Adversarially Robust Real-Time Optimization and Control with Adaptive Gaussian Process Learning

Akhil Ahmeda, Antonio Del Rio Chanonaa, Mehmet Mercangöza\*

aImperial College London, London, SW7 2BX

\*m.mercangoz@imperial.ac.uk

Abstract

Real-Time Optimization (RTO) and control face two foundational challenges: effectively managing disturbances, noise, and implementation errors at the control layers, and addressing the pervasive issue of plant-model mismatch at the RTO layer. To tackle the first challenge, our recent work introduced the Adversarially Robust Real-Time Optimization and Control (ARRTOC) algorithm which utilises the RTO layer to identify set-points that are inherently robust to control layer perturbations and implementation errors. ARRTOC ensures the chosen set-points are tailored to the underlying controller designs. However, ARRTOC's strength hinges on an accurate steady-state model. To surmount this, in this work, we extend ARRTOC with Adaptive Gaussian Process Learning (AGPL) to make it more applicable in situations where accurate models are not available. AGPL employs Gaussian Process (GP) regression to learn the underlying steady-state model of the objective function and addresses plant-model mismatch through dynamic GP adaptation. The integration of AGPL and ARRTOC in this way tackles the two primary challenges in RTO and control: ARRTOC ensures robustness against control-layer implementation errors, while AGPL addresses plant-model mismatch.

**Keywords**: Adversarially Robust Optimization, Real-Time Optimization, Gaussian Processes, Process Control, Adversarial Machine Learning

* 1. Introduction

Real-Time Optimization (RTO) plays a crucial role in the process operation hierarchy by determining optimal set-points for the lower-level controllers. However, at the control layers, these set-points may be difficult to track because of challenges in implementation due to disturbances and noise. This could be handled by robustifying the controllers to these perturbations however this may add additional complexity to the already resource constrained control layer. Instead, to tackle this challenge, our recent work introduced the Adversarially Robust Real-Time Optimization and Control (ARRTOC) algorithm (Ahmed et al. 2023). Drawing inspiration from adversarial machine learning (Bertsimas et al. 2010), ARRTOC utilises the RTO layer to identify set-points that are inherently robust to implementation errors, such as disturbances, thus alleviating demands on the resource-limited controllers. ARRTOC handles the dual problem of optimality and operability seamlessly as part of an online RTO solution which is described in section 2.1. By design, the chosen set-point is insensitive to potential disturbances and noise. This concept is illustrated in Figure 1, where the performance of a controller around two possible set-points is compared: the global optimum (scenario 1 in blue) and the adversarially robust optimum (scenario 2 in red). We observe that operating at the adversarially robust optimum yields a 30% larger mean objective value compared to operating at the global optimum due to its inherent robustness.

Figure 1: Scenario 1 shows fluctuating real-time objective value (dashed blue line) due to system disturbances, averaging 12.40—40% lower than the expected global optimum of 20.93 (solid blue line with black starred markers). In contrast, scenario 2 displays more stable real-time objective value (dashed red line), averaging 16.09, closer to the expected robust optimum of 17.90 (solid red line with black starred markers).

However, to operate successfully, ARRTOC relies on an accurate steady-state model of the system under consideration (Patrón et al. 2022). Indeed, this model dependency issue and the subsequent challenge of plant-model mismatch is common to all existing RTO formulations (Cortinovis et al. 2016). To address this, in this work, we extend ARRTOC with Adaptive Gaussian Process Learning (AGPL) (Ahmed et al. 2022). AGPL employs Gaussian Process (GP) regression to learn the underlying steady-state model of the objective function and overcomes plant-model mismatch through dynamic GP adaptation. Additionally, GPs, being a non-parametric modeling approach, possess a unique advantage – the ability to address structural mismatch in models, which is typically beyond the capabilities of current state-of-the-art approaches like model-parameter adaptation or modifier adaptation.

We illustrate the algorithm's capabilities through a case study of a multi-loop evaporator process. In this study, ARRTOC customises controller set-points for each loop based on anticipated disturbances, and AGPL minimizes plant-model disparities, ensuring an accurate solution is obtained. This extended framework showcases increased adaptability and robustness, effectively handling uncertainties at both RTO and control layers.

* 1. Background and Methodology
		1. Adversarially Robust Real-Time Optimization and Control (ARRTOC)

In this section we provide a summary of the ARRTOC algorithm. For the interested reader, the full details can be found in Ahmed et al. (2023). The nominal RTO problem is to find a set point, which optimizes some operating goal objective function, such as profit, , while satisfying operating constraints, :

|  |  |
| --- | --- |
|  | (1) |

For reasons outlined in section 1, we wish to use the RTO layer to identify set-points inherently robust to implementation errors. We denote implementation errors as  and we assume they reside within the n-ellipsoid uncertainty set defined as where represents the largest possible perturbation or implementation error we must safeguard against for the state. We seek an adversarially robust set-point i.e a set-point robust to the worst possible implementation error of our process:

|  |  |
| --- | --- |
|  | (2) |

The problem formulation implies that a robust set-point should (i) minimize the worst-case cost as defined as and (ii) simultaneously ensure no constraints are violated for any . Eq. (2) is solved via three mains steps: (i) Neighbourhood cost exploration, (ii) Neighbourhood constraint exploration and finally either (iiia) Robust local move if infeasible under perturbations or (iiib) Robust local move if feasible under perturbations. For brevity, a detailed explanation of these steps is not provided here, but Figure 2 offers an intuitive overview.

Figure 2: Left: No constraint violations in the neighborhood, algorithm aims to minimize worst-case cost. Right: Iterate infeasible due to intersecting infeasible region; algorithm seeks descent direction to restore feasibility.

Figure 2 depicts the two main scenarios during the algorithm’s execution. The left subplot shows an iterate, where during step 2 of the algorithm, no constraint violating neighbours were found in the neighbourhood of the point (depicted as a grey circle based on the uncertainty set, ) i.e. it is feasible under perturbations. In this scenario, cost considerations take precedence over constraints. The objective is to identify a robust local move that reduces the worst-case cost. This involves finding a direction, , and step-size, , which moves away from the high-cost directions, depicted as black arrows. These high-cost directions were found during step 1 of the algorithm by solving the inner maximization problem, , via a multi-start gradient ascent algorithm. The direction and step-size were found by solving a Second Order Cone Program (SOCP). Alternatively, the right subplot shows an iterate, where during step 2 of the algorithm, constraint violating neighbours were found, rendering it infeasible under perturbations. In this scenario, constraints take precedence over cost, with the goal being to find a robust local move which guides the iterate back into the feasible region. Again, we must find a direction,, and step-size, , which moves away from the constraint-violating neighbour directions, depicted as black arrows. These directions were found by solving the constraint maximization problem, . The algorithm iteratively follows the above steps until it reaches a robust local minimum, where (i) the point is surrounded by high-cost neighbours on all sides, meaning there is no direction to reduce the worst-case cost, and (ii) the neighbourhood of the point does not intersect with any of the constraint-violating regions.

* + 1. Adaptive Gaussian Process Learning (AGPL)

As specified in the RTO problem formulation in Eq. (1), an accurate model of the objective function and constraints is essential for achieving an accurate solution and, consequently, effective set-points for the controllers. Without an up-to-date and accurate model, there is a risk that the system can be operated in a suboptimal manner due to plant-model mismatch. Over time, this mismatch can worsen as the system naturally degrades. This is a known challenge in the RTO literature and several adaptation strategies have been proposed to this end (Chachuat et al. 2009, Mendoza et al. 2015). Effective state-of-the-art approaches include: (i) Model-parameter adaptation whereby the model parameters are estimated and updated based on output measurements from the system. However, it assumes the model is structurally correct with only parametric mismatches, which is rarely the case in practice. (ii) Modifier adaptation where the objective function and constraints are modified before optimization to ensure the model and plant share the same optimality conditions. While not reliant on structural assumptions regarding the model, it is highly sensitive to noisy measurements, as gradients must be estimated from data. This makes the approach less robust when dealing with significant levels of measurement noise (Mendoza et al. 2015). Alternatively, due to the rise of machine learning, non-parametric approaches, particularly Gaussian Processes (GPs), have gained attention. GPs do not assume a fixed structural form and are well-suited to handling noisy measurements. They also excel in low data scenarios, making them valuable for modeling engineering systems. Indeed, their ability to address plant-model mismatch in RTO has been recently demonstrated in a process referred to as Adaptive Gaussian Process Learning (AGPL) (Ahmed et al. 2022, del Rio Chanona et al. 2021). The objective function is approximated with a GP:

|  |  |
| --- | --- |
|  | (3) |

where is the objective function metric, , represents the GP approximation of the objective function which depends on the set point, . As the system is operated and data is obtained from the system in the form of measurements, the GP regression problem can be re-solved with additional data to give an up-to-date model. We denote measurements from the system up to the kth sampling instant as and of the objective value and state measurements respectively. Consequently, a data-driven approximation of the objective function, , can be obtained. This is achieved by fitting a GP on the input, , and output datasets, . The GP can be thought of as a multivariate Gaussian distribution over functions from which a function, , can be sampled:

|  |  |
| --- | --- |
|  | (4) |

where and , represent the mean function and covariance function of the GP trained with the input-output dataset (Rasmussen 2005).

* 1. Results and Discussion

The case study we employ in this paper is a multi-loop evaporator process (Ahmed et al. 2023). The system consists of a feed stream containing two components: a non-volatile solute dissolved in a volatile solvent. Heat is supplied via a steam line to evaporate the solvent. The evaporator has both vapour and liquid outlets. Our control strategy employs three PI controllers. The primary controlled variable is the solute composition in the liquid product stream, , which is controlled via the steam temperature, . We control the liquid level, , by manipulating the liquid product stream flowrate, , and the evaporator pressure, , which is controlled by adjusting the vapour flow rate, . The controller tuning was performed using a modified version of the sequential relay auto-tuning method combined with a derivative free optimizer (py-BOBYQA). The system model can be found in Ahmed et al. (2023). The RTO goal is to find the set-points for the states, which maximize the profit of the process subject to operational and safety constraints:

|  |  |
| --- | --- |
|  | (6) |

The objective function and constraints for the “true” system is depicted in the left subplot of Figure 3 while the mismatched model equivalent is shown in the right subplot. It is clear to see that there is a stark difference between the two plots due to the model mismatch.

Figure 3: Objective function represented as a colour gradient from red to green indicating low and high profit respectively. Left: “True” system objective function and constraints. Right: Mismatched model objective function and constraints.

We consider 4 scenarios to demonstrate the importance of adopting both ARRTOC and GP adaptation. For all scenarios, we run simulations of 100 timesteps. We use either nominal RTO or the ARRTOC algorithm to solve Eq. (6) with the assumed model of the system depending on the scenario. The chosen set-points are then sent to the PI controllers and data is collected from the system. For the scenarios which employ GP adaptation, the data is used to correct the model every 10 timesteps as per section 2.2 and the RTO problem is re-solved. We perform 10 simulations for each scenario and report the mean profit delivered over the length of the simulation in Figure 4. Scenarios 1 and 2 serve as performance benchmarks, defining the best and worst-case situations. Scenario 1, the best-case, represents ARRTOC solved assuming perfect model knowledge. This is the best case as it accounts for the underlying controller robustness via ARRTOC and returns an accurate RTO solution as there is no model mismatch. This is depicted as a horizontal dashed green line in Figure 4. Scenario 2, the worst-case, represents nominal RTO solved with the incorrect model with no adaptation. This case neither accounts for the underlying controller design via ARRTOC nor does it address the plant-model mismatch. This is depicted as a horizontal dashed red line in Figure 4. Scenarios 3 and 4 demonstrate the significance of employing both ARRTOC and GP adaptation in tandem. Scenario 3 showcases ARRTOC with GP adaptation as a solid green line in Figure 4, while Scenario 4 illustrates nominal RTO with GP adaptation as a solid red line. Clearly, ARRTOC with adaptation outperforms nominal RTO with adaptation as it tends towards the best-case performance. This arises from the following distinction: while both approaches address the plant-model mismatch through GP adaptation, nominal RTO lacks an essential element, which is accounting for the underlying controller robustness. In the absence of such consideration, nominal RTO may lead to the selection of a set-point that is incompatible with the controllers, akin to the concept illustrated in Figure 1. On the contrary, ARRTOC ensures that the chosen set-points align with the underlying controller design, while GP adaptation diligently maintains model accuracy. This synergistic combination results in superior performance and more effective control in real-world applications.

Figure 4: Scenario 1, best-case, depicted as a dashed green horizontal line represents ARRTOC with perfect model knowledge. Scenario 2, worst-case, depicted as a dashed red horizontal line represents nominal RTO with the mismatched model and no adaptation. Scenario 3, solid green line, showcases ARRTOC with GP adaptation. Scenario 4, solid red line, showcases nominal RTO with GP adaptation.

* 1. Conclusions

In conclusion, the ARRTOC algorithm, together with the AGPL extension, address the fundamental challenges of managing implementation errors and plant-model mismatch in a holistic manner. ARRTOC's ability to identify set-points robust to disturbances and noise, while AGPL's dynamic Gaussian Process adaptation overcomes plant-model disparities, collectively contribute to more adaptable and resilient process operations. The successful integration of these machine learning techniques with process systems engineering offers a promising pathway towards achieving enhanced operability and precision in real-time control strategies, ultimately facilitating more efficient and sustainable process operations.

References

A. Ahmed, E. A. del Rio-Chanona, M. Mercangöz, “ARRTOC: Adversarially Robust Real Time Optimization and Control”, arXiv preprint, arXiv:2309.04386, 2023.

A. Ahmed, M. Zagorowska, E. A. del Rio-Chanona, M. Mercangöz, “Application of gaussian processes to online approximation of compressor maps for load-sharing in a compressor station,” European Control Conference (ECC), 2022.

D. Bertsimas, O. Nohadani, and K. M. Teo, “Robust optimization for unconstrained simulation-based problems,” Operations Research, vol. 58, 2010.

B. Chachuat, B. Srinivasan, D. Bonvin, “Adaptation strategies for real-time optimization,” Computers & Chemical Engineering, vol. 33, 2009.

A. Cortinovis, M. Mercangöz, M. Zovadelli, D. Pareschi, A. De Marco S. Bittanti, “Online performance tracking and load sharing optimization for parallel operation of gas compressors,” Computers & Chemical Engineering, vol. 88, 2016.

E. A. del Rio-Chanona, P. Petsagkourakis, B. Eric, J. E. A. Graciano, B. Chachuat, “Real-time optimization meets Bayesian optimization and derivative-free optimization: A tale of modifier adaptation,” Computers & Chemical Engineering, vol. 147, 2021.

D. F. Mendoza, J. E. A. Graciano, F. S. Liporace G. Carrilo Le Roux, “Assessing the reliability of different real-time optimization methodologies,” The Canadian Journal of Chemical Engineering , vol. 94, 2015.

G. D. Patrón and L. Ricardez-Sandoval, “An integrated real-time optimization, control, and estimation scheme for post-combustion co2 capture,” Applied Energy, vol. 308, 2022.

C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2005.