting the *flare distance* parameters to 10 m (default: 0 m) and the FAP height to 50 m (default: 100 m). To evaluate the performance of this reparameterization, we run 100 MC shots under the same conditions, comparing them to a similar MC campaign of the original tuning. The results are shown in Figures 7 and 6, respectively. From these figures, the following conclusions can be extracted:

DOI: 10.13009/EUCASS2022-7355

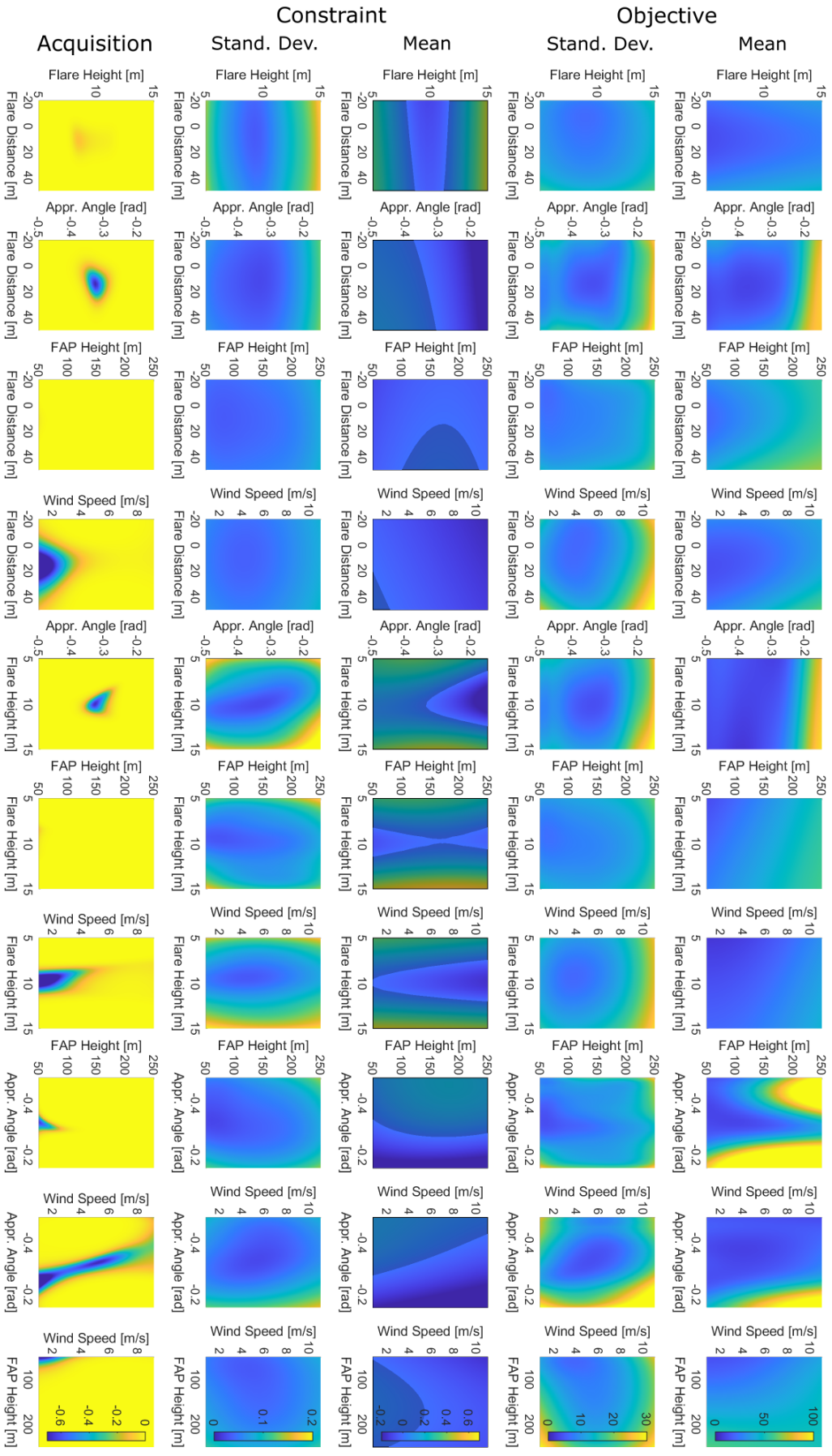
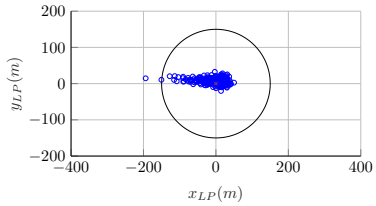


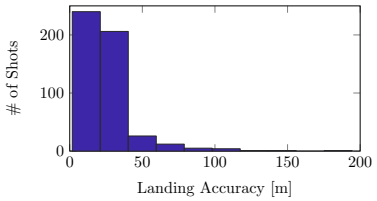
Figure 5: Final state of the final approach optimization showing the mean and variance of both objective and constraint GPs, as well as the acquisition function. For the projection, values of the remaining dimensions are set to the optimizer. Blue indicate low values, yellow high values with the objective to minimize. The shaded regions in the constraint GP illustrate posterior mean values lower zero, corresponding to infeasibility.

DOI: 10.13009/EUCASS2022-7355

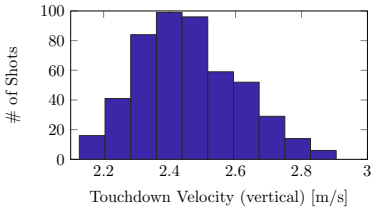
- The landing accuracy has been improved significantly; the mean of the landing accuracy has been reduced from 25.2 m to 15.6 m, and also the non-compliant cases (landing accuracy over 150 meters) have been eliminated. This corresponds to an improvement of 38%.
- Both parameterizations are able to fulfill the vertical velocity requirement with margin. nevertheless, the GNC retuning has reduced the touchdown vertical velocity from a mean of 2.46 m/s to 2.31 m/s, or by 6.1%.



(a) Spatial distribution of landing locations with target radius of 150 m

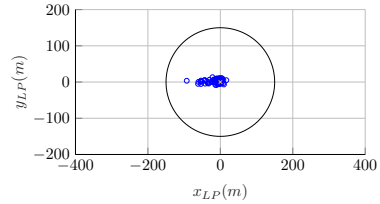


(b) Histogram of landing distances

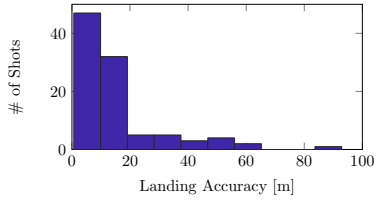


(c) Histogram of vertical touchdown velocities to be kept below 3 m/s

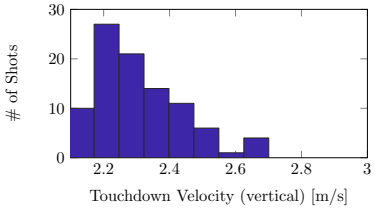
Figure 6: GNG performance before retuning of the final approach phase



(a) Spatial distribution of landing locations with target radius of 150 m



(b) Histogram of landing distances



(c) Histogram of vertical touchdown velocities to be kept below 3 m/s

Figure 7: GNC performance results after BO retuning of the final approach.

4. Conclusions

In this paper, we have described and tested a toolbox for tuning GNC systems in constrained and stochastic environments. On the one hand, the toolbox has been able to capture a complex and time-dependent requirement using STL while also showing its competitiveness against naive optimization by random sampling in the inverted pendulum scenario. In a second example, we have tested *GNCTuner* against a real-world problem related to the tuning of a GNC system. GNCTuner has shown a twofold success: it has been able to reduce the landing accuracy while maintaining a high margin of security in satisfying the vertical velocity constraint while also providing insights into how the parameters affect the performance of the whole GNC system. Additionally, the optimization effectively provides insight on the GNC scenario and parameter interdependencies, also thanks to the contextual optimization.

5. Acknowledgments

This research work was supported by the European Space Agency (ESA) within the Technology Development Element contract 4000133595/0/0/0 - "Artificial Intelligence techniques for GNC design, implementation and verification" (Technical Officer: Valentin Preda). The views expressed in this paper can in no way be taken to reflect the official opinion of ESA

References

[1] K. Åström, T. Hägglund, C. Hang, and W. Ho, "Automatic tuning and adaptation for pid controllers - a survey," *Control Engineering Practice*, vol. 1, no. 4, pp. 699–714, 1993.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE.

DOI: 10.13009/EUCASS2022-7355

[3] M. I. Solihin, L. F. Tack, and M. L. Kean, "Tuning of PID controller using particle swarm optimization (PSO)," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, no. 4, p. 458, 2011.

[4] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, pp. 28–39, nov 2006.

[5] I. Chiha, N. Liouane, and P. Borne, "Tuning PID controller using multiobjective ant colony optimization," *Applied Computational Intelligence and Soft Computing*, vol. 2012, pp. 1–7, 2012.

[6] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, Mass: MIT Press, 1996.

[7] P. Zhang, M. Yuan, and H. Wang, "Self-tuning PID based on adaptive genetic algorithms with the application of activated sludge aeration process," in *2006 6th World Congress on Intelligent Control and Automation*, IEEE, 2006.

[8] M. Neumann-Brosig, A. Marco, D. Schwarzmann, and S. Trimpe, "Data-efficient autotuning with bayesian optimization: An industrial control study," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 730–740, 2020.

[9] C. König, M. Khosravi, M. Maier, R. S. Smith, A. Rupenyan, and J. Lygeros, "Safety-aware cascade controller tuning using constrained bayesian optimization," *arXiv preprint arXiv:2010.15211*, 2020.

[10] D. Kim, H. Oh, and I. Moon, "Black-box modeling for aircraft maneuver control with bayesian optimization.," *Int. J. Control Autom. Syst.*, vol. 17, pp. 1558–1568, 2019.

[11] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020," in *NeurIPS 2020 Competition and Demonstration Track*, pp. 3–26, PMLR, 2021.

[12] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, (Beijing, China), JMLR.org, 2014.

[13] P. I. Frazier, "A tutorial on bayesian optimization," 2018.

[14] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[15] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[16] Q. V. Le, A. J. Smola, and S. Canu, "Heteroscedastic gaussian process regression," in *ICML*, pp. 489–496, 2005.

[17] C. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," *Advances in neural information processing systems*, vol. 14, 2001.

[18] A. Daemi, H. Kodamana, and B. Huang, "Gaussian process modelling with gaussian mixture likelihood," *Journal of Process Control*, vol. 81, pp. 209–220, 2019.

[19] J. Zand and S. Roberts, "Midgap: Mixture density gaussian processes," *NIPS Time Series Workshop*, 2017.

[20] J. Vanhatalo, P. Jylänki, and A. Vehtari, "Gaussian process regression with student-t likelihood," *Advances in neural information processing systems*, vol. 22, 2009.

[21] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[22] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.

[23] B. Letham, B. Karrer, G. Ottoni, E. Bakshy, *et al.*, "Constrained bayesian optimization with noisy experiments," *Bayesian Analysis*, vol. 14, no. 2, pp. 495–519, 2019.

[24] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization.," in *Nips*, pp. 2447–2455, 2011.

[25] C. Baier, *Principles of model checking*. Cambridge, Mass: MIT Press, 2008.

DOI: 10.13009/EUCASS2022-7355

[26] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems," *ACM Computing Surveys*, vol. 52, pp. 1–41, oct 2019.

[27] S. Haesaert, S. Soudjani, and A. Abate, "Temporal logic control of general markov decision processes by approximate policy refinement," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 73–78, 2018.

[28] S. Haesaert, R. Thakker, R. Nilsson, A. Agha-mohammadi, and R. M. Murray, "Temporal logic planning in uncertain environments with probabilistic roadmaps and belief spaces," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, dec 2019.

[29] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, Springer Berlin Heidelberg, 2004.

[30] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems* (K. Chatterjee and T. A. Henzinger, eds.), (Berlin, Heidelberg), pp. 92–106, Springer Berlin Heidelberg, 2010.

[31] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[32] A. Donzé, T. Ferrère, and O. Maler, "Efficient robust monitoring for STL," in *Computer Aided Verification*, pp. 264–279, Springer Berlin Heidelberg, 2013.

[33] G. E. Fainekos, *Robustness of Temporal Logic Specifications*. PhD thesis, University of Pennsylvania, 2008.

[34] W. B. Powell, *Reinforcement learning and stochastic optimization*. John Wiley & Sons, 2021.

[35] G. Fainekos, B. Hoxha, and S. Sankaranarayanan, "Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with s-taliro," in *Runtime Verification* (B. Finkbeiner and L. Mariani, eds.), (Cham), pp. 27–47, Springer International Publishing, 2019.

[36] S. Ghosh, F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor, "Verifying controllers against adversarial examples with bayesian optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7306–7313, IEEE, 2018.

[37] A. Figueroa-GonzÃ¡lez, F. Cacciatore, and R. Haya-Ramos, "Landing guidance strategy of space rider," *Journal of Spacecraft and Rockets*, vol. 0, no. 0, pp. 1–12, 2021.

[38] K. Åström and K. Furuta, "Swinging up a pendulum by energy control," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 1919–1924, 1996. 13th World Congress of IFAC, 1996, San Francisco USA, 30 June - 5 July.