Learning of dynamical systems from a finite number of closed loop data points

Mukul Chodhary
1172562, mchodhary@student.unimelb.edu.au

Conor Hars
914787, chars@student.unimelb.edu.au

Jake Piddington
834267, jpiddington@student.unimelb.edu.au

***Executive Summary**: Model-based control strategies are widely employed to regulate dynamical systems, ensuring trajectory tracking and stability. However, these approaches rely on accurate system models, which may not always be available due to unmodeled dynamics, parameter variations, or environmental uncertainties. This project investigates the application of the Sign-Perturbed Sums (SPS) algorithm to estimate the system dynamics using a finite number of closed-loop data points. The SPS method provides a set of system models with exact probabilistic guarantees that the true system lies within this set. By continuously updating this set in real time with new data, we aim to enhance the robustness and adaptability of control strategies, mitigating performance degradation due to model uncertainty.We developed a modular software architecture that integrates SPS seamlessly into real-time control loops. The architecture supports asynchronous operation, allowing the computationally intensive SPS process to run on a separate processor or device, ideal for distributed systems or embedded applications with limited resources. This design provides flexibility for deployment across various platforms and simplifies the integration of SPS with existing controllers. By combining real-time SPS-based model updates with robust control synthesis, this project enables adaptive, data-driven closed-loop control. Bayesian methods are employed to fuse SPS confidence regions, refining model estimates over time and improving both safety and performance in dynamic or uncertain environments. This framework enhances the reliability of autonomous systems under changing conditions and offers a practical approach for deploying advanced system identification techniques in real-world control applications.*

1 Introduction

Control systems play a crucial role in modern engineering, enabling systems to achieve desired behaviors such as stabilisation, tracking, and disturbance rejection. Feedback control is most common in practice, allowing systems to maintain desired performance in the face of model uncertainty and disturbances, without requiring manual intervention or a sophisticated understanding of the uncertain parameters. This capability is particularly important in tasks such as reference tracking, where the system output must

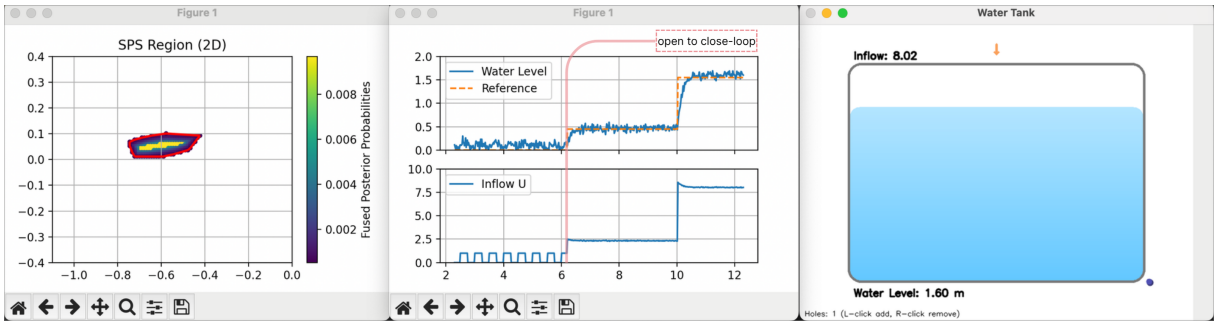


Figure 1: Water tank simulation with a water leak, performing real-time reference tracking. Initial open-loop operation identifies the first set of plants, followed by closed-loop control.

follow a predefined trajectory, as seen in robotic manipulators, autonomous vehicles, and satellite attitude control systems. The capacity to operate autonomously under varying environmental conditions makes control systems essential in both industrial automation and safety-critical applications.

Another key function of control theory is ensuring stability and robustness in the presence of uncertainties. Concepts like gain, phase, and disk margins provide rigorous methods for designing systems that remain stable despite model inaccuracies, measurement noise, and external disturbances. Mathematical tools such as Lyapunov stability theory (Khalil, 2002) are central to establishing guarantees for a system's controllability and observability, under both open and closed-loop control. Furthermore, advanced control strategies like robust adaptive control (Ioannou & Sun, 1995) and Model Predictive Control (MPC) (Kirk, 2004) enable systems to meet performance objectives while minimising energy consumption or control effort (Ogata, 2010). Control theory underpins the robust and effective operation of real-world control systems, with a variety of applications spanning robotics, aerospace, biomedical devices, and large-scale infrastructure systems.

Model-based control approaches dominate in practice. With a system model, control engineers can leverage results from control theory to design robust, high-performance controllers before deployment on the real system. However, the real-world effectiveness of any controller hinges on the accuracy of the nominal model, and, crucially, how well system uncertainties are accounted for. When the system and its uncertainties cannot be readily modelled before deployment, plant model estimation - the process of identifying mathematical representations of system dynamics from data - becomes essential. While model-free methods such as reinforcement learning (Sutton & Barto, 2018) or neural networks (Cao et al., 2022; Ljung et al., 2020; Pillonetto et al., 2014) have gained attention due to their data-driven nature, they often lack theoretical guarantees or require large amounts of data, which can be impractical or unsafe in real-time or safety-critical settings.

Traditional system identification methods typically provide asymptotic guarantees, relying on assumptions of infinite data or persistent excitation (Ljung, 1999). This restricts their applicability in real-time adaptive control schemes, where system parameters and confidence sets must be reliably estimated, often from limited data. In contrast, the Sign-Perturbed Sums (SPS) method (Csaji et al., 2015) provides a framework for constructing finite-sample confidence regions with exact, non-asymptotic probabilistic guarantees. SPS allows us to build a set of plants that, with a predefined confidence level, is guaranteed to contain the true plant. This uncertainty set can then be used to design robust controllers that perform reliably across all models in the set, enhancing the system's resilience to modeling errors and uncertainty. This represents a principled approach to combining learning with control in a manner that is both theoretically grounded and practically implementable.

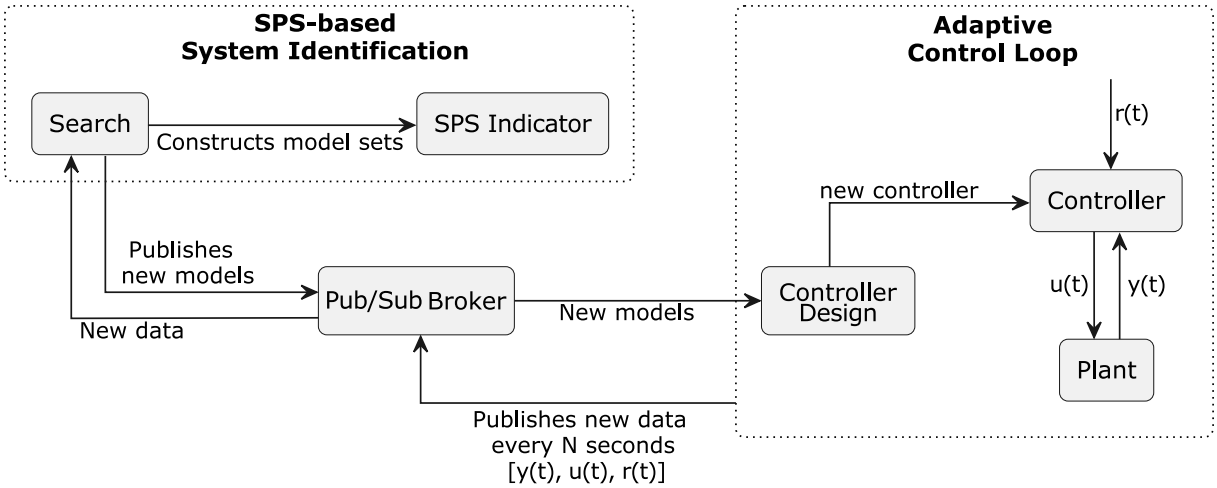


Figure 2: System Architecture: Real-time Adaptive Control Loop with Asynchronous Identification

1.1 Objectives

The aim of this project is to integrate the SPS algorithm into real-time control frameworks to enable robust, adaptive control of dynamic systems under uncertainty. To achieve this, the system architecture must be modular, efficient, and robust, supporting asynchronous operation between the SPS-based model estimation and the closed-loop controller. Specifically, the objectives of this work are to:

- Demonstrate the feasibility of using SPS for system identification in real-time, particularly for closed-loop and multi-input multi-output (MIMO) systems.
- Generate accurate confidence regions for system parameters of first- and second-order plants using SPS, and verify that these regions contain the true parameters as guaranteed.
- Implement a robust adaptive Linear Quadratic Regulator (LQR) controller that utilises SPS-generated uncertainty sets to adapt control laws safely and efficiently.
- Ensure the closed-loop system remains stable and safe during operation, with exploration confined to safe and well-defined regions for the controller and model parameter spaces.
- Show real-time adaptation capability, with convergence towards true plant parameters under parameter perturbations or initial uncertainties.
- Develop a modular architecture to enable deployment of the SPS algorithm in real-time closed-loop systems, accommodating embedded and/or distributed computing environments.

1.2 Contributions

To address the objectives outlined above, this project has produced the following key contributions:

- Design and implementation of a modular system architecture that supports asynchronous communication between SPS-based model identification and the real-time control loop, enabling operation in embedded and distributed settings.
- Development of a high-performance, Numba-accelerated Python library for efficient evaluation of MIMO transfer functions, supporting fast simulation and search over parameter space.
- Synthesis of optimal control laws that make full use of the exact uncertainty sets derived via SPS to ensure robust performance.
- Implementation of a smart, SPS-guided search algorithm that efficiently identifies confidence regions in parameter space with reduced computational overhead.

The remainder of this report is structured as follows: Section 2 reviews the SPS algorithm in both SISO and MIMO settings, comparing it with conventional system identification methods and highlighting their respective limitations. Section 3 introduces the control problem, surveys relevant literature, and proposes a robust optimal control law designed to operate over the set of plant models generated by SPS. Section 4 details efficient parameter space exploration strategies guided by the SPS indicator, along with Bayesian fusion methods that incorporate historical SPS confidence regions. Section 5 presents the modular software architecture developed for real-time integration of SPS with control systems. Section 6 showcases experimental results demonstrating real-time system identification and adaptive control performance. Section 7 discusses the implications of the results, analyses limitations of the current work, and suggests potential improvements. Finally, Section 8 concludes the report and outlines directions for future work. Additional technical details and supplementary material are provided in the Appendices.

2 Sign-Perturbed Sums (SPS)

Sign-Perturbed Sums (SPS) is a probabilistically robust method for system identification that utilises finite data samples to construct non-asymptotic confidence regions for the true system parameters. Under mild assumptions, these regions are guaranteed to contain the true parameter vector with a user-specified confidence level (Csaji et al., 2015). SPS was originally developed for single-input single-output (SISO) systems operating under open-loop conditions, and a closed-loop extension was subsequently proposed in Csáji and Weyer (2015). The system model considered is:

$$Y_t = G(z^{-1}, \theta^*)U_t + H(z^{-1}, \theta^*)N_t, \tag{1}$$

where Y_t denotes the output, U_t the input, and N_t the disturbance at discrete time t. The transfer functions G and H are assumed to be causal and rational, and z^{-1} denotes the backward shift operator. This section introduces the classical SISO SPS formulation and outlines its extension to multiple-input multiple-output (MIMO) systems.

Recursive variants of SPS are briefly discussed in Appendix D. Although the recursive formulation naturally incorporates historical information, it is excluded from the final methodology due to its substantial memory overhead, which poses scalability challenges in high-dimensional scenarios. In particular, recursive SPS requires storing auxiliary data for all evaluated parameter vectors across a fine grid, resulting in significant resource demands. To address this limitation, we propose efficient search strategies in Section 4.1 and fusion-based heuristics in Section 4.2, which aim to aggregate past information across the parameter space without incurring excessive computational or memory costs.

2.1 Literature Review

System identification plays a central role in adaptive control, with reliable *confidence sets* desirable along with the nominal parameter estimate. Since the controller is synthesised and deployed automatically, knowledge of the uncertainty in parameter estimates is critical. With only point estimates, sensitivity to model mismatch is opaque, and thus a synthesised controller risks degrading or destabilising the system.

The classic approach to system identification is based on the least squares estimate (LSE), which can establish an ellipsoidal confidence region (Ljung, 1999) containing the true system parameter with some user-chosen probability. While computationally light, the validity of the confidence region is guaranteed only *asymptotically*; examples demonstrating its insuitability for small sample sizes are well-known in the literature (Garatti et al., 2004). In contrast, finite-sample methods establish confidence regions with rigorous guarantees from limited data. Leave-out Sign-dominant Correlation Regions (LSCR) is one such method, offering a framework for constructing confidence regions containing the true parameter with

guaranteed user-chosen probability in any finite-sample setting. However, the LSCR framework tends to be conservative in high dimensional settings (Carè et al., 2018), with confidence sets that contain the true parameter with *at least* the user-specified probability - not an exact coverage. In contrast, Sign-Perturbed Sums (SPS) yields guaranteed finite-sample confidence regions with *exact* user-chosen probability (Csaji et al., 2015). In high-dimensional settings, this becomes particularly advantageous over LSCR. Though SPS is more computationally demanding, effective inner and outer ellipsoidal approximations of the SPS confidence region are known and can be efficiently solved via a convex program ((Csaji et al., 2015)). The SPS algorithm is also applicable in the closed-loop setting, with direct, indirect, and joint input-output methods established in (Csáji and Weyer, 2015).

Adaptive control strategies have been established based on exploration-exploitation, with the Probably Approximately Correct (PAC) framework often used to establish rigorous guarantees for such adaptive control strategies, ensuring (ϵ, δ)-optimality - roughly speaking, the algorithm converges to an ϵ-accurate hypothesis with probability of at least $1 - \delta$. A key advantage of the PAC framework is finite-time performance guarantees, crucial for real-world applications. However, while rigorous, establishing PAC bounds can introduce non-trivial conservatism (Fiechter, 1997).

In online adaptive control, confidence sets with finite-sample guarantees provide attractive robustness and adaptability in comparison with asymptotic confidence set construction and point estimate methods. While LSCR and PAC approaches provide attractive probabilistic guarantees, conservatism tends to limit performance and applicability, whereas SPS can provide finite-sample non-asymptotic guarantees with *exact* probability, providing a robust and practical foundation for an online adaptive control scheme.

Summary of Methods

2.2 SISO SPS

In open-loop setting, it is assumed that U_t and N_t are independent

The prediction error is then defined as,

$$\hat{N}_t(\theta) \triangleq H^{-1}(z^{-1}, \theta)(Y_t - G(z^{-1}, \theta)U_t) \tag{2}$$

and the prediction error estimate, $\hat{\theta}_n$, for points, given by quadratic cost criterion is

$$\hat{\theta}_n \triangleq \underset{\theta \in \Theta}{\arg\min} \sum_{t=1}^{n} \hat{N}_t^2(\theta) \tag{3}$$

where Θ is the set of allowed models. The solution for the optimisation problem satisfies the "normal" equation

$$\begin{aligned} &\sum_{t=1}^{n} \hat{N}_t(\hat{\theta}_n)\psi_t(\hat{\theta}_n) = 0, \\ &\text{with } \psi_t(\theta) = \left[\frac{\partial}{\partial\theta}\hat{N}_t(\theta)\right]^T, \end{aligned} \tag{4}$$

where ψ_t is the gradient vector of dimension d.

2.2.1 SPS open-loop indicator

The prediction error $\hat{N}_t$ is used to build m alternate output trajectories,

$$\bar{Y}_t(\theta, \alpha_i) = G(z^{-1}; \theta)U_t + H(z^{-1}; \theta)(\alpha_{i,t}\hat{N}_t(\theta)) \tag{5}$$

Let $\{\alpha_{i,t}\} \in \mathbb{R}^{m\times n}$ denote a set of sign sequences used to construct output trajectories, where $i \in \{0, 1, \ldots, m-1\}$ indexes each trajectory and $t \in \{0, \ldots, n-1\}$ indexes time.

- For $i = 0$, set $\alpha_{0,t} = 1$ for all t, which yields the original (unperturbed) output trajectory .
- For each $i \in \{1, \ldots, m-1\}$, the signs $\alpha_{i,t}$ are drawn independently as random variables with $\Pr(\alpha_{i,t} = \pm 1) = \frac{1}{2}$. These random signs are applied pointwise over the trajectory to construct $m-1$ sign-perturbed output trajectories.

Algorithm 1: Open Loop SPS indicator

Input: θ, Y_t, U_t

Output: in_sps_region ▷ bool

Function `Open_Loop_SPS_Indicator()`:

- $G(z^{-1}, \theta), H(z^{-1}, \theta) \longleftarrow Map\_to\_filters(\theta)$ ▷ causal G, H
- $\alpha \longleftarrow sign(random(m, n))$;
- $\alpha[0, :] \longleftarrow 1$
- $\hat{N}_t \longleftarrow Compute\_N\_hat$ ▷ Eq. 2
- $\bar{Y}_t \longleftarrow Compute\_Perturbed\_Trajectories(\hat{N}_t, \alpha, G, H, U_t)$ ▷ Eq. 5
- $\bar{\psi}_t \longleftarrow Compute\_Gradients(\hat{N}_t, \alpha, G, H, \bar{Y}_t, U_t)$
- $S \longleftarrow Compute\_S(\bar{\psi}_t, \hat{N}_t)$ ▷ Eq. 6
- $R \longleftarrow L2\_Norms(S)$
- $\mathcal{R} \longleftarrow Sort\_And\_Rank\_0(R)$ ▷ Eq. 7 - 8
- **return** $\mathcal{R} \leq m - q$

End Function

The gradient vector $\psi_t(\theta)$ can be written in terms of vector-valued linear filters W and Q,

$$\bar{\psi}_t(\theta, \alpha_i) = W(z^{-1}; \theta)\bar{Y}_t(\theta, \alpha_i) + Q(z^{-1}; \theta)U_t(\theta)$$

The metric $S_i(\theta)$ quantifies how well a candidate parameter θ fits the data when the prediction errors are perturbed by the i-th random sign sequence. It is defined as

$$\begin{aligned} S_i(\theta) &\triangleq \bar{\Psi}_n^{-\frac{1}{2}}(\theta, \alpha_i) \sum_{t=1}^{n} \alpha_{i,t}\bar{\psi}_t(\theta, \alpha_i)\hat{N}_t(\theta), \\ \bar{\Psi}_n(\theta, \alpha_i) &\triangleq \sum_{t=1}^{n} \bar{\psi}_t(\theta, \alpha_i)\bar{\psi}_t^T(\theta, \alpha_i). \end{aligned} \tag{6}$$

The summation $\sum_{t=1}^{n} \alpha_{i,t}\bar{\psi}_t(\theta, \alpha_i)\hat{N}_t(\theta)$ combines the prediction errors with their corresponding sensitivity vectors, modulated by the random sign sequence $\alpha_{i,t}$. This construction captures how well the parameter θ aligns with the observed data under a particular realisation of perturbed residuals. Larger values of this sum indicate stronger alignment between the residuals and the model sensitivity, while smaller values suggest weaker alignment under the given parameter. To ensure the metric is scale-invariant across parameter directions, the sum is normalised by the inverse square root of the empirical covariance matrix $\bar{\Psi}_n(\theta, \alpha_i)$. As a result, $S_i(\theta)$ provides a normalised, scale-invariant summary of the perturbed, weighted residuals, and its squared norm $\|S_i(\theta)\|^2$ serves as a scalar measure to rank the quality of the parameter fit across all perturbed trajectories.

The collection of scores $R = \{R_i = \|S_i(\theta)\|^2\}, i \in \{0, \ldots, m-1\}$, is then strictly ordered in ascending order. To resolve ties, a uniform random permutation ν is applied over the indices to enforce a strict ranking, such that

$$\begin{aligned} R_k \succ_\nu R_j \iff \\ (R_k > R_j) \text{ or } (R_k = R_j \text{ and } \nu(k) > \nu(j)). \end{aligned} \tag{7}$$

The rank of the unperturbed score R_0, denoted $\mathcal{R}(\theta)$, determines whether the parameter θ lies within the confidence region. Specifically, the SPS confidence region is defined as

$$\hat{\Theta}_n \triangleq \{\theta \in \Theta : \mathcal{R}(\theta) \leq m - q\}. \tag{8}$$

This means that θ is accepted if its unperturbed score ranks among the lowest $m - q$ values. Conversely, if the rank $\mathcal{R}(\theta)$ exceeds $m - q$, it indicates that the parameter θ fits the randomly perturbed residual trajectories better than the original unperturbed trajectory. This suggests that θ is less consistent with the true data and therefore falls outside the SPS confidence region, leading to its rejection as a plausible parameter estimate.

The SPS indicator algorithm, summarised in Algorithm 1, returns `true` when this condition holds.

2.2.2 Closed-loop SPS

In this project, we focus on the indirect identification method introduced in Csáji and Weyer, 2015. The indirect identification method runs open-loop SPS on the open-loop equivalent of the original system (Eq. 1). The feedback controller is of the form,

$$U_t = L(z^{-1})R_t - F(z^{-1})Y_t \tag{9}$$

with known F and L.

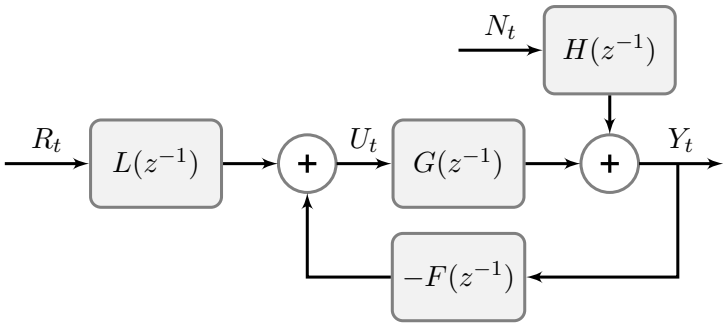


Figure 3: General closed-loop structure considered

The open-loop equivalent system of (Eq. 1) is

$$\begin{aligned} Y_t &= G_o(z^{-1};\kappa^*)R_t + H_o(z^{-1};\kappa^*)N_t \\ G_o(z^{-1};\kappa^*) &\triangleq (1+GF)^{-1}GL \\ H_o(z^{-1};\kappa^*) &\triangleq (1+GF)^{-1}H \end{aligned} \tag{10}$$

The gradient vector $\psi_t(\theta)$ defined with linear filters W and Q,

$$\begin{aligned} \bar{\psi}_t(\theta,\alpha_i) &= W(z^{-1};\theta)\bar{Y}_t(\theta,\alpha_i) + Q(z^{-1};\theta)\bar{U}_t(\theta,\alpha_i), \\ \text{and, } \bar{U}_t(\theta,\alpha_i) &= L(z^{-1},\eta^*)R_t - F(z^{-1},\eta^*)\bar{Y}_t(\theta,\alpha_i) \end{aligned}$$

ARMAX model class

Assuming original system belongs to the ARMAX model class,

$$A(z^{-1},\theta)Y_t = B(z^{-1},\theta)U_t + C(z^{-1},\theta)N_t$$

$$A(z^{-1},\theta) \triangleq 1 + \sum_{i=1}^{d_A} a_i z^{-i}, B(z^{-1},\theta) \triangleq \sum_{i=1}^{d_B} b_i z^{-i}, C(z^{-1},\theta) \triangleq \sum_{i=0}^{d_C} c_i z^{-i}$$

where $\theta = [a_1, \ldots, a_{d_A}, b_1, \ldots, b_{d_B}, c_0, \ldots, c_{d_C}]$, the transfer functions G, H are defined as

$$G(z^{-1}, \theta) = \frac{B(z^{-1}, \theta)}{A(z^{-1}, \theta)}, H(z^{-1}, \theta) = \frac{C(z^{-1}, \theta)}{A(z^{-1}, \theta)}$$

The open loop equivalent system is given by,

$$\begin{aligned} G_o(z^{-1}; \kappa^*) &= (A + BF)^{-1} BL \\ H_o(z^{-1}; \kappa^*) &= (A + BF)^{-1} C \end{aligned} \tag{11}$$

and the closed-loop prediction error is given by

$$\hat{N}_t(\theta) \triangleq H_o^{-1}(z^{-1}, \theta)(Y_t - G_o(z^{-1}, \theta) R_t) \tag{12}$$

Now the open-loop SPS algorithm can be run with Y_t, R_t pair and G_o, H_o. The Indirect SPS algorithm, Algorithm 2, returns $true$ when the original θ is in the confidence region.

Algorithm 2: Indirect SPS indicator

Input: $\theta, F(z^{-1}), L(z^{-1}), Y_t, R_t$ ▷ causal G, H
Output: in_sps_region ▷ bool

Function `Indirect_SPS_Indicator()`:
  $G_o(z^{-1}, \kappa) H_o(z^{-1}, \kappa) \longleftarrow convert\_to\_open\_loop(\theta, F, L)$;
  $in\_sps\_region \longleftarrow Open\_Loop\_SPS\_Indicator(\kappa, Y_t, R_t)$
  return in_sps_region
End Function

2.2.3 Assumptions

The required conditions for Sign-Perturbed Sums (SPS) are often satisfied by linear time-invariant (LTI) systems in practice (Csáji & Weyer, 2015). While some assumptions, particularly on the noise, are relatively mild, the method requires that the true system lies within the assumed model class, which is a stronger condition. For completeness, the full set of assumptions is:

1. The model orders of $G(z^{-1}; \theta)$ and $H(z^{-1}; \theta)$ are correctly specified, and the true system lies within the chosen model class. Additionally, the noise model $H(z^{-1}; \theta)$ has a stable inverse, and the transfer functions satisfy $G(0; \theta) = 0$ and $H(0; \theta) = 1$ for all $\theta \in \Theta$. *(This is the strongest assumption, as it requires correct model structure.)*
2. The noise sequence $\{N_t\}$ consists of independent random variables, each with a symmetric distribution about zero.
3. The noise sequence $\{N_t\}$ is independent of the reference signal $\{R_t\}$.
4. The subsystems from $\{N_t\}$ and $\{R_t\}$ to the output $\{Y_t\}$ are asymptotically stable and contain no unstable hidden modes.
5. The system initialisation is known for all $t \leq 0$.

2.3 MIMO SPS

Consider a general MIMO, linear time-invariant (LTI) system,

$$Y_t \triangleq G(z^{-1}, \theta^*) U_t + H(z^{-1}, \theta^*) N_t \tag{13}$$

where, like the SISO case, t denotes discrete time, z^{-1} is the backward shift operator, $Y_t \in \mathbb{R}^{q\times 1}$ denotes the output vector, $U_t \in \mathbb{R}^{r\times 1}$ denotes the input vector, $G : q \times r$ and $H : q \times q$ are causal matrix filter and $\theta \in \mathbb{R}^{d\times 1}$, i.e. there are d free parameters.

The feedback rule is given by

$$U_t \triangleq L(z^{-1}, \theta^*)R_t - F(z^{-1}, \theta^*)Y_t \tag{14}$$

where $R_t \in \mathbb{R}^{s\times 1}$ is the reference signal, $L : r \times s$ and $F : r \times q$ are causal matrix filters.

2.3.1 Predictor Error method (PEM)

The PEM estimate is given by

$$\hat{\theta}_n \triangleq \underset{\theta\in\Theta}{\arg\min} \sum_{t=1}^{n} \|\epsilon_t(\theta)\|_\Lambda^2 = \underset{\theta\in\Theta}{\arg\min} \sum_{t=1}^{N} \epsilon_t^\top(\theta)\Lambda^{-1}\epsilon_t(\theta) \tag{15}$$

where Λ is a positive-definite weighting matrix or the covariance matrix of the noise. The solution of the optimisation problem satisfies the normal equation,

$$\begin{aligned} &\sum_{t=1}^{N} \psi_t(\hat{\theta}_{PEM})\Lambda^{-1}\epsilon_t(\hat{\theta}_{PEM}) = \mathbf{0}, \\ &\text{with } \psi_t(\theta) = \left[\frac{\partial}{\partial\theta}\epsilon_t(\theta)\right]^\top, \end{aligned} \tag{16}$$

where ψ_t is of dimensions $d \times q$.

The MIMO prediction errors takes similar form to SISO prediction errors (2), (12).

Open-loop prediction error:

$$\epsilon_t(\theta) \triangleq H^{-1}(z^{-1}, \theta)(Y_t - G(z^{-1}, \theta)U_t) \tag{17}$$

Closed-loop prediction error:

$$\epsilon_t(\theta) \triangleq H_o^{-1}(z^{-1}, \theta)(Y_t - G_o(z^{-1}, \theta)R_t) \tag{18}$$

where the G_o, H_o are open-loop equivalent of the closed-loop system.

$$\begin{aligned} G_o &= (I + GF)^{-1}GL \\ H_o &= (I + GF)^{-1}H \end{aligned} \tag{19}$$

The ϵ_t can be written in terms of G, H, F, L

$$\begin{aligned} \epsilon_t(\theta) &= H^{-1}(I + GF)(Y_t - (I + GF)^{-1}GLR_t) \\ &= H^{-1}((I + GF)Y_t - GLR_t) \end{aligned} \tag{20}$$

2.3.2 Estimating the Noise Covariance Matrix Λ

When Λ is chosen as the covariance matrix of the prediction error $\epsilon_t(\theta)$, it captures not only the variability of the noise but also the effect of estimating the noise dynamics. A common approach is to use the

unbiased sample covariance matrix (SCM) based on the available data. Given the prediction error matrix $\epsilon(\theta) \in \mathbb{R}^{q \times N}$, the SCM is computed as:

$$\Lambda = \frac{1}{N-1} \sum_{t=1}^{N} \left(\epsilon_t(\theta) - \bar{\epsilon}(\theta)\right) \left(\epsilon_t(\theta) - \bar{\epsilon}(\theta)\right)^\top$$

where the sample mean $\bar{\epsilon}(\theta)$ is defined by:

$$\bar{\epsilon}(\theta) = \frac{1}{N} \sum_{t=1}^{N} \epsilon_t(\theta)$$

This formulation ensures that the covariance estimate remains unbiased, which is important when accounting for the contribution of the estimated noise model in subsequent steps.

2.4 Star centre property of PEM estimate

It is often advantageous to compute the prediction error method (PEM) estimate to guide the search over the parameter space. In the open-loop case, this is particularly effective due to the star-convexity of SPS confidence regions for FIR models, with the PEM estimate acting as the star centre (Csaji et al., 2015). This geometric property can be exploited to inform point selection strategies within the SPS region search algorithm.

Although star-convexity is not guaranteed in the closed-loop setting, the PEM estimate remains a useful anchor point for initiating the SPS region search, as it provides a well-informed and computationally efficient starting location. This strategy is further elaborated in Section 4.1.

2.5 Evaluating ψ_t

While the evaluation of ψ_t may initially appear straightforward, it is in fact strongly influenced by the construction of the parameter vector θ, which is dependent on the model class. The efficiency and tractability of this evaluation can be significantly improved by exploiting known model structures or employing canonical representations. In this work, we adopt a state-space formulation that shares parameters with the noise model, consistent with an ARMAX-type structure. For conciseness, the construction of θ and the corresponding evaluation of ψ_t for both SISO and MIMO systems are deferred to Appendix B.

3 Robust Control

The results of SPS are a set of discrete-time transfer functions. In order to synthesise a controller for these systems, we chose to use the LQR cost function with infinite horizon. While a finite horizon or MPC approach can yield better transient performance, robust controller synthesis is often computationally intensive and thus by the time an optimal finite horizon control scheme has been synthesised, the system state will likely have diverged from that assumed during optimisation; in contrast, an infinite horizon approach yields a static time-invariant controller that remains suitable across the expected operating conditions, without requiring continual re-optimisation. Thus, an infinite horizon approach ensures that other essential modules retain the processing capacity they need to operate effectively, such as SPS set construction. Finally, since all modules operate asynchronously, a consistent time-invariant controller is preferable. Hence we focus on synthesising the controller $u_k = -Kx_k$ with the quadratic cost function

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k\right)$$

Next, controller synthesis must consider the uncertainty region of (A, B). There are several established approaches. *Thompson sampling*, as in Kargin et al., 2022, uses a posterior probability distribution to pick a reasonable candidate (A_i, B_i) from the uncertainty region, and optimises J for only this plant. This is a heuristic, computationally cheap approach which tends to be effective in practice, however there is no consideration of worst-case performance, or even stability. *Regret bounds*, as in Abeille and Lazaric, 2018, seek to minimise the performance gap between the optimal cost J^* and the implemented controller $J(K)$. *Optimism in the face of uncertainty* (OFU), as in Abbasi-Yadkori and Szepesvári, 2011, assumes the plant with best-case dynamics, and finds the optimal controller for this plant. In contrast, *design for worst-case dynamics*, as in Umenberger et al., 2019, aims to minimise the worst-case J across the entire uncertainty region.

For controller synthesis, we use the latter approach and design a controller that ensures optimal *worst-case* performance, i.e. the optimisation problem which defines the optimal K is

$$\min_{K} \max_{(A,B)\in\Theta} J, \quad \text{subject to } \begin{cases} x_{k+1} = Ax_k + Bu_k \\ (A, B) \in \Theta \\ u_k = -Kx_k \end{cases} \tag{21}$$

where Θ is the set of state space matrices describing the system, identified via SPS and conversion to OCF. This approach appropriately leverages the guaranteed confidence regions provided by SPS, so as to give guaranteed robust performance.

Finally, we use Observable Canonical Form (OCF) for the state-space representation of transfer functions, as in B.2.1.2. This yields unique A and B from each transfer function, but a fixed C matrix for all plants, along the lines of $C = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}$. In this way, the output matrix is fixed across all state-space realisations, and we can restrict attention to optimising over the (A_i, B_i) with no need to consider how different C affects the state representations. The potential downsides are a) numerical conditioning can be poor with OCF and b) while $x_N = y$ is clear, the states $x_0, \dots, x_{N-1}$ do not have a clear interpretation, making the interpretation of the LQR cost w.r.t. these states less transparent.

3.1 Literature review

The problem of synthesising a feedback controller in the presence of plant uncertainty is a key problem in robust control theory, and has been the subject of considerable research. Additionally, consideration of a worst-case LQR cost has also been studied for some uncertainty structures.

Continuous-time
In a continuous-time setting, Petersen and Hollot, 1986 introduce an augmented Riccati equation, which extends the linear quadratic regulator design procedure to uncertain systems. The uncertainty considered is time-varying yet norm-bounded, s.t. it can be parametrised as follows

$$\begin{gathered} \dot{x}(t) = \left[A_0 + \sum_{i=1}^{k} A_i r_i(t)\right] x(t) + \left[B_0 + \sum_{i=1}^{l} B_i s_i(t)\right] u(t); \\ |r_i(t)| \le \bar{r}, \quad i = 1, 2, \dots, k; \quad \bar{r} \ge 0; \\ |s_i(t)| \le \bar{s}, \quad i = 1, 2, \dots, l; \quad \bar{s} \ge 0; \end{gathered} \tag{22}$$

where the A_i and B_i are rank-one matrices. The presented result stems from the construction of a common Lyapunov function $V(x) = x^T P x$ and demonstrating that it is stable for all admissible plant uncertainties; furthermore, a procedure allows one to synthesise a feedback gain with some guaranteed upper bound on the Lyapunov derivative, referred to as the "quadratic bound method" to synthesise a stabilising controller

for the uncertain plants. The LQR cost is not explicitly considered, though the authors show that it is recovered when there is no uncertainty and thus the procedure can be considered analogous to an LQR problem across a set of uncertain plants.

Again considering continuous-time systems with this uncertainty structure, Chang and Peng, 1972 present a similar result, again based on an augmented Riccati equation, which allows one to synthesise a controller with *guaranteed cost control*. They remark on the complexity and expense of computing an explicit solution for a minimax-optimal control law, and propose this method as a computationally feasible alternative.

Discrete-time

Garcia et al., 1994, p. 1 remark how the extension of continuous-time results to discrete-time is not usually straightforward, "due to the relative complexity of the discrete Lyapunov equation which is nonlinear with respect to the dynamical matrix, and when uncertain systems are considered the uncertainty matrix also appears in a nonlinear manner". They describe a solution for a stabilising controller for a system with norm-bounded time-varying uncertainty with the following structure

$$\begin{aligned} x_{k+1} &= (A + \Delta A)x_k + (B + \Delta B)u_k \\ \begin{bmatrix}\Delta A, & \Delta B\end{bmatrix} &= DF(k)(E_1 \quad E_2) \\ F(k) : F(k)'F(k) &\leq I \end{aligned} \tag{23}$$

which is quite a common structure for results in the literature, known as "matching conditions". Xie and Soh, 1993 assume the same description of the plant uncertainty and demonstrate a method for guaranteed LQR cost control, thus extending the results of Chang and Peng, 1972 to discrete-time.

An alternative, less structured uncertainty structure is considered in Fang and Chang, 1993, where the A_{ij} assume some nominal value plus a bounded perturbation

$$|\Delta A|_m \leq qH. \tag{24}$$

The authors present an algorithm for assigning the poles of the closed-loop system within a user-specified disk.

Finally, de Oliveira et al., 1999 analyse the robust stability of discrete-time systems with polytopic uncertainty

$$\begin{aligned} x_{k+1} &= A(\alpha)x_k + B(\beta)u_k \\ \mathscr{A} &:= \left\{ A(\alpha) : A(\alpha) = \sum_{i=1}^{N} \alpha_i A_i,\ \sum_{i=1}^{N} \alpha_i = 1,\ \alpha_i \geq 0 \right\}, \\ \mathscr{B} &:= \left\{ B(\beta) : B(\beta) = \sum_{i=1}^{M} \beta_i B_i,\ \sum_{i=1}^{M} \beta_i = 1,\ \beta_i \geq 0 \right\} \end{aligned} \tag{25}$$

where the $A(\alpha)$ and $B(\beta)$ are allowed to vary independently within their convex polytope sets. A sufficient condition formulated as a set of Linear Matrix Inequalities (LMI's) is presented for determining the robust stability of the uncertain plant

$$\begin{bmatrix} P_{ij} & A_i G + B_j L \\ G'A_i' + L'B_j' & G + G' - P_{ij} \end{bmatrix} > 0 \qquad \forall\, i = 1, \ldots, N,\ j = 1, \ldots, M \tag{26}$$

where P_{ij} are symmetric matrices and G is square. The solution also yields a simultaneously stabilising controller $K = LG^{-1}$.

In Peaucelle 2000, an analogous result is presented for the robust stability of discrete-time systems with polytopic uncertainty

$$x_{k+1} = A(\gamma)x_k + B(\gamma)u_k$$
$$\mathscr{AB} := \left\{ (A(\gamma), B(\gamma)) : A(\gamma) = \sum_{i=1}^{N} \gamma_i A_i,\ B(\gamma) = \sum_{i=1}^{N} \gamma_i B_i,\ \sum_{i=1}^{N} \gamma_i = 1,\ \gamma_i \geq 0 \right\} \tag{27}$$

i.e. the result from Oliveira 2000 is extended to uncertain systems where the uncertainty in A and B matrices are no longer independent but *coupled*. Similarly, the LMI problem is

$$\begin{bmatrix} P_i & A_iG + B_iL \\ G'A_i' + L'B_i' & G + G' - P_i \end{bmatrix} > 0 \qquad \forall\, i = 1, \ldots, N \tag{28}$$

where again the P_i are symmetric matrices, G is square, and a simultaneously stabilising controller is given by $K = LG^{-1}$

Evaluation of relevant literature
While there are powerful results available for the continuous-time case, the extension to discrete-time is not straightforward and hence only discrete-time results were considered.

Though there are many results available for systems with uncertainty described by the matching conditions, including guaranteed LQR cost control, the matching conditions are a strong assumption on the uncertainty structure. Using these algorithms, we would be constrained to scenarios where the uncertainty can be captured by the matching conditions. Furthermore, even restricting our attention to these scenarios, it is not clear how one would compute the matrices D, E_1, E_2 from a general set of identified (A, B).

Though the uncertainty of Fang and Chang, 1993 is less restrictive and easily computed from a set of identified (A, B), the uncertainty structure was considered too broad. Specifically, elements A_{ij} are allowed to vary independently; this results in an expanded confidence region, which is far too conservative in most cases.

Ultimately, the LMI method from de Oliveira et al., 1999 and Peaucelle et al., 2000 was chosen. By not initially considering the LQR cost and restricting our attention to the existence of a stabilising controller, the complexity of the problem is reduced to a *convex LMI problem*, which can be solved efficiently. Additionally, when no stabilising controller is possible over the set of uncertainties, we get an explicit statement of infeasibility from the optimiser. This can be fed back to the system identification and fusion modules s.t. a smaller uncertainty region is requested and provided. Also, we get a consistent and repeatable search time for a stabilising K, which is not guaranteed with other heuristics, and instead of endlessly searching for a stabilising controller when one doesn't exist, we would get a conclusive answer, abort the process and feed back the infeasibility status to other modules.

The convex polytope uncertainty descriptions are well-suited to our problem setup; by computing the convex hull of the SPS-identified (A, B), we can get an precise representation of the uncertainty, then synthesise a controller using this precise description. In this way, we are able to leverage the exactness of the confidence regions provided by SPS. Additionally, in higher dimensions 8D+ when convex hull computations are not feasible, other convex polytopes can be used to capture the uncertainty and although overbounding, the uncertainty structure is feasible to compute and far less conservative than the uncertainty of Fang and Chang, 1993.

The independent variation in A and B from de Oliveira et al., 1999 is a drawback which makes the problem formulation more conservative than required. Additionally, the number of equations can quickly

become large from a relatively small set of (A, B) matrices. For example, $N = 50$ vertices defining $A(\alpha)$ and $M = 60$ vertices defining $B(\beta)$ leads to 3000 LMI's.

The problem formulation provided in Peaucelle et al., 2000 extends Oliveira's result to paired (A, B) matrices, with a polytope defined by (27) describing the coupled uncertainty. This provides an excellent representation of the plant uncertainty, essentially exactly as identified, while also reducing the number of LMI's considerably. A tradeoff is computing the convex hull over both (A, B) could become infeasible with large numbers of uncertain parameters 8+. Thus, both methods provide a solid approach to the problem; the paired uncertainty description is preferable, but when convex hull computations become infeasible, allowing independent variation in A, B reduces the problem complexity, enabling the LMI approach to be applied to these more complex systems while maintaining a precise representation of the individual uncertainty regions $\mathscr{A}$ and $\mathscr{B}$.

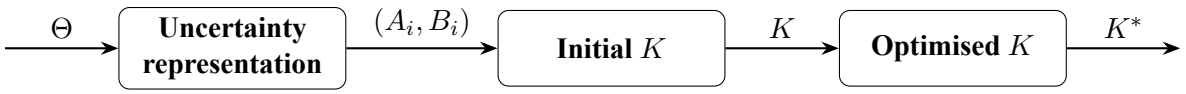
Worth noting is that both Oliveira and Peaucelle's LMI formulation provide only a *sufficient* condition for a simultaneously stabilising K, though as discussed in the paper, it is assumed not-too-conservative, particularly in relation to typical approaches to quadratic stability which require a common Lyapunov matrix P instead of a parametric $P(\alpha, \beta)$. Though a precise condition for a suitable K is desirable, simultaneous stabilisation is an NP-hard problem as discussed in the appendix, and is not directly tractable.

3.2 LMI-based solution

Briefly, the approach to the min-max problem is a) first find a stabilising K using the LMI-based solution, then b) optimise the LQR cost using a nonlinear optimiser.

The majority of computation time is spent on finding a simultaneously stabilising K. Finding such a K is most important theoretically, since even with stable open-loop plants, the K that is optimal for one plant may not be stabilising across the entire set. Furthermore, finding such a K is required practically, since the optimiser needs to be initialised with a valid (simultaneously stabilising) K. Also worth noting is that with this approach, the LQR optimisation can only return a locally optimal value, in the vicinity of the initial stabilising K - this is considered an acceptable tradeoff, given the NP-hard complexity of finding the global optimum.

The overall flowchart describing the controller design is The following flowchart describes the overall controller design:



Uncertainty representation. This stage involves computing the polytope for $\mathscr{A}\mathscr{B}$, which defines a set of n plants at the vertices of the uncertainty region.

Given the set of (A, B) matrices, we first compute the polytope $\mathscr{A}\mathscr{B}$ using a convex hull algorithm Qhull. More precisely, the uncertain entries across (A, B) are identified, vectorised, and the hull is computed across this vector space, then mapped back to matrix space.

The set of n vertex plants then used to form the n constraints of the LMI problem. Considering only the vertex plants significantly reduces the number of plants needing considered during controller synthesis, while retaining an exact characterisation of the convex region.

Initial K. With the reduced set of n plants, the LMI problem is solved according to

$$\begin{bmatrix} P_i & A_iG + B_iL \\ G'A_i' + L'B_i' & G + G' - P_i \end{bmatrix} > 0 \qquad \forall\, i = 1, \ldots, n$$

An SDP solver (Mosek) is used to solve this, yielding a simultaneously stabilising controller $K = LG^{-1}$ if the problem is feasible, or an infeasibility certificate otherwise.

Optimised K. With an initial feasible value for K, we can now optimise this gain to yield the final optimised controller K^*. For this we use scipy's BFGS optimiser. This is a *gradient-based* iterative method for unconstrained nonlinear problems, with efficient runtime $\mathcal{O}(n^2)$ due to gradient-based Hessian estimation. Though LQR cost is non-convex in K, it is known to be suitable for gradient-based algorithms due to satisfying the Polyak Lojasiewicz (PL) condition. While the robust performance formulation (21) differs from single-plant LQR cost optimisation, it is assumed that *locally* the min-max LQR cost can be considered similarly.

During the optimisation process, we repeatedly need to find the *worst-case* J for a given K. To evaluate this, first we check the eigenvalues $A - BK$ are within the unit circle, and break as soon as an unstable closed-loop plant is found - in this case, we return $J = \infty$, and thus the unconstrained optimiser becomes forced to find a local optimum. After checking stability, we then calculate J directly for each plant by solving a discrete Lyapunov equation (dlyap), and return the worst-case J. This procedure is exhaustive but works well for the reasonably small set of vertex plants, $n \approx 10 - 100$. Checking stability first via eigenvalues ensures we avoid unnecessary computation of discrete Lyapunov equations, which are more expensive calculations.

4 Methods: Algorithm Design

4.1 SPS Search Algorithm

Generating SPS confidence regions is practically achieved by sampling a search space, which is a subspace of $\mathbb{R}^d$, for d unknown parameters. The sampled points are then tested using the SPS indicator algorithm to determine whether points lie inside or outside the confidence region.

Initially, confidence regions were generated via a naive grid search. Each dimension was uniformly sampled with n points between appropriate minimum and maximum bounds. An n^d mesh grid was then constructed, and each coordinate in the mesh grid was tested using the SPS indicator function, generating the corresponding confidence region output.

From initial benchmark results, it became clear that confidence region generation was the largest bottleneck in the overall system architecture. Even relatively coarse grids resulted in millions of SPS indicator evaluations and required several minutes to generate a single confidence region. Despite efforts to optimise the computation of the SPS indicator itself, this approach quickly became impractical for systems with more than two unknown parameters, particularly in real-time applications.

For this reason, several alternative algorithms were proposed to efficiently determine the confidence region. The following subsections detail each of the candidate algorithms and their associated shortcomings. A detailed evaluation of the performance of these candidate algorithms is presented in Section 6.2.

4.1.1 Candidate Algorithms

Random Search with KNN Interpolation
K-Nearest Neighbours (KNN) is a simple, non-parametric classification algorithm which, as the name

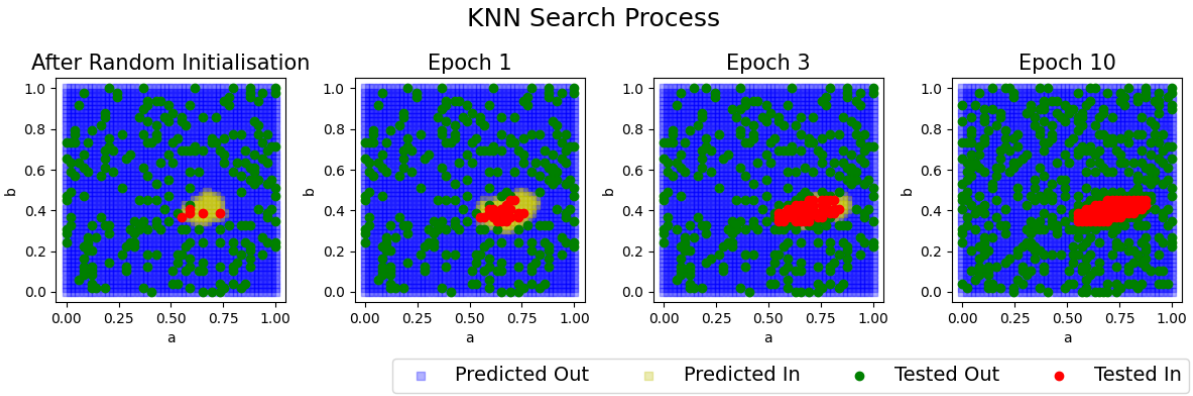


Figure 4: KNN search process, beginning with random initialisation and biasing subsequent samples toward points predicted to lie inside the confidence region (shown in yellow).

suggests, classifies a test datapoint according to the labels of its k nearest neighbours. Larger values of k result in smoother classification boundaries. While KNN can become computationally expensive for large datasets, this cost can be mitigated by employing tree-based search structures such as KD-trees or ball trees, which allow for faster nearest-neighbour queries (Bishop, 2006, p. 127).

Although KNN is limited in the complexity of the distributions it can represent, when combined with Euclidean distance it is well suited to approximating Gaussian-like confidence regions, which often characterise SPS outputs in practice. The algorithm begins with the same mesh grid used in the naive grid search. A subset of this grid is randomly sampled to initialise the search, and KNN is then used to predict whether the remaining points lie inside or outside the confidence region. Subsequent iterations proceed by testing N points using a stochastic sampling rule. At each iteration, a Bernoulli random variable $X \sim \text{Bernoulli}(p)$ is drawn. If $X = 1$, a point is sampled at random from the set of coordinates predicted to be inside the confidence region; if $X = 0$, a point predicted to lie outside is selected instead. After testing, the KNN model is retrained on the updated set of labelled points, and the procedure repeats until a fixed proportion of the search space has been evaluated, as determined by a user-defined parameter.

This adaptive strategy focuses the search on regions with a high likelihood of being inside the confidence region, improving sampling efficiency compared to uniform approaches. An illustration of this process is shown in Figure 4. However, this approach introduces additional computational overhead. Since KNN is a lazy learner with no internal model to update incrementally, the model must be re-trained from scratch at every iteration using the full, increasingly large, set of labelled points. While this cost may be negligible in low-dimensional or small-scale problems, it becomes non-trivial as the number of iterations increases, particularly in higher dimensions, where more points are required for adequate coverage.

Furthermore, this method is fundamentally limited by the curse of dimensionality. As the dimensionality d increases, datapoints become increasingly sparse, and Euclidean distance loses its discriminative power. The volume of the space concentrates near the boundary of the distribution (Domingos, 2012, p. 82), reducing the reliability of neighbourhood-based predictions. In such settings, the KNN model may fail to meaningfully represent the confidence region, leading to inefficient or misleading sampling behaviour.

Radial Search

To address the limitations of the KNN-based approach - particularly its inefficiency in high-dimensional spaces and the overhead of repeated model retraining - a more targeted and informed method was proposed: *Radial Search*. Rather than interpolating across the full space, this algorithm focuses on efficiently identifying the boundary of the confidence region using prior estimates.

The search is initialised by testing a point likely to lie within the confidence region. For open-loop FIR

systems, this is typically the PEM estimate; in closed-loop settings, a likelihood map or open-loop region estimate is used. Once confirmed to be inside, the point becomes the centre of the Radial Search.

From this centre, N unit vectors are generated uniformly in d-dimensional space, where $N \gg d$ to ensure adequate coverage. A binary search is then conducted along each direction to locate the boundary. To avoid redundant computations, scaled versions of boundary vectors are reused as starting points in subsequent iterations. The resulting boundary points are used to construct a convex hull or, optionally, a minimum volume enclosing ellipsoid (MVEE) to approximate the confidence region. An example output is shown in Figure 5.

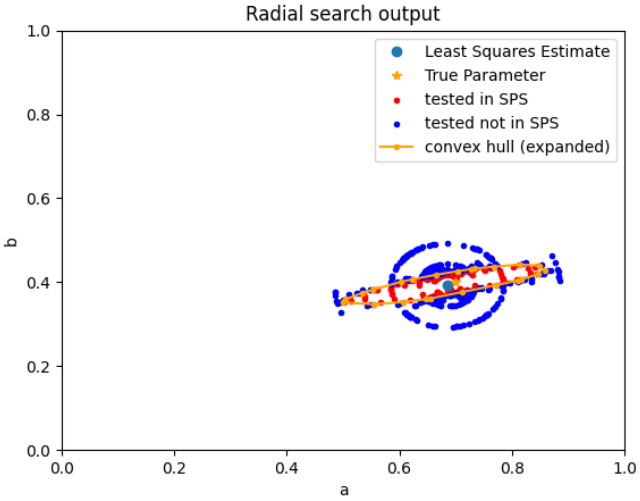


Figure 5: Example output of the Radial Search with two unknown parameters.

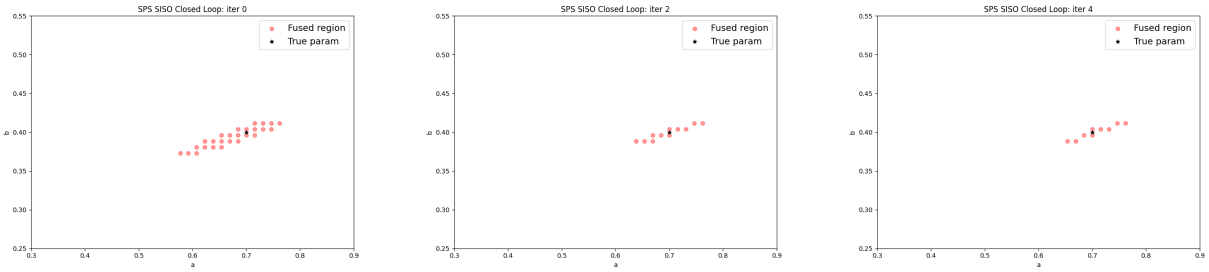
Compared to KNN, Radial Search is significantly more efficient in higher dimensions as it avoids dense sampling and retraining, focusing only on informative directions. This directional strategy scales more gracefully as d increases, unlike KNN which suffers from the curse of dimensionality and increasingly sparse neighbourhoods.

Nonetheless, the method assumes a connected and approximately convex region. It may miss non-convex or disjoint regions, which are not considered in this work, and is sensitive to the quality of the initialisation. Additionally, insufficient angular resolution can lead to poor boundary approximation but can be tuned offline for optimised results. Despite these limitations, Radial Search offers a scalable and computationally efficient approach to estimating confidence regions in high-dimensional spaces.

4.2 Recursive SPS via Log-Likelihood Fusion

The SPS algorithm produces finite-sample, exact confidence regions for unknown system parameters based on batches of input-output data. When applied sequentially, each batch yields a new confidence region; however, treating each in isolation leads to loss of accumulated knowledge and reduced statistical efficiency, as region size remains fixed with constant batch size. Incorporating information from prior SPS runs is desirable but non-trivial. SPS regions are defined implicitly via indicator functions, and storing high-dimensional sets online is computationally expensive. To address this, boundary points are efficiently computed via Radial Search (Section 4.1.1) and used to form compact convex hull approximations for downstream tasks such as set-membership testing or control synthesis.

While a fully recursive SPS algorithm is infeasible due to the overhead of maintaining perturbation structure over time (see Section 2 and Appendix D), a Bayesian reinterpretation enables efficient fusion of past



(a) Iteration 0 (b) Iteration 2 (c) Iteration 4

Figure 6: Evolution of confidence regions at successive data publishing steps to the SPS: after iteration 0, iteration 2, and iteration 4.

and present information. Specifically, we treat the confidence region as a surrogate likelihood function: for each SPS run, any point θ inside the region is assigned a likelihood of $1 - q/m$, and any point outside is assigned q/m. Thus, each region yields a piecewise-constant pseudo-likelihood over Θ.

Importantly, directly working with probability densities in high-dimensional spaces often results in extremely flat distributions, leading to numerical underflow and poor discrimination between plausible and implausible parameter values. To address this, we adopt a log-domain formulation for recursive updates. Treating the SPS indicator (Alg. 1, 2) as a pseudo-likelihood, we define $\mathcal{I}_k(\theta)$ to be `TRUE` if θ lies within the SPS confidence region at iteration k, and `FALSE` otherwise. The cumulative log-likelihood is then updated as

$$\log L_n(\theta) = \log L_{n-1}(\theta) + \log \pi_n(\theta),$$

where

$$\log \pi_n(\theta) = \begin{cases} \log(1 - q/m), & \text{if } \mathcal{I}_n(\theta) = \text{TRUE}, \\ \log(q/m), & \text{otherwise.} \end{cases}$$

This formulation avoids the numerical instability associated with repeated multiplications of small probabilities and enables robust accumulation of evidence across SPS runs, even in high-dimensional spaces or over long time horizons.

Constructing SPS-Like Confidence Sets. Since controller synthesis often requires a *set* of admissible parameters rather than a probability distribution, we convert the accumulated log-likelihood to a confidence region by selecting the most likely points whose total normalised probability mass approximates $1 - q/m$. This avoids arbitrary thresholds (e.g., p_{min}) and yields interpretable sets with tunable coverage.

Forgetting Mechanism. To retain adaptivity in non-stationary environments, a forgetting factor $\lambda \in [0, 1]$ is introduced to prevent older information from overwhelming new observations. Let $\bar{L}_{n-1} = \text{mean}(\log L_{n-1}(\Theta))$ denote the average log-likelihood over the grid. We then flatten the prior as

$$\tilde{L}_{n-1}(\theta) = (1 - \lambda) \log L_{n-1}(\theta) + \lambda \cdot \bar{L}_{n-1},$$

which is used in the recursive update:

$$\log L_n(\theta) = \tilde{L}_{n-1}(\theta) + \log \pi_n(\theta).$$

Setting $\lambda = 0$ recovers full-memory fusion, favouring convergence but reducing adaptivity. At the other extreme, $\lambda = 1$ corresponds to standard SPS without fusion, discarding all past information. In practice, a choice of $\lambda \in (0, 1)$ offers a tunable trade-off between statistical efficiency and responsiveness to parameter shifts. In the examples presented in this work, we fix $\lambda = 0$ to prioritise fast convergence, though the implementation supports general λ for broader applicability.

This recursive framework allows efficient, compact, and adaptive tracking of SPS regions over time, enabling long-horizon inference and control in evolving environments.

5 Methods: Software Design

5.1 System Architecture Overview

The proposed architecture is designed to manage the interaction between fast system dynamics and slower model identification and control updates. Unlike traditional adaptive control loops, where identification, controller updates, and actuation occur on the same time scale, this architecture explicitly decouples these processes. Control actions are executed in real time, while model identification and controller refinement operate asynchronously, potentially across slower or irregular intervals. This separation improves robustness, allows for more complex optimisation strategies, and supports scalability across multiple devices or plants.

A key advantage of this architecture is its flexibility. To maintain real-time responsiveness on constrained hardware, computationally intensive tasks, such as identification and synthesis, are offloaded asynchronously to remote or parallel processes. The modular design ensures consistent implementation across embedded and distributed environments, and accommodates asynchronous operation of control and identification tasks on different time scales. Figure 2 shows a simplified adaptive loop: a closed-loop plant for reference tracking, an online SPS-based identification module, and a feedback path for controller updates. A full architectural diagram is available in Appendix N.

5.2 Design Patterns and Architectural Principles

To support the system's modular and asynchronous operation, several well-established software design patterns and engineering principles are applied. These patterns ensure that different parts of the system, such as real-time control, simulation, logging, and identification, can be developed, tested, and executed independently, without interfering with one another. To support the system's modular and asynchronous operation, several well-established software design patterns and engineering principles are applied. These patterns ensure that components, such as real-time control, simulation, logging, and identification, can be developed, tested, and executed independently without interference.

- **Factory Pattern**: The system selects at runtime whether to use a real-time backend, simulation model, or a user-specified search algorithm. This runtime flexibility enables seamless switching between environments for development, testing, or deployment (cf. Gamma et al., 1995, p. 107).
- **Singleton Pattern**: Shared resources such as the database and Redis broker exist as globally accessible, single instances. This ensures consistency across modules and avoids conflicts in resource usage (cf. Gamma et al., 1995, p. 127).
- **Observer Pattern**: Modules like dashboards and loggers react automatically to parameter updates or system changes. This event-driven architecture supports real-time diagnostics and removes the need for manual polling (cf. Gamma et al., 1995, p. 239).
- **Adapter Pattern**: The system interacts with hardware platforms, simulation engines, or external tools via a standard interface. This allows core control and identification logic to remain unchanged across different environments, and makes it easy to swap in new algorithms like an updated SPS module and different search strategies. (cf. Gamma et al., 1995, p. 139).
- **Facade Pattern**: Users and external tools interact with the system through a unified API instead of accessing internal components directly. This abstraction simplifies integration, testing, and automation by hiding internal complexity behind a consistent interface (cf. Gamma et al., 1995, p. 185). For example, the SPS module exposes a single indicator function that abstracts away the

underlying details, such as batch handling, perturbation generation, and radial search, providing a consistent interface across both open-loop and closed-loop identification settings.

- **Publish-Subscribe (Pub/Sub)**: Modules communicate asynchronously through Redis without blocking one another. For example, the identification module can send model updates to the controller without pausing the control loop. Redis supports distributed operation across processes or devices, enabling scalable deployment (Redis Ltd., 2024).

5.3 Modularity, Testability, and Deployment Flexibility

The architecture emphasises modularity and scalability by applying **Inversion of Control**, where components receive essential services (e.g., logging, communication, configuration) externally rather than managing dependencies internally. This decoupling enables flexible reconfiguration and simplifies testing. Each major subsystem, such as the Simulation Engine, Real-Time Controller, SPS algorithm, hardware interface, or Monitoring unit, defines clear service interfaces, abstracting behaviour from implementation. This facilitates easy replacement or extension of components with minimal system disruption.

Dependency injection further enhances flexibility by supplying services like loggers or Redis clients at runtime. This enables components (e.g., SPS and simulation engine) to run in separate processes or machines while coordinating asynchronously via Redis. Similarly, the search algorithm receives the SPS indicator function as a callback, remaining agnostic to underlying logic and adaptable to different evaluation contexts. Modularity also improves **testability and maintainability**: components are unit-tested in isolation with mocks substituting real dependencies, leveraging Python's `unittest` framework. This reduces integration errors and supports continuous, confident development.

To ensure consistent behaviour across environments - local, remote, or embedded - all components are containerised with **Docker** (Merkel, 2014). By combining modular interfaces, asynchronous communication, and containerisation, the system achieves adaptability, maintainability, and scalable deployment.

5.3.1 Just-In-Time (JIT) Compilation for Numerical Optimisation

To support real-time execution and efficient numerical computation, performance-critical modules are optimised using **Numba**, a Just-In-Time (JIT) compiler for Python (**numba**). Numba translates selected Python functions into low-level machine code at runtime, significantly improving performance while preserving the clarity and flexibility of high-level Python.

This optimisation strategy is applied in a custom library for discrete-time transfer function operations. The library implements core arithmetic operations - **addition, subtraction, multiplication**, and **division** - on filters, and applies them to time-series data. It also supports operations over multi-dimensional arrays, including **element-wise manipulation** and **matrix inversion** of transfer function matrices. Such functionality is essential for multivariate system modelling, where computational efficiency is critical for simulation and control tasks.

Additionally, components of the **SPS** algorithm, particularly those suitable to static typing, such as gradient computation and rank estimation, are compiled using JIT (Just-In-Time) compilation to reduce execution latency. This optimisation enables the SPS indicator to run with microsecond-level latency in certain configurations, making it viable for time-critical deployment. Empirical benchmarks show that the JIT-compiled implementation significantly outperforms traditional symbolic computation tools in both Python and MATLAB. While symbolic methods provide abstraction, they introduce considerable overhead in large-scale or real-time contexts. In contrast, our Numba-based approach delivers substantial speedups, making it well-suited for embedded and resource-constrained systems. For brevity, performance benchmarks are omitted from this work.

6 Results and Analysis

This section first introduces the systems used for evaluation, followed by a performance comparison of the proposed search algorithms from Section 4. The best-performing method, Radial Search, is then integrated into real-time simulations to evaluate tracking performance and the effectiveness of the full adaptive framework. For each tested system, an SPS class instance was created with $q = 5$ and $m = 100$, i.e. $p = 0.95$, and batch size $N = 200$ samples.

6.1 Data generating systems

The complete system architecture is evaluated on three representative data-generating systems:

1. **First-Order ARMAX**
$$y(k) + a_1 y(k-1) = b_1 u(k-1) + e(k)$$
2. **Second-Order ARMAX**
$$y(k) + a_1 y(k-1) + a_2 y(k-2) = b_1 u(k-1) + b_2 u(k-2) + e(k)$$
3. **Water Tank Model**
$$\text{level}(k+1) = (1 - r \cdot \text{dt}) \cdot \text{level}(k) + b \cdot \text{dt} \cdot u_k, \quad \text{level}(k+1) \in [0, H]$$
where r is the leak rate from the tank bottom, b is the inflow gain, and H is the maximum tank height. The level is clipped to remain within physical bounds.

Additional systems were implemented and tested during development, but not fully integrated into the final framework. These are listed in Appendix A for reference.

System	State Dimension	Inputs/Outputs	No. of Parameters (Estimated)
First-Order ARMAX	1	1 / 1	2
Second-Order ARMAX	2	1 / 1	4
Water Tank Model	1	1 / 1	2

Table 1: Summary of benchmark systems, with their structure and number of parameters estimated via SPS.

6.2 Search algorithms performance

6.2.1 Evaluation Metrics

To select the most suitable search algorithm, we compared each method's confidence regions against a ground-truth baseline generated via dense grid search. Various hyperparameter settings were tested to identify the best balance between region accuracy and computational cost.

All algorithms were evaluated on identical fixed data and SPS perturbations to ensure fair comparison. The baseline grid spanned the range $[\theta_i^\star - 0.5, \theta_i^\star + 0.5]$ with 50 points per dimension. For each hyperparameter configuration, 50 confidence regions were generated and assessed based on the number of SPS calls, runtime, Intersection over Union (IoU), and Bhattacharyya distance. Due to computational limitations, the KNN algorithm was tuned on a 20^4 grid and validated on a finer 40^4 grid.

Bhattacharyya Distance
Bhattacharyya distance quantifies the similarity between two Gaussian distributions with means $\vec{M}_i$ and covariance matrices Σ_i:

$$\mu(1/2) = \frac{1}{8}(\vec{M}_2 - \vec{M}_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\vec{M}_2 - \vec{M}_1) + \frac{1}{2}\log \frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}.$$

The first term becomes zero if the means coincide, and the second term vanishes if the covariances are equal. By approximating confidence regions as ellipsoidal Gaussians, this metric captures differences in both their centers and shapes. Lower values indicate closer agreement with the baseline.

Intersection over Union (IoU)
IoU, or the Jaccard Similarity Index, is widely used in computer vision for evaluating overlap between predicted and true bounding boxes (Szeliski, 2022, p. 303). For ground truth region B_{true} and prediction B_{pred}, IoU is defined as

$$\text{IoU} = \frac{B_{\text{true}} \cap B_{\text{pred}}}{B_{\text{true}} \cup B_{\text{pred}}}.$$

In our context, IoU measures the similarity between confidence regions. For first-order systems with two parameters, we compute IoU using the areas of convex hull intersections. For higher-dimensional systems, we approximate volumetric overlap by random sampling points in the search space and calculating the ratio of points lying within both regions relative to their union. Detailed benchmark results are provided in Appendix M and summary of the results are presented here.

6.2.2 Results

Radial Search: The time complexity for Radial Search was found to be approximately linear with respect to the number of unit vectors tested.
For first order systems with two unknown parameters, the performance is maximised using 250 unit vectors. This set of hyperparameters resulted in an IoU score of 0.807 (± 0.007), Bhattacharyya Distance of 0.135 (± 0.005), while generating a confidence region within 302ms.
For second order systems with four unknown parameters, the radial search performance was maximised using 1500 unit vectors. These settings resulted in an IoU score of 0.585 (± 0.023), Bhattacharyya Distance of 0.707 (± 0.012) in a closed-loop setting, with a mean time of 2494ms. Despite a comparably low IoU score, the generated confidence regions were sufficient for reference tracking in practice. Furthermore, the fast generation of a confidence region makes this approach feasible for real-time applications. These benchmarks were conducted in a Jupyter notebook environment, which does not benefit from

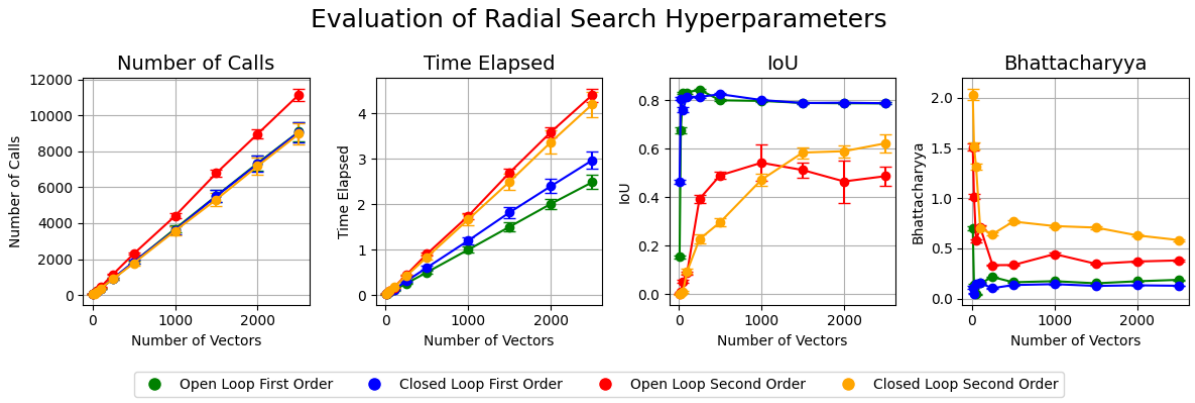


Figure 7: Radial Search evaluation over number of radial unit vectors.

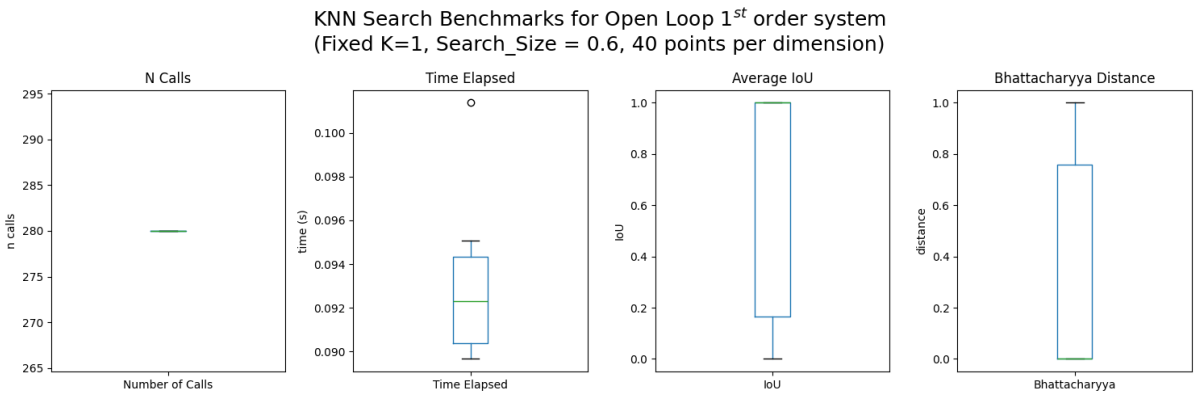


Figure 8: First Order Open Loop Benchmarks for KNN Search with optimal hyperparameters

caching and has limited memory and CPU resources. In real-time execution, both first- and second-order systems performed significantly faster, with complete searches typically completing within 500 ms.

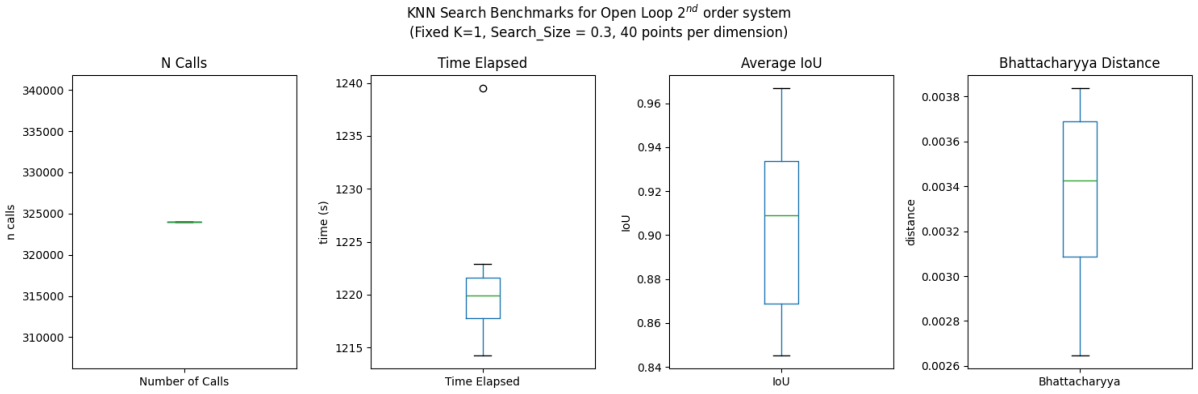


Figure 9: Second Order Open Loop Benchmarks for KNN Search with optimal hyperparameters

KNN Search: KNN Search was found to be infeasible for systems of order greater than one. For second order systems, the KNN Search failed to outperform the baseline grid search, taking in excess of 15 minutes to generate a confidence region using powerful hardware, despite testing only 30 percent of the search space. Full benchmarks for 2nd order systems were performed on a coarse grid and can be seen in Appendix M.

However, for first order systems, this search method performed well. After tuning hyperparameters, this search algorithm attained near-perfect IoU scores with a mean of 0.98, while generating the confidence region in under 100ms.

6.3 Evaluation of real-time adaptive control loop with SPS

All systems are executed in real time with a sampling interval of $d_t = 0.02$. Initially, each system operates in open-loop mode with a square-wave input to excite the system dynamics and capture an initial confidence region. This region is estimated using PEM, which serves as the initialisation for the radial search algorithm.

Subsequent iterations are performed in closed-loop mode, where the confidence regions from each iteration are merged using the fusion algorithm. System performance is evaluated over these iterations based on the updated merged confidence regions.

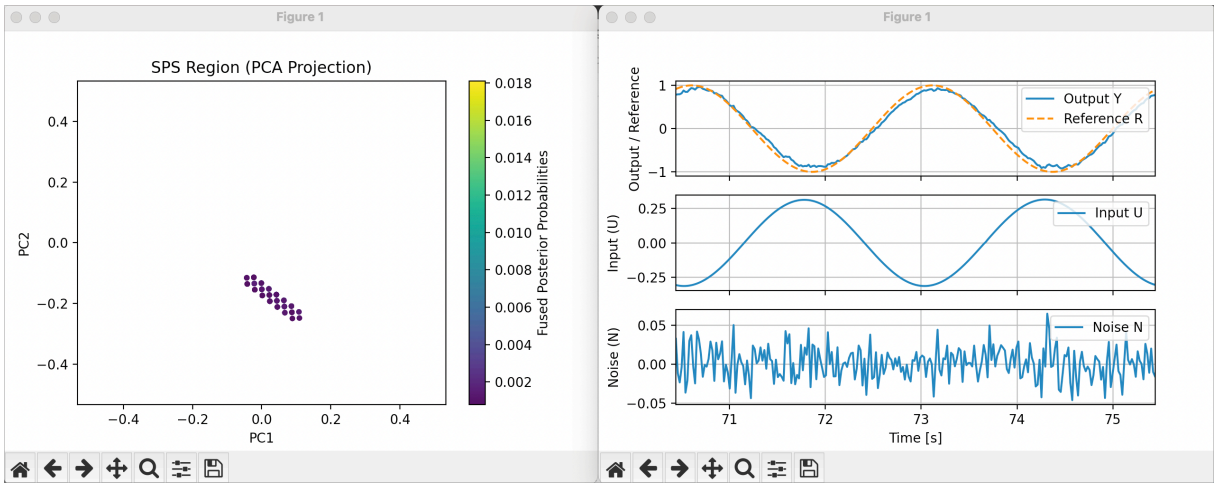


Figure 10: ARMAX second order tracking a sinusoidal reference in real-time.

6.3.1 Evaluation Metrics

Hull Volume
The volume of the convex hull is used to quantify the size of the confidence region and evaluate how it contracts after each fusion step. The convex hull is computed using the `scipy.spatial.ConvexHull` function from the SciPy library SciPy Community, 2025.

Mean Relative Absolute Error (MRAE)
MRAE quantifies the absolute tracking error relative to the reference signal's magnitude:

$$\text{MRAE} = \sum_{k=1}^{N} \frac{|y(k) - r(k)|}{|r(k)|}$$

This metric is sensitive to steady-state errors, lag, and slow convergence. High MRAE values indicate persistent deviation from the reference, even if the general shape is followed.

Root Mean Square Error (RMSE)
RMSE captures both variance and bias in tracking by penalising larger deviations more heavily:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y(k) - r(k))^2}$$

Like MRAE, RMSE is sensitive to lag and transient errors but more strongly penalises occasional large deviations.

Coefficient of Determination (R^2)
The R^2 score measures how well the system output explains the variance of the reference:

$$R^2 = 1 - \frac{\sum_{k=1}^{N} (y(k) - r(k))^2}{\sum_{k=1}^{N} (r(k) - \bar{r})^2}$$

where $\bar{r}$ is the mean of the reference. An R^2 value close to 1 indicates good overall fit, though it may overlook steady-state bias or slow responses in systems with low reference variance (e.g., step signals).

6.3.2 Results

Figure 11 illustrates that the convex hull volume consistently decreases with each iteration, indicating that the confidence region contracts over time. This contraction corresponds to improved trasient performance, as evidenced by the increasing R^2 values and reduced relative errors noted by MRAE and RMSE. Notably, the R^2 metric converges rapidly for the first-order systems, including the water tank model, suggesting that both plant identification and controller tuning are effectively achieved early in the process.

For all three systems, both MRAE and RMSE metrics show a clear decline over time. This improvement is likely attributable to the longer time scales and periods of constant reference, which enable clearer distinction between transient and steady-state errors. These relative errors quickly converge to an optimal non-zero value, these errors are never going to converge to zero due to presence of the noise. The second order system shows the most amount of improvement, drastic improvement in MRAE highlighting better control effort and less ringing.

Furthermore, Figure 10 demonstrates that the controller tracks the second-order ARMAX system with minimal phase lag, highlighting the framework's capability for real-time system identification and enhanced control performance. Overall, these results confirm the efficacy of the proposed approach in rapidly refining model confidence and improving closed-loop behaviour.

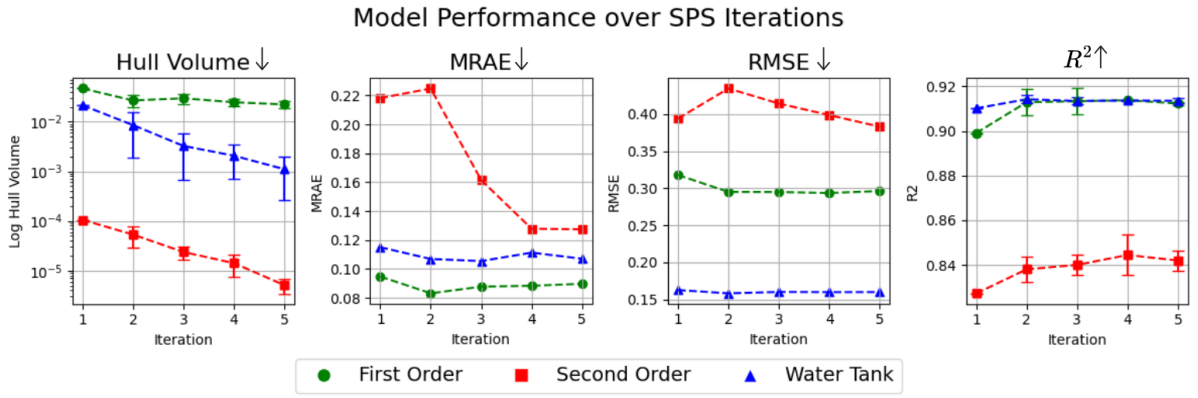


Figure 11: Model performance of tracking square wave reference over SPS iterations. Note that hull volume uses logrithmic scale.

7 Discussion

7.1 SPS with real world systems

While many real-world systems exhibit nonlinear behaviour, they are often well approximated by linear time-invariant (LTI) models within a nominal operating range. For instance, the nonlinear outflow dynamics of a water tank can be linearised to a first-order system; a damped pendulum is effectively represented by a second-order system; and a rotary pendulum such as the Quanser QUBE is well modelled as a third-order system. These dynamics can be captured in ARX form, and an ARMAX model provides a more accurate model by accounting for correlated disturbances.

Our work showed that both computational time and memory usage increase with the dimensionality of the parameter space. This can limit the practical use of SPS for fast dynamical systems requiring rapid parameter estimation. However, SPS remains well suited for slower systems, such as chemical plants or dams, where the estimation timescales are more relaxed. Furthermore, in many real-time applications,

not all model parameters need to be estimated online as many correspond to fixed physical constants. By applying SPS selectively to only those parameters that exhibit variability or are challenging to estimate offline, the search space can be effectively reduced. This targeted approach enables SPS to achieve real-time or near-real-time performance in practical settings.

7.2 Controller Design

Alternative approaches to uncertainty representation: Computing the convex hull $\mathscr{A}\mathscr{B}$ captures the SPS-identified confidence region precisely, with conservativeness introduced only by the convexification of the uncertainty region. Crucially, we get a reduced set of n plants, which define an LMI problem that can be efficiently solved to find the initial simultaneously stabilising K. However, the major drawback is computational scalability; the cost of computing the convex hull grows rapidly with dimension. Some examples of when convex hull construction becomes prohibitively slow can be found in the appendix - as a rule of thumb, it is generally intractable to compute the convex hull for systems with four or more states, taking 20s+. In these cases, two alternative approaches to uncertainty region representation are provided, based on an ellipsoid-like polytope ("LowResMVEE") or computing independent convex hulls $\mathscr{A}$ and $\mathscr{B}$. Further details are in the appendix H.1. Finally, the option to use all plants directly is also provided.

The four approaches to representing the uncertainty region are visually demonstrated in Figure 12 for clarity.

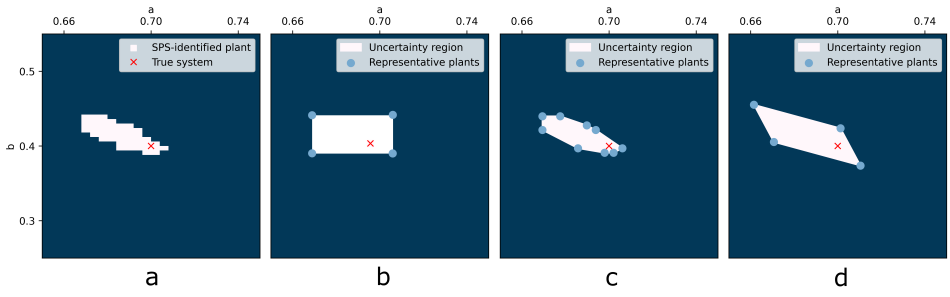


Figure 12: Four available approaches for representing the uncertainty region. In (a), all plants are used directly with no simplification. In (b), the Peaucelle convex hull $\mathscr{A}\mathscr{B}$ is used. In (c), the Oliveira convex hull approach with independent $\mathscr{A}$, $\mathscr{B}$ is used. Finally, (d) uses the LowResMVEE approach with $m_d = 3$ yielding a cross-polytope.

Riccati heuristic: While solving for a stabilising K via LMI's is the default approach, another approach via a *Riccati heuristic* is also available, the details of which are deferred to the appendix. Briefly, this option tends to be less computationally demanding and returns not only a stabilising K but an approximately *min-max optimal* K. However, this heuristic comes with no guarantees, and may fail to find even a stabilising K, whereas the LMI approach would succeed. Nevertheless, the heuristic is often applicable and provides significant speed up to the controller synthesis, thus it is included as an option for the controller synthesis, to be chosen in scenarios where a designer deems it suitable.

Evaluating J for a given (A, B, K): With LTI state-space dynamics and full-state feedback, the cost function J can be expressed as

$$J = \sum_{k=0}^{\infty} x_0^T (A - BK)^{kT} (Q + K^T R K)(A - BK)^k x_0 \tag{29}$$

This corresponds to evaluating $x_k^T Q x_k + u_k^T R u_k$ over the closed-loop response to an initial condition x_0. The infinite sum can be eliminated to yield a more explicit expression

$$J = x_0^T S x_0$$

where $S(A, B, K, Q, R)$ is the solution to the discrete Lyapunov equation

$$(A - BK)^T S(A - BK) - S + Q + K^T R K = 0$$

or more concisely, let $\tilde{Q} = Q + K^T R K$ and $\Phi = A - BK$ so that the discrete Lyapunov equation becomes

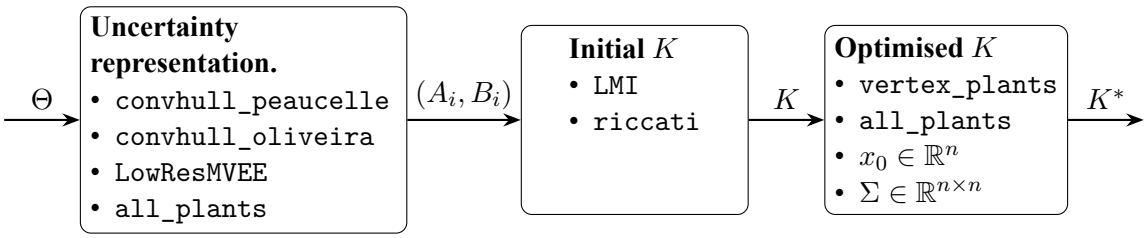
$$\Phi^T S \Phi - S + \tilde{Q} = 0$$

We now have an explicit value for $J(A, B, K, Q, R)$, with no need to truncate the infinite sum. This is how the cost $J = x_0^T S x_0$ is computed during optimisation, given controller K and a plant (A, B) .

Cost function optimisation : As presented, the cost of a given plant (A, B) with controller K depends on its closed-loop response to initial condition x_0. Thus, minimising $J = x_0^T S x_0$ yields a controller K which has the optimal response to initial condition x_0. While the initial condition x_0 may be known at the initial time, and thus optimising using this x_0 is a suitable strategy, it is generally no longer the same by the time the min-max optimisation problem is solved and the controller is implemented. Thus, it is preferable to use a typical assumption in this context, to assume x_0 is a zero-mean random vector with identity covariance, such that the cost becomes $J = \mathbb{E}[J] = \text{trace}(S)$, or, assuming x_0 is zero mean and the covariance Σ is known, $J = \mathbb{E}[J] = \text{trace}(S\Sigma)$. The expectation-based formulations removes explicit dependence on x_0, ensuring a controller that is not biased toward any particular initial condition.

Finally, it was mentioned that the cost for a given K is evaluated across all vertex (A, B), instead of the SPS-identified (A, B). Using these vertex (A, B) provides an accurate representation of the plant uncertainty, but simplifies it significantly so that optimisation is efficient. When SPS identifies a large number of plants $n \approx 1000+$, this step is crucial. However, it introduces some conservatism. Thus, an option is provided to instead use all SPS-identified (A, B) directly during optimisation. Though this is more costly, in practice it's often preferable to spend an extra second or two and optimise w.r.t. the actual plants, in situations where this is feasible. Hence it is provided as an option.

Flowchart. In light of the previous discussion, additional methods are provided for controller synthesis. Briefly, these allow trade offs between efficiency, accuracy, and robustness, and enable the solver to approach more complex systems which would not be tractable with the default solve method (`convhull_peaucelle` + `LMI` + `vertex_plants`).



Validity of LQR control: With the LQR cost function, min-max optimisation across a set of possible plants is well-defined, and grounded in a meaningful notion of a controller's robust performance. For example, if the cost matrices Q and R are selected according to Bryson's rule, then the min-max optimal controller K minimises the worst-case state deviations and input efforts relative to specified limits. Thus, in general, the min-max optimisation has a clear interpretation, and provides a meaningful way of evaluating a controller's robust performance.

Although we focus on designing simple state feedback regulators, the synthesis procedure developed can be applied to a broader range of control schemes which make use of full state feedback K. For example, controllers designed with integral action or for disturbance rejection can be handled by augmenting the state space matrices A, B as appropriate. Design of observer gains can also be done with the same optimisation framework via duality.

Finally, LQR offers strong stability properties, with a guaranteed phase margin of at least 60 degrees. This inherent robustness is valuable in the adaptive control setting, where system uncertainty is unavoidable. In the case where full state measurement is not available, LQG can be used instead, though this comes at the cost of guaranteed stability margins, and may reduce the effectiveness of SPS due to filtering out informative state noise.

Future improvements: A key direction for future work is improving the optimisation method used for the min-max LQR cost. More theoretically grounded approaches should yield greater efficiency. Additionally, investigating the convexity of the cost function could help identify conditions under which local minima are also global, or provide guarantees on convergence.

7.3 Search Algorithm

Limitations of Radial Search on other Model Classes Our search algorithms, especially the Radial Search algorithm are tailored to the specific model classes for which we were designing our system, relying on assumptions of a convex confidence region to efficiently find neighbouring and boundary points of a *single* confidence region. Properties of FIR systems were also utilised to consistently find a starting point inside the confidence region using PEM. For nonlinear systems, convexity is not guaranteed and for non-FIR systems, the confidence region is not guaranteed to be star-convex.

Expanding real-time SPS-based adapative control to systems outside of this model class would require additional effort into developing efficient search strategies without relying on these assumptions.

Future improvements of KNN: While generating confidence regions using KNN for 2nd Order took too long to be useful in a real-time scenario, its high IoU and low Bhattacharyya distances mean that it could outperform other search methods if its speed is addressed. Furthermore, introducing an alternate stopping criteria could prevent many unnecessary iterations of the KNN search. For example, the search could conclude upon exhausting the predicted-in points after calculating a confidence region of a minimum size.

As KNN search is not restricted to a single convex confidence region, a more performant algorithm could be useful in expanding real-time SPS identification to other model classes. KNN Search could also be combined with the Radial Search algorithm to densely test a confidence region proposed by the Radial Search. In this way, the Radial Search's number of vectors could be minimised while still being performant.

Limitations of the IoU metric: While this heuristic is easy to implement and intuitive to analyse, it does not necessarily create a fully appropriate measurement for our use case. In particular, IoU suffers from treating the case where $B_{true} \subset B_{pred}$ equivalently to the case where $B_{pred} \subset B_{true}$. That is to say, a predicted confidence region which includes points outside of the ground truth region may result in the same score as a 'more conservative' subset of the ground truth. Developing additional penalties for introducing points outside the baseline confidence region, which may introduce difficult or impossible plants into the confidence region, may be worthwhile to distinguish between these cases.

The IoU metric also depends heavily on the accuracy of the baseline confidence region. While for low-order systems, it is possible to generate a fine enough meshgrid for accurate comparison, extending this to higher dimensions requires a coarser baseline search which may hamper the comparison. Future im-

plementations of the benchmarking harness could perform a one-time ultra-fine meshgrid search which could be tested against for all algorithms before validating them against more varied models.

7.4 Fusion

A similar fusion approach via *normalised SPS ranks* was considered, i.e. $\pi(\theta) \doteq 1 - \mathcal{R}(\theta)/m$. The SPS ranks provide information which is otherwise discarded after applying the rank threshold, and appropriately normalised ranks can be interpreted as a pseudo-probability distribution about the LSE. By making use of this granular distribution in the update step, as opposed to the binary-valued distribution used in the established approach, convergence about the true parameter can be improved and information about the system provided by SPS can be fully utilised. However, the theoretical justification for this approach is unclear. Additionally, though fine with grid search, with radial search a heuristic would be required to infer ranks across the grid region based on the established boundary; this heuristic would introduce additional complexity and decrease robustness. Thus, this approach was ultimately discarded.

There is room for improvement in the efficiency of the current fusion algorithm. Currently, likelihoods are stored against each coordinate on a grid inside certain bounds. Dense grids or higher-dimensional models create inefficiencies in both memory and computation as many coordinates with very low likelihoods are stored in memory. This approach could be improved by using the convex hull generated by the Search Algorithm and storing only points in the union of the previous and current confidence sets before using the union-find algorithm to fuse them.

8 Conclusion and Recommendations

In summary, this project successfully developed and implemented a closed-loop robust adaptive control framework based on SPS, addressing both SISO and MIMO systems, with results presented for the SISO case. A high-performance Python library for efficient MIMO transfer function operations enabled a computationally tractable implementation of indirect SPS for closed-loop MIMO systems.

An LQR controller design algorithm was established that leverages the exact, non-asymptotic confidence regions provided by SPS to guarantee robust performance. To extend applicability to higher-dimensional settings, alternative methods such as the Oliveira hull and LowResMVEE approaches were introduced, providing conservative yet computationally feasible controller synthesis. Additionally, a Riccati-based heuristic was implemented to efficiently approximate the min-max LQR solution when plant uncertainty is moderate.

The radial search algorithm was developed and benchmarked as an effective method for constructing SPS confidence sets in higher-dimensional parameter spaces. This was further enhanced with a likelihood-based strategy to find search centers, improving computational efficiency. Recursive SPS region fusion was introduced through a Bayesian interpretation, incorporating a tunable parameter that balances adaptation and convergence, thereby ensuring the framework's flexibility across diverse scenarios. To overcome the computational challenges of storing and fusing high-dimensional SPS confidence sets, the LowResMVEE method was employed to achieve more compact uncertainty set representations.

A modular system architecture was implemented, supporting asynchronous interaction between system identification, controller synthesis, and fusion modules. The integrated system successfully demonstrated robust adaptive control across linear, nonlinear, SISO, and MIMO plants, with real-time performance demonstrated at Endeavour in OpenCV-based simulations. Ultimately, all project aims as outlined in (1.1) were realised, save for demonstrating real-time adapation capability. With further time and effort, real-time adaptation to time-varying models could be demonstrated. No changes are expected to be needed to achieve this, only a suitable simulation environment or deployment on a real-world prototype.

Future improvements. A number of directions can further develop our approach to adaptive closed-loop control.

From a computational perspective, much of our work may be implemented in a more efficient, lower level language such as C. Additionally, parallel computing could further improve performance.

Furthermore, better understanding of the min-max LQR cost landscape w.r.t. K could enable development of more efficient approaches to controller optimisation. Results are known for the smoothness and convexity of the min-max LQR cost landscape for a single plant, and deeper investigation of these properties in the min-max case could prove fruitful.

From a theoretical perspective, the underpinnings of Bayesian fusion with forgetting require deeper investigation and foundations, to ensure the rigorous SPS confidence guarantees are retained after fusion. Other recursive fusion approaches compatible with SPS's frequentist probability guarantees may be preferable. Robustness of the framework under nonlinear and linear time-varying (LTV) dynamics is another open question, and establishing performance and confidence guarantees under such conditions would significantly enhance general applicability. Finally, a theoretical basis for selecting or adapting the fusion parameter λ should be developed, ideally based on known or estimated bounds on the rate of parameter drift in practical systems.

A more effective reference tracking scheme should be implemented. Due to system complexity and time constraints, it was not possible to test the system with an integrator in the loop, thus reference tracking is effectively open loop based on reference scaling, and the controller can only effectively regulate the system dynamics/transients. The existing LQR synthesis approach remains compatible with integrator-augmented systems, provided SPS conditions are still met after augmentation. Similarly, feedforward terms can be introduced, enabling optimal tracking of arbitrary known reference signals - this can be achieved with output feedback and computed by iterating a Riccati difference equation.

To handle partial state measurement and reduce noise amplification, LQG-based control could be used. The required G_0 and H_0 formulations for indirect SPS are given in the appendix. In parallel, applicability to general ARMAX systems, where states are not in observable canonical form and $C \neq 1$, should be demonstrated, preferably on a physical prototype.

Regarding the radial search algorithm, theoretical analysis of the number of rays required for a reliable region estimate should be established, and an ellipsoidal instead of spherical basis may be used to improve performance. Likewise, optimal design of reference signals and controller switching strategies should be investigated; with more informative excitation, intersecting SPS confidence sets across iterations couold lead to more effective parameter identification.

Finally, deploying the system on a real-world prototype remains the most important step for future work, to validate and further refine the control scheme. The asynchronous architecture enables the computational burden to be offloaded from the system's microcontroller, for example by communication with a desktop PC or cloud-based service, thus such a practical online implementation is feasible and recommended for future work.

9 Acknowledgements

We would like to express our deepest gratitude to our supervisor, Erik Weyer, for his invaluable guidance and support throughout this project.

References

Abbasi-Yadkori, Y., & Szepesvári, C. (2011, September). Regret bounds for the adaptive control of linear quadratic systems. In S. M. Kakade & U. von Luxburg (Eds.), *Proceedings of the 24th annual conference on learning theory* (pp. 1–26, Vol. 19). PMLR. https://proceedings.mlr.press/v19/abbasi-yadkori11a.html

Abeille, M., & Lazaric, A. (2018, October). Improved regret bounds for thompson sampling in linear quadratic control problems. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (pp. 1–9, Vol. 80). PMLR. https://proceedings.mlr.press/v80/abeille18a.html

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Cao, R., Lu, Y., & He, Z. (2022). System identification method based on interpretable machine learning for unknown aircraft dynamics. *Aerospace Science and Technology*, *126*, 107593. https://doi.org/https://doi.org/10.1016/j.ast.2022.107593

Carè, A., Csáji, B. C., Campi, M. C., & Weyer, E. (2018). Finite-sample system identification: An overview and a new correlation method. *IEEE Control Systems Letters*, *2*(1), 61–66. https://doi.org/10.1109/LCSYS.2017.2720969

Chang, S., & Peng, T. (1972). Adaptive guaranteed cost control of systems with uncertain parameters. *IEEE Transactions on Automatic Control*, *17*(4), 474–483. https://doi.org/10.1109/TAC.1972.1100037

Csaji, B. C., Campi, M. C., & Weyer, E. (2015). Sign-Perturbed Sums: A New System Identification Approach for Constructing Exact Non-Asymptotic Confidence Regions in Linear Regression Models. *IEEE Transactions on Signal Processing*, *63*(1), 169–181. https://doi.org/10.1109/TSP.2014.2369000

Csáji, B. C., & Weyer, E. (2015). Closed-loop applicability of the sign-perturbed sums method. *2015 54th IEEE Conference on Decision and Control (CDC)*, 1441–1446. https://doi.org/10.1109/CDC.2015.7402413

de Oliveira, M. C., Bernussoub, J., & Geromela, J. C. (1999). A new discrete-time robust stability condition (. https://api.semanticscholar.org/CorpusID:55007593

Domingos, P. (2012). A few useful things to know about machine learning. *Commun. ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Fang, C.-H., & Chang, F.-R. (1993). Robust control analysis and design for discrete-time singular systems [12th Triennal Wold Congress of the International Federation of Automatic control. Volume 2 Robust Control, Design and Software, Sydney, Australia, 18-23 July]. *IFAC Proceedings Volumes*, *26*(2, Part 2), 83–88. https://doi.org/https://doi.org/10.1016/S1474-6670(17)48899-9

Fiechter, C.-N. (1997). Pac adaptive control of linear systems. *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, 72–80. https://doi.org/10.1145/267460.267481

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc.

Garatti, S., Campi, M. C., & Bittanti, S. (2004). Assessing the quality of identified models through the asymptotic theory-when is the result reliable? *Automatica*, *40*(8), 1319–1332. https://doi.org/10.1016/j.automatica.2004.03.005

Garcia, G., Bernussou, J., & Arzelier, D. (1994). Robust stabilization of discrete-time linear systems with norm-bounded time-varying uncertainty. *Systems and Control Letters*, *22*(5), 327–339. https://doi.org/https://doi.org/10.1016/0167-6911(94)90030-2

Gillis, J. T. (2011). State space. In W. S. Levine (Ed.), *Control system fundamentals* (2nd). CRC Press.

Ioannou, P. A., & Sun, J. (1995). *Robust adaptive control*. Prentice-Hall, Inc.

Kargin, T., Lale, S., Azizzadenesheli, K., Anandkumar, A., & Hassibi, B. (2022, February). Thompson sampling achieves $\tilde{O}(\sqrt{T})$ regret in linear quadratic control. In P.-L. Loh & M. Raginsky (Eds.),

Proceedings of thirty fifth conference on learning theory (pp. 3235–3284, Vol. 178). PMLR. https://proceedings.mlr.press/v178/kargin22a.html

Khalil, H. K. (2002). *Nonlinear systems* (3rd) [Chapter on Lyapunov Stability Theory]. Prentice Hall.

Kirk, D. (2004). *Optimal control theory: An introduction*. Dover Publications. https://books.google.com.au/books?id=fCh2SAtWIdwC

Ljung, L. (1999). *System identification (2nd ed.): Theory for the user*. Prentice Hall PTR.

Ljung, L., Andersson, C., Tiels, K., & Schön, T. B. (2020). Deep learning and system identification □ □this research was financially supported by the swedish foundation for strategic research (ssf) via the project assemble (contract number: Rit15-0012) and by the swedish research council via the projects learning flexible models for nonlinear dynamics (contract number: 2017-03807) and newleads - new directions in learning dynamical systems (contract number: 621-2016-06079). ljung's work was supported by vinnova's competence center linksic. [21st IFAC World Congress]. *IFAC-PapersOnLine*, *53*(2), 1175–1181. https://doi.org/https://doi.org/10.1016/j.ifacol.2020.12.1329

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, *2014*(239).

Ogata, K. (2010). *Modern control engineering* (5th). Prentice Hall.

Peaucelle, D., Arzelier, D., Bachelier, O., & Bernussou, J. (2000). A new robust d-stability condition for real convex polytopic uncertainty. *Systems and Control Letters*, *40*(1), 21–30. https://doi.org/https://doi.org/10.1016/S0167-6911(99)00119-X

Petersen, I. R., & Hollot, C. V. (1986). A riccati equation approach to the stabilization of uncertain linear systems. *Autom.*, *22*, 397–411. https://api.semanticscholar.org/CorpusID:31354237

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., & Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, *50*(3), 657–682. https://doi.org/https://doi.org/10.1016/j.automatica.2014.01.001

Redis Ltd. (2024). Redis [Accessed: 2025-05-10].

SciPy Community. (2025). *Scipy.spatial.convexhull — scipy v1.11.1 manual* [Accessed: 2025-05-30]. SciPy.org. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. A Bradford Book.

Szeliski, R. (2022, January). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer Cham.

Umenberger, J., Ferizbegovic, M., Schön, T. B., & Hjalmarsson, H. (2019). Robust exploration in linear quadratic reinforcement learning. https://proceedings.neurips.cc/paper_files/paper/2019/file/060fd70a06ead2e1079d27612b84aff4-Paper.pdf

Xie, L., & Soh, Y. C. (1993). Control of uncertain discrete-time systems with guaranteed cost. *Proceedings of 32nd IEEE Conference on Decision and Control*, 56–61 vol.1. https://doi.org/10.1109/CDC.1993.325190

Appendices

A Other Simulations implemented

Aside from the simulations presented in the Section 6, these simulated systems were also considered are part of the source code presented. However, due to limited times, these were not tuned to work with the overall architecture. Integration of these systems are left as part of the future work.

A.1 Simple Pendulum

State vector:

$$x = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T$$

Nonlinear dynamics:

$$\ddot{\theta} = \frac{u - d\dot{\theta} - mgL\cos\theta}{I}$$

A.2 Cart-Pendulum

State vector:

$$x = \begin{bmatrix} x_c & \dot{x}_c & \theta & \dot{\theta} \end{bmatrix}^T$$

Nonlinear dynamics:

$$\ddot{x}_c = \frac{u - mL\dot{\theta}^2\cos\theta + mg\sin\theta\cos\theta}{M + m - m\sin^2\theta}$$
$$\ddot{\theta} = -\frac{g}{L}\cos\theta - \frac{\sin\theta}{L}\ddot{x}_c$$

where u is the input force applied to the cart, x is the cart position, and the pendulum angle θ is measured from the positive x-axis.

B Evaluate ψ_t

B.1 SISO SPS

B.1.1 Open-loop

B.1.1.1 Case: General open-loop system

For a general system of the form

$$Y_t = G(z^{-1}, \theta^*)U_t + H(z^{-1}, \theta^*)N_t$$
$$\hat{N}_t(\theta) \triangleq H^{-1}(z^{-1}, \theta)(Y_t - G(z^{-1}, \theta)U_t)$$

Let G, H be causal transfer functions of the forms

$$G(z^{-1}, \theta) = \frac{G_{num}(z^{-1}, \theta)}{G_{den}(z^{-1}, \theta)}, H(z^{-1}, \theta) = \frac{H_{num}(z^{-1}, \theta)}{H_{den}(z^{-1}, \theta)} \tag{30}$$

Let the polynomials be defined as:

$$G_{\text{num}}(z^{-1}, \theta) = \sum_{i=1}^{d_{G_{\text{num}}}} g_i^{\text{num}} z^{-i}, \qquad G_{\text{den}}(z^{-1}, \theta) = 1 + \sum_{i=1}^{d_{G_{\text{den}}}} g_i^{\text{den}} z^{-i},$$
$$H_{\text{num}}(z^{-1}, \theta) = \sum_{i=0}^{d_{H_{\text{num}}}} h_i^{\text{num}} z^{-i}, \qquad H_{\text{den}}(z^{-1}, \theta) = 1 + \sum_{i=1}^{d_{H_{\text{den}}}} h_i^{\text{den}} z^{-i}$$

We assume that $g_0^{\text{num}} = 0$, i.e. there is input signal is delayed by at least 1 unit time. We set $g_0^{\text{den}} = h_0^{\text{den}} = 1$.

Then, the parameter row vector $\theta^T \in \mathbb{R}^{1\times(d_{G_{\text{num}}}+d_{G_{\text{den}}}+d_{H_{\text{num}}}+d_{H_{\text{den}}}+1)}$ is given by:

$$\theta^T = \left[g_1^{\text{num}}, \ldots, g_{d_{G_{\text{num}}}}^{\text{num}}, g_1^{\text{den}}, \ldots, g_{d_{G_{\text{den}}}}^{\text{den}}, h_0^{\text{num}}, \ldots, h_{d_{H_{\text{num}}}}^{\text{num}}, h_1^{\text{den}}, \ldots, h_{d_{H_{\text{den}}}}^{\text{den}}\right] \tag{31}$$

From the construction of θ, the gradient vector $\psi_t(\theta)$ can be evaluated as

$$\psi^T = \left[\frac{\partial\hat{N}_t(\theta)}{\partial g_1^{\text{num}}}, \ldots, \frac{\partial\hat{N}_t(\theta)}{\partial g_{d_{G_{\text{num}}}}^{\text{num}}}, \frac{\partial\hat{N}_t(\theta)}{\partial g_1^{\text{den}}}, \ldots, \frac{\partial\hat{N}_t(\theta)}{\partial g_{d_{G_{\text{den}}}}^{\text{den}}}, \frac{\partial\hat{N}_t(\theta)}{\partial h_0^{\text{num}}}, \ldots, \frac{\partial\hat{N}_t(\theta)}{\partial h_{d_{H_{\text{num}}}}^{\text{num}}}, \frac{\partial\hat{N}_t(\theta)}{\partial h_1^{\text{den}}}, \ldots, \frac{\partial\hat{N}_t(\theta)}{\partial h_{d_{H_{\text{den}}}}^{\text{den}}}\right] \tag{32}$$

where,

$$\begin{aligned}
\frac{\partial\hat{N}_t(\theta)}{\partial g_i^{\text{num}}} &= -H^{-1}(z^{-1}, \theta)\frac{z^{-i}}{G_{\text{den}}(z^{-1}, \theta)}U_t \\
\frac{\partial\hat{N}_t(\theta)}{\partial g_i^{\text{den}}} &= H^{-1}(z^{-1}, \theta)G(z^{-1}, \theta)\frac{z^{-i}}{G_{\text{den}}(z^{-1}, \theta)}U_t \\
\frac{\partial\hat{N}_t(\theta)}{\partial h_i^{\text{num}}} &= -\frac{z^{-i}}{H_{\text{num}}(z^{-1}, \theta)}\hat{N}_t(\theta) = -\frac{z^{-i}}{H_{\text{num}}^2(z^{-1}, \theta)}H_{\text{den}}(z^{-1}, \theta)(Y_t - G(z^{-1}, \theta)U_t) \\
\frac{\partial\hat{N}_t(\theta)}{\partial h_i^{\text{den}}} &= \frac{z^{-i}}{H_{\text{den}}(z^{-1}, \theta)}\hat{N}_t(\theta) = \frac{z^{-i}}{H_{\text{num}}(z^{-1}, \theta)}(z^{-1}, \theta)(Y_t - G(z^{-1}, \theta)U_t)
\end{aligned} \tag{33}$$

B.1.1.2 Case: ARMAX model

Consider an ARMAX system of the form $A(z^{-1},\theta)Y_t = B(z^{-1},\theta)U_t + C(z^{-1},\theta)N_t$. Then the causal transfer functions G, H become

$$G(z^{-1},\theta) = \frac{B(z^{-1},\theta)}{A(z^{-1},\theta)}, \quad H(z^{-1},\theta) = \frac{C(z^{-1},\theta)}{A(z^{-1},\theta)} \tag{34}$$

Similar to the general case, the polynomial A, B, C are

$$A(z^{-1},\theta) = \sum_{i=0}^{d_A} a_i z^{-i}, B(z^{-1},\theta) = \sum_{i=0}^{d_B} b_i z^{-i}, C(z^{-1},\theta) = \sum_{i=0}^{d_C} c_i z^{-i}$$

We note that $b_0 = 0$ to allow for input delay and set $a_0 = 1$. $d_{G_{\text{den}}} = d_{H_{\text{den}}} = d_A, d_{G_{\text{num}}} = d_B, d_{H_{\text{num}}} = d_C$ are polynomial degrees.

Then, the parameter row vector $\theta^T \in \mathbb{R}^{1\times(d_A+d_B+d_C+1)}$ or $\theta^T \in \mathbb{R}^{1\times(d_{G_{\text{num}}}+d_{G_{\text{den}}}+d_{H_{\text{num}}}+1)}$ is given by:

$$\theta^T = \Big[a_1,\ldots,a_{d_A},b_1,\ldots,b_{d_B},c_0,\ldots,c_{d_C}\Big] \tag{35}$$

$$\hat{N}_t = \frac{1}{C(z^{-1},\theta)}(A(z^{-1},\theta)Y_t - B(z^{-1},\theta)U_t)$$

From the construction of this θ, the gradient vector $\psi_t(\theta)$ simplifies to

$$\psi^T = \Big[\tfrac{\partial\hat{N}_t(\theta)}{\partial a_1},\ldots,\tfrac{\partial\hat{N}_t(\theta)}{\partial a_{d_A}},\tfrac{\partial\hat{N}_t(\theta)}{\partial b_1},\ldots,\tfrac{\partial\hat{N}_t(\theta)}{\partial b_{d_B}},\tfrac{\partial\hat{N}_t(\theta)}{\partial c_0},\ldots,\tfrac{\partial\hat{N}_t(\theta)}{\partial c_{d_C}}\Big] \tag{36}$$

where,

$$\begin{aligned}
\frac{\partial\hat{N}_t(\theta)}{\partial a_i} &= \frac{z^{-i}}{C(z^{-1},\theta)}Y_t \\
\frac{\partial\hat{N}_t(\theta)}{\partial b_i} &= -\frac{z^{-i}}{C(z^{-1},\theta)}U_t \\
\frac{\partial\hat{N}_t(\theta)}{\partial c_i} &= -\frac{z^{-i}}{C(z^{-1},\theta)}\hat{N}_t(\theta) = -\frac{z^{-i}}{C^2(z^{-1},\theta)}(A(z^{-1},\theta)Y_t - B(z^{-1},\theta)U_t)
\end{aligned} \tag{37}$$

Note that here we can see that the regression vector $\phi = -C(z^{-1},\theta)\psi^T$.

B.1.1.3 Case: State space model

Consider a standard state space model with n states,

$$\begin{aligned}
x(t+1) &= A_{\text{obs}}(\theta)x(t) + B_{\text{obs}}(\theta)u(t) \\
y(t) &= C_{\text{obs}}(\theta)x(t) + v(t)
\end{aligned} \tag{38}$$

where, $v(t) = H(z^{-1}, \theta)N_t$ is the noise model and the state space matrices, $A_{\text{obs}}(\theta) \in \mathbb{R}^{n\times n}, B_{\text{obs}}(\theta) \in \mathbb{R}^{n\times 1}, C_{\text{obs}}(\theta) \in \mathbb{R}^{1\times n}$ are in observer canonical form Gillis, 2011 s.t.

$$A_{\text{obs}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & \cdots & 0 & -a_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix}, B_{\text{obs}} = \begin{bmatrix} b_n \\ b_{n-1} \\ b_{n-2} \\ \vdots \\ b_1 \end{bmatrix}, C_{\text{obs}} = [0 \quad 0 \quad \cdots \quad 0 \quad 1] \tag{39}$$

corresponding to $Y_t = G(z^{-1}, \theta)U_t + H(z^{-1}, \theta)N_t$, where $H(z^{-1}, \theta)$ may have common poles with $G(z^{-1}, \theta)$ Let the polynomial A, B, C be

$$A(z^{-1}, \theta) \triangleq \sum_{i=0}^{d_A} a_i z^{-i}, B(z^{-1}, \theta) \triangleq \sum_{i=0}^{d_B} b_i z^{-i}, C(z^{-1}, \theta) \triangleq \sum_{i=0}^{d_C} c_i z^{-i}$$

The causal transfer functions G and H with common polynomial $A(z^{-1}, \theta)$ (ARMAX structure considered in this work) are given by:

$$\begin{aligned} G(z^{-1}, \theta) &= C_{\text{obs}}(\theta)[zI - A_{\text{obs}}(\theta)]^{-1}B(\theta) = \frac{B(z^{-1}, \theta)}{A(z^{-1}, \theta)} \\ H(z^{-1}, \theta) &= \frac{C(z^{-1}, \theta)}{A(z^{-1}, \theta)} \end{aligned} \tag{40}$$

with $d_A = d_B = n, a_0 = 1, b_0 = 0$.

Then, the parameter row vector $\theta^T \in \mathbb{R}^{1\times(d_A+d_B+d_C+1)}$ or $\theta^T \in \mathbb{R}^{1\times(2n+d_C+1)}$ is given by:

$$\theta^T = \begin{bmatrix} a_1, \ldots, a_{d_A}, b_1, \ldots, b_{d_B}, c_0, \ldots, c_{d_C} \end{bmatrix} \tag{41}$$

The predicton error is given by,

$$\hat{N}_t = \frac{1}{C(z^{-1}, \theta)}(A(z^{-1}, \theta)Y_t - B(z^{-1}, \theta)U_t)$$

From the construction of this θ, the gradient vector $\psi_t(\theta)$ simplifies to

$$\psi^T = \left[\frac{\partial \hat{N}_t(\theta)}{\partial a_1}, \ldots, \frac{\partial \hat{N}_t(\theta)}{\partial a_{d_A}}, \frac{\partial \hat{N}_t(\theta)}{\partial b_1}, \ldots, \frac{\partial \hat{N}_t(\theta)}{\partial b_{d_B}}, \frac{\partial \hat{N}_t(\theta)}{\partial c_0}, \ldots, \frac{\partial \hat{N}_t(\theta)}{\partial c_{d_C}} \right] \tag{42}$$

where,

$$\begin{aligned} \frac{\partial \hat{N}_t(\theta)}{\partial a_i} &= \frac{z^{-i}}{C(z^{-1}, \theta)}Y_t \\ \frac{\partial \hat{N}_t(\theta)}{\partial b_i} &= -\frac{z^{-i}}{C(z^{-1}, \theta)}U_t \\ \frac{\partial \hat{N}_t(\theta)}{\partial c_i} &= -\frac{z^{-i}}{C^2(z^{-1}, \theta)}(A(z^{-1}, \theta)Y_t - B(z^{-1}, \theta)U_t) \end{aligned} \tag{43}$$

B.2 MIMO SPS

B.2.1 Open loop

B.2.1.1 Case: General open loop

Lets consider θ of the form:

$$\theta = [\theta_G, \theta_H]^\top \tag{44}$$

, i.e. one large column vector of all free parameters in that order.

Then, ψ_t can be written as:

$$\frac{\partial \epsilon_t(\theta)}{\partial \theta} == \left[\frac{\partial \epsilon_t(\theta)}{\partial \theta_G}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_H}\right] \tag{45}$$

where,

$$\begin{aligned} \frac{\partial \epsilon_t(\theta)}{\partial \theta_G} &= -H^{-1}\frac{\partial G}{\partial \theta_G}U_t \\ \frac{\partial \epsilon_t(\theta)}{\partial \theta_H} &= -H^{-1}\frac{\partial H}{\partial \theta_H}H^{-1}(Y_t - GU_t) \end{aligned} \tag{46}$$

The partial derivatives of a filter matrix M, such as $G(z^{-1})$ or $H(z^{-1})$, with respect to its individual parameters can be succinctly expressed using the standard basis matrices $E_{i,j}$, where $E_{i,j}$ has the same dimensions as the corresponding filter and contains a 1 at the (i,j)-th position and zeros elsewhere. Specifically, for a parameter $\theta_{M_{i,j,k}}$ corresponding to the k-th coefficient of the (i,j)-th filter entry of the matrix M, the derivative $\eta = \frac{\partial M}{\partial \theta_M}$ is given component-wise by:

$$\eta_{i,j+k} = \frac{\partial M}{\partial \theta_{M_{i,j,k}}} = \begin{cases} z^{-k}E_{i,j}, & \text{if } \theta_{M_{i,j,k}} \in M_{i,j}^{\text{num}}(z^{-1}), \\ -\dfrac{z^{-k}}{M_{i,j}^{\text{den}}(z^{-1})}M_{i,j}(z^{-1})E_{i,j}, & \text{if } \theta_{M_{i,j,k}} \in M_{i,j}^{\text{den}}(z^{-1}). \end{cases}$$

B.2.1.2 Case: State-space formulation

Consider a standard state-space model with n states and structured similarly to (38), with appropriate dimensions. Let

$$A_{\text{obs}}(\theta) \in \mathbb{R}^{n\times n}, \quad B_{\text{obs}}(\theta) \in \mathbb{R}^{n\times r}, \quad C_{\text{obs}}(\theta) \in \mathbb{R}^{q\times n}$$

denote the system matrices in observer canonical form Gillis, 2011.

Let $A(z^{-1}) = [1, a_1, \cdots, a_n]$ denote the matrix of polynomial coefficients.

$$A_{\text{obs}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & \cdots & 0 & -a_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix} \tag{47}$$

Let $B(z^{-1}) \in \mathbb{R}^{r\times(n+1)}$ denote the matrix of polynomial coefficients, where each row contains the coefficients $[0, b_{(i-1)n+1}, \ldots, b_{in}]$ of the i-th input polynomial. Then, we define:

$$B_{\text{obs}} := \begin{bmatrix} b_n & b_{2n} & \cdots & b_{rn} \\ b_{n-1} & b_{2n-1} & \cdots & b_{rn-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_{n+1} & \cdots & b_{(r-1)n+1} \end{bmatrix} \in \mathbb{R}^{n\times r},$$

which is formed by removing the leading zero column from $B(z^{-1})$, transposing the result, and reversing the order of the coefficients in each column.

The output matrix $C_{\text{obs}} \in \mathbb{R}^{q\times n}$ selects the j-th state corresponding to the i-th output, i.e., $C_{i,j} = 1$ and zero elsewhere. For example,

$$C_{\text{obs}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

corresponds to the mapping $y_0 = x_2$, $y_1 = x_0$.

The causal transfer function, G, is given by

$$G(z^{-1}, \theta) = C_{\text{obs}}(\theta)[zI - A_{\text{obs}}(\theta)]^{-1}B_{\text{obs}}(\theta).$$

We can now decompose G with respect to the parameters $\theta_G = [\theta_A, \theta_B, \theta_C]$ as

$$\frac{\partial G}{\partial \theta_G} = \left[\frac{\partial G}{\partial \theta_A}, \frac{\partial G}{\partial \theta_B}, \frac{\partial G}{\partial \theta_C}\right].$$

Let $\theta_A = [a_1, \cdots, a_n]$. The Jacobian with respect to θ_A becomes

$$\frac{\partial G}{\partial \theta_A} = C_{\text{obs}}(\theta)[zI - A_{\text{obs}}(\theta)]^{-1}\frac{\partial A_{\text{obs}}}{\partial \theta_A}[zI - A_{\text{obs}}(\theta)]^{-1}B_{\text{obs}}(\theta),$$

where the partial derivative of A_{obs} with respect to each a_k is

$$\frac{\partial A_{\text{obs}}}{\partial a_k} = -E_{n-k+1,\,n}, \quad \text{for } k = 1, \ldots, n,$$

and $E_{i,j}$ denotes the matrix with 1 in the (i, j)-th position and zero elsewhere. Then, stacking these gives

$$\frac{\partial A_{\text{obs}}}{\partial \theta_A} = \begin{bmatrix} \frac{\partial A_{\text{obs}}}{\partial a_1} & \cdots & \frac{\partial A_{\text{obs}}}{\partial a_n} \end{bmatrix} \in \mathbb{R}^{n^2\times n}.$$

Equivalently, the Jacobian can be expressed as

$$\frac{\partial G}{\partial \theta_A} = \sum_{k=1}^{n}\left[C_{\text{obs}}(z)(zI - A_{\text{obs}})^{-1}\left(\frac{\partial A_{\text{obs}}}{\partial a_k}\right)(zI - A_{\text{obs}})^{-1}B_{\text{obs}}\right] \otimes e_k^\top \in \mathbb{R}^{q\times r\times n},$$

where e_k is the k-th standard basis vector in $\mathbb{R}^n$.

Let $\theta_B = [b_1, \ldots, b_{rn}]$. The Jacobian with respect to θ_B becomes:

$$\frac{\partial G}{\partial \theta_B} = \sum_{k=1}^{rn}\left[C_{\text{obs}}(\theta)(zI - A_{\text{obs}}(\theta))^{-1}\left(\frac{\partial B_{\text{obs}}}{\partial b_k}\right)\right] \otimes e_k^\top \in \mathbb{R}^{q\times r\times rn}.$$

where the partial derivative $\frac{\partial B_{\text{obs}}}{\partial b_k} = E_{i,j}$ is the standard basis matrix with a 1 at the (i, j)-th position and zeros elsewhere, corresponding to the k-th element in B_{obs}.

Now $\frac{\partial \epsilon_t}{\partial \theta_G}$ and $\frac{\partial \epsilon_t}{\partial \theta_H}$ can be calculated as stated in the general open-loop case (46) without any shared parameters.

B.2.1.3 Case: State-space with shared $A(z^{-1})$

Usually, a commonly imposed structure on H is:

$$H(z^{-1}) = \frac{1}{A(z^{-1})} H^{\text{num}}(z^{-1}) = \frac{1}{A(z^{-1})} \begin{bmatrix} C_1(z^{-1}) & 0 & \cdots & 0 \\ 0 & C_2(z^{-1}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_q(z^{-1}) \end{bmatrix}, \tag{48}$$

where each $C_i(z^{-1})$ is a stable polynomial associated with the i-th output, $A(z^{-1})$ is the shared monic denominator polynomial and $e_k \in \mathbb{R}^n$ is the k-th standard basis vector. Now the Jacobians of H are

$$\begin{aligned} \frac{\partial H}{\partial \theta_A} &= \sum_{k=1}^{n} \left[-\frac{z^{-k}}{A(z^{-1})} H(z^{-1}) \right] \otimes e_k^\top \in \mathbb{R}^{q \times q \times n}, \\ \frac{\partial H}{\partial \theta_C} &= \sum_{k=1}^{qn} \left[\frac{z^{-j_k}}{A(z^{-1})} E_{i_k, i_k} \right] \otimes e_k^\top \in \mathbb{R}^{q \times q \times qn}, \text{where, } \theta_C = [c_1, \ldots, c_{qn}] \end{aligned} \tag{49}$$

Therfore, the ψ can be written as follows

$$\frac{\partial \epsilon_t(\theta)}{\partial \theta} = \left[\frac{\partial \epsilon_t(\theta)}{\partial \theta_A}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_B}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_C} \right] \tag{50}$$

$$\begin{aligned} \frac{\partial \epsilon_t(\theta)}{\partial \theta_A} &= -H^{-1} \frac{\partial G}{\partial \theta_A} U_t - H^{-1} \frac{\partial H}{\partial \theta_A} H^{-1} (Y_t - G U_t) \\ \frac{\partial \epsilon_t(\theta)}{\partial \theta_B} &= -H^{-1} \frac{\partial G}{\partial \theta_B} U_t \\ \frac{\partial \epsilon_t(\theta)}{\partial \theta_C} &= -H^{-1} \frac{\partial H}{\partial \theta_C} H^{-1} (Y_t - G U_t) \end{aligned} \tag{51}$$

B.2.2 Closed loop

Using the ϵ_t closed loop formulation in (20), and the open loop jacobians, the closed-loop ψ_t can be written. In general,

$$\theta = [\theta_G, \theta_H]^\top, \frac{\partial \epsilon_t(\theta)}{\partial \theta} = \left[\frac{\partial \epsilon_t(\theta)}{\partial \theta_G}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_H} \right]$$

$$\begin{aligned} \frac{\partial \epsilon_t}{\partial \theta_G} &= H^{-1} \frac{\partial G}{\partial \theta_G} (F Y_t - L R_t) \\ \frac{\partial \epsilon_t}{\partial \theta_H} &= -H^{-1} \frac{\partial H}{\partial \theta_H} H^{-1} (Y_t - G(F Y_t - L R_t)) \end{aligned}$$

With the state-space structure and shared $A(z^{-1})$, then:

$$\theta = [\theta_A, \theta_B, \theta_C]^\top, \frac{\partial \epsilon_t(\theta)}{\partial \theta} = \left[\frac{\partial \epsilon_t(\theta)}{\partial \theta_A}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_B}, \frac{\partial \epsilon_t(\theta)}{\partial \theta_C} \right]$$

$$\frac{\partial \epsilon_t(\theta)}{\partial \theta_A} = H^{-1}\frac{\partial G}{\partial \theta_A}(FY_t - LR_t) - H^{-1}\frac{\partial H}{\partial \theta_A}H^{-1}(Y_t - G(FY_t - LR_t))$$
$$\frac{\partial \epsilon_t(\theta)}{\partial \theta_B} = H^{-1}\frac{\partial G}{\partial \theta_B}(FY_t - LR_t) \tag{52}$$
$$\frac{\partial \epsilon_t(\theta)}{\partial \theta_C} = -H^{-1}\frac{\partial H}{\partial \theta_C}H^{-1}(Y_t - G(FY_t - LR_t))$$

C Evaluating SPS ψ_t for the simulations considered

C.1 SISO system with 2 states

The SISO system is 1-input-output system modelled with 2 states. where,

$$A_{obs} = \begin{bmatrix} 0 & -a_2 \\ 1 & -a_1 \end{bmatrix}, B_{obs} = \begin{bmatrix} b_2 \\ b_1 \end{bmatrix}, C_obs = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$\begin{aligned} A(z^{-1}, \theta) &= 1 + a_1 z^{-1} + a_2 z^{-2} \\ B(z^{-1}, \theta) &= b_1 z^{-1} + b_2 z^{-2} \\ C(z^{-1}, \theta) &= c_0 \end{aligned}$$

$$\begin{aligned} \frac{\partial \epsilon_t(\theta)}{\partial a_i} &= \frac{z^{-i}}{C(z^{-1}, \theta)} Y_t \\ \frac{\partial \hat{\epsilon}_t(\theta)}{\partial b_i} &= -\frac{z^{-i}}{C(z^{-1}, \theta)} U_t \\ \frac{\partial \epsilon_t(\theta)}{\partial c_i} &= -\frac{z^{-i}}{C^2(z^{-1}, \theta)} (A(z^{-1}, \theta) Y_t - B(z^{-1}, \theta) U_t) \end{aligned}$$

C.2 Simple pendulum with 2 states

The Simple pendulum system is modelled with 2 states, 1 input. We fully observe the system where,

$$A_{obs} = \begin{bmatrix} 0 & -a_2 \\ 1 & -a_1 \end{bmatrix}, B_{obs} = \begin{bmatrix} b_2 \\ b_1 \end{bmatrix}, C_obs = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{aligned} A(z^{-1}, \theta) &= 1 + a_1 z^{-1} + a_2 z^{-2} \\ B(z^{-1}, \theta) &= b_1 z^{-1} + b_2 z^{-2} \\ P(z^{-1}, \theta) &= b_2 z^{-1} + (a_1 b_2 - a_2 b_1) z^{-2} \\ C(z^{-1}, \theta) &= 1 \end{aligned}$$

$$G(z^{-1}, \theta) = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} B(z^{-1}, \theta) \\ P(z^{-1}, \theta) \end{bmatrix}, H(z^{-1}, \theta) = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\frac{\partial \epsilon_t(\theta)}{\partial \theta} = \begin{bmatrix} z^{-1} Y_t^{(1)}, & z^{-2} Y_t^{(1)}, & -z^{-1} U_t, & -z^{-2} U_t \\ z^{-1} Y_t^{(2)} - b_2 z^{-2} U_t, & z^{-2} Y_t^{(2)} + b_1 z^{-2} U_t, & a_2 z^{-2} U_t, & -(z^{-1} + a_1 z^{-2}) U_t \end{bmatrix}$$

C.3 Cart-Pendulum

The cart-pendulum system is a 1-input 2-output system, modelled with 4 states and degree of noise polynomials is 1, where

$$A_{obs} = \begin{bmatrix} 0 & 0 & 0 & -a_4 \\ 1 & 0 & 0 & -a_3 \\ 0 & 1 & 0 & -a_2 \\ 0 & 0 & 1 & -a_1 \end{bmatrix}, B_{obs} = \begin{bmatrix} b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix}, C_obs = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A(z^{-1}, \theta) = 1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}$$

$$B(z^{-1}, \theta) = b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}$$

C.3.1 Open-loop

C.3.1.1 Filter Matrix G,H

$$G(z^{-1}, \theta) = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} \\ b_3 z^{-1} + (b_4 + a_1 b_3 - a_3 b_1) z^{-2} + (a_1 b_4 + a_2 b_3 - a_3 b_2 - a_4 b_1) z^{-3} + (a_2 b_4 - a_4 b_2) z^{-4} \end{bmatrix}$$

$$H(z^{-1}, \theta) = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} C_1(z^{-1}) & 0 \\ 0 & C_2(z^{-1}) \end{bmatrix} = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} c_0 + c_1 z^{-1} & 0 \\ 0 & c_2 + c_3 z^{-1} \end{bmatrix}$$

Let,

$$P(z^{-1}, \theta) = b_3 z^{-1} + (b_4 + a_1 b_3 - a_3 b_1) z^{-2} + (a_1 b_4 + a_2 b_3 - a_3 b_2 - a_4 b_1) z^{-3} + (a_2 b_4 - a_4 b_2) z^{-4}$$

C.3.1.2 Jacobian of G

The derivatives of $G_{1,1}$ with respect to θ are basically the SISO case. The derivatives of $G_{2,2}$ can be found in close form as

$$\frac{\partial G(2,1)}{\partial a_1} = \frac{\left(a_3 z^{-2} + a_4 z^{-3}\right) B(z^{-1})}{A^2(z^{-1})}, \qquad \frac{\partial G(2,1)}{\partial a_2} = \frac{\left(a_3 z^{-3} + a_4 z^{-4}\right) B(z^{-1})}{A^2(z^{-1})},$$

$$\frac{\partial G(2,1)}{\partial a_3} = \frac{-\left(a_2 z^{-3} + a_1 z^{-2} + z^{-1}\right) B(z^{-1})}{A^2(z^{-1})}, \qquad \frac{\partial G(2,1)}{\partial a_4} = \frac{-\left(a_2 z^{-4} + a_1 z^{-3} + z^{-2}\right) B(z^{-1})}{A^2(z^{-1})}$$

$$\frac{\partial G(2,1)}{\partial b_1} = \frac{-a_3 z^{-2} - a_4 z^{-3}}{A(z^{-1})}, \qquad \frac{\partial G(2,1)}{\partial b_2} = \frac{-a_3 z^{-3} - a_4 z^{-4})}{A(z^{-1})},$$

$$\frac{\partial G(2,1)}{\partial b_3} = \frac{z^{-1} + a_1 z^{-2} + a_2 z^{-3}}{A(z^{-1})}, \qquad \frac{\partial G(2,1)}{\partial b_4} = \frac{z^{-2} + a_1 z^{-3} + a_2 z^{-4}}{A(z^{-1})}$$

Then the Jacobians of G are

$$\frac{\partial G}{\partial \theta_A} = \frac{B(z^{-1})}{A^2(z^{-1})} \begin{bmatrix} -z^{-1} & -z^{-2} & -z^{-3} & -z^{-4} \\ a_3 z^{-2} + a_4 z^{-3} & +a_3 z^{-3} + a_4 z^{-4} & -(z^{-1} + a_1 z^{-2} + a_2 z^{-3}) & -(z^{-2} + a_1 z^{-3} + a_2 z^{-4}) \end{bmatrix}$$

$$\frac{\partial G}{\partial \theta_B} = \frac{1}{A(z^{-1})} \begin{bmatrix} z^{-1} & z^{-2} & z^{-3} & z^{-4} \\ -a_3 z^{-2} - a_4 z^{-3} & -a_3 z^{-3} - a_4 z^{-4} & z^{-1} + a_1 z^{-2} + a_2 z^{-3} & z^{-2} + a_1 z^{-3} + a_2 z^{-4} \end{bmatrix}$$

C.3.1.3 Jacobian of H

The Jacobians of H are,

$$\frac{\partial H}{\partial \theta_A} = \frac{1}{A^2(z^{-1})}\begin{bmatrix} -z^{-1}H & -z^{-2}H & -z^{-3}H & -z^{-4}H \end{bmatrix}$$

$$\frac{\partial H}{\partial \theta_C} = \begin{bmatrix} \frac{\partial H}{\partial \theta_{c_1}} & \frac{\partial H}{\partial \theta_{c_2}} & \frac{\partial H}{\partial \theta_{c_3}} & \frac{\partial H}{\partial \theta_{c_4}} \end{bmatrix}$$

$$\frac{\partial H}{\partial \theta_{c_0}} = \frac{1}{A(z^{-1})}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \frac{\partial H}{\partial \theta_{c_1}} = \frac{1}{A(z^{-1})}\begin{bmatrix} z^{-1} & 0 \\ 0 & 0 \end{bmatrix}, \frac{\partial H}{\partial \theta_{c_2}} = \frac{1}{A(z^{-1})}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \frac{\partial H}{\partial \theta_{c_3}} = \frac{1}{A(z^{-1})}\begin{bmatrix} 0 & 0 \\ 0 & z^{-1} \end{bmatrix}$$

C.3.1.4 Jacobians of Prediction error

The Jacobians of the ϵ are:

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_A} = \begin{bmatrix} \frac{z^{-1}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-2}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-3}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-4}}{C_1(z^{-1})}Y_t^{(1)} \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_B} = \begin{bmatrix} \frac{-z^{-1}}{C_1(z^{-1})}U_t, & \frac{-z^{-2}}{C_1(z^{-1})}U_t, & \frac{-z^{-3}}{C_1(z^{-1})}U_t, & \frac{-z^{-4}}{C_1(z^{-1})}U_t \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_C} = \begin{bmatrix} \frac{-1}{(C_1(z^{-1}))^2}(A(z^{-1})Y_t^{(1)} - B(z^{-1})U_t), & \frac{-z^{-1}}{(C_1(z^{-1}))^2}(A(z^{-1})Y_t^{(1)} - B(z^{-1})U_t), & 0, & 0 \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_A} = \begin{bmatrix} \frac{z^{-1}}{C_2(z^{-1})}(Y_t^{(2)} - (b_3z^{-1} + b_4z^{-2})U_t), & \frac{z^{-2}}{C_2(z^{-1})}(Y_t^{(2)} - (b_3z^{-1} + b_4z^{-2})U_t) \\ \frac{z^{-2}}{C_2(z^{-1})}(z^{-1}Y_t^{(2)} + (b_1 + b_2z^{-1})U_t), & \frac{z^{-3}}{C_2(z^{-1})}(z^{-1}Y_t^{(2)} + (b_1 + b_2z^{-1})U_t) \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_B} = \begin{bmatrix} \frac{z^{-2}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t, & \frac{z^{-3}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t, & \frac{-z^{-1}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t, & \frac{-z^{-2}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_C} = \begin{bmatrix} 0, & 0, & \frac{-1}{(C_2(z^{-1}))^2}(A(z^{-1})Y_t^{(2)} - P(z^{-1})U_t), & \frac{-z^{-1}}{(C_2(z^{-1}))^2}(A(z^{-1})Y_t^{(2)} - P(z^{-1})U_t) \end{bmatrix}$$

where $(\cdot)^{(i)that}$ represents the i-th element.

C.3.2 Closed-loop

C.3.2.1 Jacobians of Prediction error

The Jacobian $\frac{\partial \epsilon}{\partial \theta}$ is same as open loop case, however substitute U_t with

$$U_t = FY_t - LY_t = [F_1, F_2]\begin{bmatrix} Y_t^{(1)} \\ Y_t^{(2)} \end{bmatrix} - [L_1, L_2]\begin{bmatrix} R_t^{(1)} \\ R_t^{(2)} \end{bmatrix} = F_1Y_t^{(1)} + F_2Y_t^{(2)} - L_1R_t^{(1)} - L_2R_t^{(2)}$$

C.4 CARLA

The state space formulation for 4 states, 2 input, 2 output, noise polynomial of degree 1 are

$$A_{obs} = \begin{bmatrix} 0 & 0 & 0 & -a_4 \\ 1 & 0 & 0 & -a_3 \\ 0 & 1 & 0 & -a_2 \\ 0 & 0 & 1 & -a_1 \end{bmatrix}, B_{obs} = \begin{bmatrix} b_4 & b_8 \\ b_3 & b_7 \\ b_2 & b_6 \\ b_1 & b_5 \end{bmatrix}, C_obs = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} A(z^{-1}, \theta) &= 1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} \\ B_1(z^{-1}, \theta) &= b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} \\ B_2(z^{-1}, \theta) &= b_5 z^{-1} + b_6 z^{-2} + b_7 z^{-3} + b_8 z^{-4} \\ C_1(z^{-1}, \theta) &= c_0 + c_1 z^{-1} \\ C_2(z^{-1}, \theta) &= c_2 + c_3 z^{-1} \end{aligned}$$

C.4.1 Open-loop

C.4.1.1 Filter matrix G

Let $F(P(z^{-1}), A(z^{-1})) \coloneqq F_P$ be a function which takes in a Polynomial $P = p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3} + p_4 z^{-4}$ and A defined above and returns a new Polynomial:

$$F_P \triangleq p_3 z^{-1} + (p_4 + a_1 p_3 - a_3 p_1) z^{-2} + (a_1 p_4 + a_2 p_3 - a_3 p_2 - a_4 p_1) z^{-3} + (a_2 p_4 - a_4 p_2) z^{-4}$$

$$G = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} B_1(z^{-1}) & B_2(z^{-1}) \\ F_{B_1}(z^{-1}) & F_{B_2}(z^{-1}) \end{bmatrix}, H(z^{-1}, \theta) = \frac{1}{A(z^{-1}, \theta)} \begin{bmatrix} C_1(z^{-1}) & 0 \\ 0 & C_2(z^{-1}) \end{bmatrix}$$

H is same as Cart-pendulum case.

C.4.1.2 Jacobians of G

First Column

$$\frac{\partial G(:,1)}{\partial \theta_A} = \frac{B_1(z^{-1})}{A^2(z^{-1})} \begin{bmatrix} -z^{-1} & -z^{-2} & -z^{-3} & -z^{-4} \\ a_3 z^{-2} + a_4 z^{-3} & +a_3 z^{-3} + a_4 z^{-4} & -(z^{-1} + a_1 z^{-2} + a_2 z^{-3}) & -(z^{-2} + a_1 z^{-3} + a_2 z^{-4}) \end{bmatrix}$$

$$\frac{\partial G(:,1)}{\partial \theta_{B_1}} = \frac{1}{A(z^{-1})} \begin{bmatrix} z^{-1} & z^{-2} & z^{-3} & z^{-4} \\ -a_3 z^{-2} - a_4 z^{-3} & -a_3 z^{-3} - a_4 z^{-4} & z^{-1} + a_1 z^{-2} + a_2 z^{-3} & z^{-2} + a_1 z^{-3} + a_2 z^{-4} \end{bmatrix}$$

$$\frac{\partial G(:,1)}{\partial \theta_{B_2}} = \mathbf{0}_{2\times 4}$$

Second Column

$$\frac{\partial G(:,2)}{\partial \theta_A} = \frac{B_2(z^{-1})}{A^2(z^{-1})}\begin{bmatrix} -z^{-1} & -z^{-2} & -z^{-3} & -z^{-4} \\ a_3z^{-2}+a_4z^{-3} & +a_3z^{-3}+a_4z^{-4} & -(z^{-1}+a_1z^{-2}+a_2z^{-3}) & -(z^{-2}+a_1z^{-3}+a_2z^{-4}) \end{bmatrix}$$

$$\frac{\partial G(:,2)}{\partial \theta_{B_1}} = \mathbf{0}_{2\times 4}$$

$$\frac{\partial G(:,2)}{\partial \theta_{B_2}} = \frac{1}{A(z^{-1})}\begin{bmatrix} z^{-1} & z^{-2} & z^{-3} & z^{-4} \\ -a_3z^{-2}-a_4z^{-3} & -a_3z^{-3}-a_4z^{-4} & z^{-1}+a_1z^{-2}+a_2z^{-3} & z^{-2}+a_1z^{-3}+a_2z^{-4} \end{bmatrix}$$

C.4.1.3 Jacobians of Predicton error ϵ

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_A} = \begin{bmatrix} \frac{z^{-1}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-2}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-3}}{C_1(z^{-1})}Y_t^{(1)}, & \frac{z^{-4}}{C_1(z^{-1})}Y_t^{(1)} \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_B} = \begin{bmatrix} \frac{-z^{-1}}{C_1(z^{-1})}U_t^{(1)}, & \frac{-z^{-2}}{C_1(z^{-1})}U_t^{(1)}, & \frac{-z^{-3}}{C_1(z^{-1})}U_t^{(1)}, & \frac{-z^{-4}}{C_1(z^{-1})}U_t^{(1)} \\ \frac{-z^{-1}}{C_1(z^{-1})}U_t^{(2)}, & \frac{-z^{-2}}{C_1(z^{-1})}U_t^{(2)}, & \frac{-z^{-3}}{C_1(z^{-1})}U_t^{(2)}, & \frac{-z^{-4}}{C_1(z^{-1})}U_t^{(2)}, \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(1)}}{\partial \theta_C} = \begin{bmatrix} \frac{1}{(C_1(z^{-1}))^2}(B_1(z^{-1})U_t^{(1)} + B_2(z^{-1})U_t^{(2)} - A(z^{-1})Y_t^{(1)}) \\ \frac{z^{-1}}{(C_1(z^{-1}))^2}(B_1(z^{-1})U_t^{(1)} + B_2(z^{-1})U_t^{(2)} - A(z^{-1})Y_t^{(1)}), & 0, & 0 \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_A} = \begin{bmatrix} \frac{z^{-1}}{C_2(z^{-1})}(Y_t^{(2)} - (b_3z^{-1}+b_4z^{-2})U_t^{(1)} - (b_7z^{-1}+b_8z^{-2})U_t^{(2)}) \\ \frac{z^{-2}}{C_2(z^{-1})}(Y_t^{(2)} - (b_3z^{-1}+b_4z^{-2})U_t^{(1)} - (b_7z^{-1}+b_8z^{-2})U_t^{(2)}) \\ \frac{z^{-2}}{C_2(z^{-1})}(z^{-1}Y_t^{(2)} + (b_1+b_2z^{-1})U_t^{(1)} + (b_5+b_6z^{-1})U_t^{(2)}) \\ \frac{z^{-3}}{C_2(z^{-1})}(z^{-1}Y_t^{(2)} + (b_1+b_2z^{-1})U_t + (b_5+b_6z^{-1})U_t^{(2)}) \end{bmatrix}^\top$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_B} = \begin{bmatrix} \frac{z^{-2}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t^{(1)}, & \frac{z^{-3}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t^{(1)}, & \frac{-z^{-1}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t^{(1)}, & \frac{-z^{-2}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t^{(1)} \\ \frac{z^{-2}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t^{(2)}, & \frac{z^{-3}(a_3+a_4z^{-1})}{C_2(z^{-1})}U_t^{(2)}, & \frac{-z^{-1}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t^{(2)}, & \frac{-z^{-2}(1+a_1z^{-1}+a_2z^{-2})}{C_2(z^{-1})}U_t^{(2)} \end{bmatrix}$$

$$\frac{\partial \epsilon_t^{(2)}}{\partial \theta_C} = \begin{bmatrix} 0, & 0, & \frac{1}{(C_2(z^{-1}))^2}(A(z^{-1})Y_t^{(2)} - F_{B_1}(z^{-1})U_t^{(1)} - F_{B_2}(z^{-1})U_t^{(2)}), \\ \frac{-z^{-1}}{(C_2(z^{-1}))^2}(A(z^{-1})Y_t^{(2)} - F_{B_1}(z^{-1})U_t^{(1)} - F_{B_2}(z^{-1})U_t^{(2)}) \end{bmatrix}$$

where $(\cdot)^i$ represents the i-th element.

C.4.2 Closed-Loop

C.4.2.1 Jacobian of Prediction error

The jacobian is the same as open loop case, however replace U_t with controller rule.

$$U_t = FY_t - LR_t = \begin{bmatrix} F_{1,1}Y_t^{(1)} + F_{1,2}Y_t^{(2)} - L_{1,1}R_t^{(1)} + L_{1,2}R_t^{(2)} \\ F_{2,1}Y_t^{(1)} + F_{2,2}Y_t^{(2)} - L_{2,1}R_t^{(1)} + L_{2,2}R_t^{(2)} \end{bmatrix}$$

D SPS Recursive

The SPS recursive algorithm may be of interest, as it allows incorporating past ranking information for a given point of interest. However there are a few limitations which prevent us from achieving truly recursive algorithm. First and formost, the α used for generating m perturbations must be initialised once and kept the same to ensure each index corresponds to the same perturbation throughout the run time.

Then S_i^n, i-th perturbation at $t = n$ can be written as:

$$\begin{aligned} S_i^n(\theta) &= \left(\sum_{t=1}^{n} R_i^t(\theta, \alpha_i)\right)^{-\frac{1}{2}} \sum_{t=1}^{n} P_i^t(\theta, \alpha_i) \\ R_i^t(\theta, \alpha_i) &= \bar{\psi}_t(\theta, \alpha_i)\bar{\psi}_t^T(\theta, \alpha_i), \\ P_i^t(\theta, \alpha_i) &= \alpha_{i,t}\bar{\psi}_t(\theta, \alpha_i)\hat{N}_t(\theta) \end{aligned}$$

However, due to the nonlinear matrix inverse square root in the weighting term of $S_n(\theta)$, it is not possible to express $S_i^n(\theta)$ directly in terms of $S_i^{n-1}(\theta)$ alone. It is possible to perform recursive updates on R_i^t and P_i^t to achieve a recursive SPS, where past information is included.

$$\begin{aligned} S_i^n(\theta) &= R_i^n(\theta, \alpha_i)^{-\frac{1}{2}} P_i^n(\theta, \alpha_i) \\ R_i^n(\theta, \alpha_i) &= R_i^{n-1}(\theta, \alpha_i) + \bar{\psi}_n(\theta, \alpha_i)\bar{\psi}_n^T(\theta, \alpha_i), \\ P_i^n(\theta, \alpha_i) &= P_i^{n-1}(\theta, \alpha_i) + \alpha_{i,n}\bar{\psi}_n(\theta, \alpha_i)\hat{N}_n(\theta) \end{aligned} \tag{53}$$

The main issue with any recursive approach is memory and time constraints, while theoretically a fine grid search over a region can be performed using the recursive SPS however computing the inverse and sorting operations every time step for m perturbation would be computationally intensive and doesnt not scale well with dimensions. And if a random search based approach or a more complex search approach is adopted, then the sums would either have to be interpolated based on neighbours or a timer based approach could be adopted where if a point hasnt recieved an update within a given time window then the information stored regarding that point could be safely ignored. Both of these approaches bring in complications and lead to heuristic-based region constructions

The issue of repetitive inverse calculations can be mitigated by performing batch updates, however other issues of time constraints regarding searching the SPS regions remain in place.

D.1 Batch Update (Batch Size k)

Batch updates can be performed in an almost recursive fashion by accumulating multiple updates at once. For a batch of size k, the updates generalise as follows:

$$\begin{aligned} S_{n+k}(\theta) &= R_{n+k}^{-\frac{1}{2}}(\theta)\, P_{n+k}(\theta), \\ R_{n+k}(\theta) &= R_{n-1}(\theta) + \sum_{t=n}^{n+k} \psi_t(\theta)\psi_t^T(\theta), \\ P_{n+k}(\theta) &= P_{n-1}(\theta) + \sum_{t=n}^{n+k} \psi_t(\theta)\hat{N}_t(\theta). \end{aligned} \tag{54}$$

This approach ensures that future rankings incorporate past data. However, for each $\theta \in \mathbb{R}^d$ (i.e., with d free parameters), the algorithm must store $R_n \in \mathbb{R}^{d\times d}$ and $P_n \in \mathbb{R}^{d\times 1}$ for all m disturbances. This results in a memory requirement of

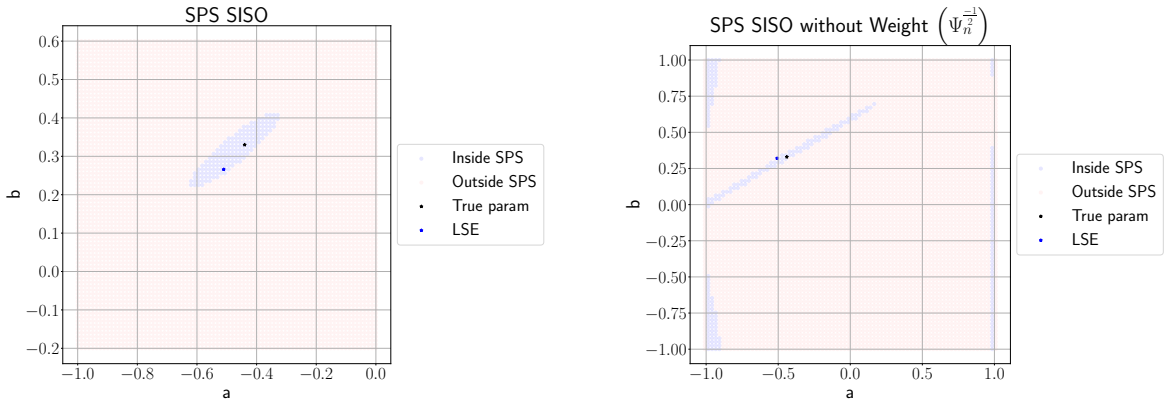
$$8md^2 + 8md = 8md(d+1) \text{ bytes.}$$

Assuming $m = 100$, this gives a total of $800d(d+1)$ bytes. For instance, when $d = 6$, the storage needed is approximately 32.81 kB. While this may seem modest, the SPS indicator is typically used in a search over a large number of θ values to construct a region. For example, searching over 6000 points would require approximately 192.2 MB of memory **not including any additional overhead**. While this approach is optimistic in terms of memory requirements at a given time t, the concern of searching for SPS region in real time is still present and the approach is only applicable when all θ and α are same for all times. Else, an heuristic based approach needs to adopted to combine the past infromation from all θ ever encountered. The fusion based heuristic approaches are further explored in Section 4.2.

D.2 Recursive formulation without $\Psi_n^{-\frac{1}{2}}$ weight

The weight matrix, $\Psi_n^{-\frac{1}{2}}$, is utilised to shape the confidence region and it is possible to write the SPS algorithm without it which can lead to a fully recursive solutions without computing expensive inverses (Csaji et al., 2015). However it does imply that regions now are no longer nicely shaped and may lead to incorrect results when further away form the least square estimate. This leads to recursive updates being:

$$S_i^n(\theta) = S_i^{n-1}(\theta) + \alpha_{i,n}\bar{\psi}_n(\theta, \alpha_i)\hat{N}_n(\theta)$$



(a) With weight

(b) Without weight

Figure 13: Comparison of the system with and without weighting. Note (a) and (b) have different x and y axis scales.

E Min-max problem for a first order system

Given proportional ("full-state") feedback $u = -kx$ the cost function becomes:

$$J = \sum_{k=0}^{\infty} (q + rk^2)x_k^2 = (q + rk^2) \sum_{k=0}^{\infty} x_k^2$$

and with the state dynamics $x_{k+1} = x_k(a - bk)$ we have

$$\begin{aligned}
\sum_{k=0}^{\infty} x_k^2 &= x_0^2 + x_1^2 + x_2^2 + ... \\
&= x_0^2 + x_0^2(a - bk)^2 + x_0^2(a - bk)^4 + ... \\
&= x_0^2 \sum_{k=0}^{\infty} ((a - bk)^2)^k \\
&= x_0^2 \frac{1}{1 - (a - bk)^2}
\end{aligned}$$

assuming $(a - bk)^2 < 1 \leftrightarrow |a - bk| < 1$ i.e. the closed-loop system is stable.

Therefore, the cost function to optimise is

$$J = x_0^2 \frac{(q + rk^2)}{1 - (a - bk)^2}$$

$$\leftrightarrow J = \frac{q + rk^2}{1 - (a - bk)^2}$$

and thus the optimisation problem can be poised

$$\min_{k \in \mathbb{R}} \max_{(a,b) \in \Theta} \frac{q + rk^2}{1 - (a - bk)^2}$$

F DLQR solution derivation

We start with the cost function and system dynamics:

$$J = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k); \quad x_{k+1} = Ax_k + Bu_k$$

Introduce a symmetric matrix $P = P^T$ and rewrite J

$$J = x_0^T P x_0 - x_0^T P x_0 + \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k)$$

Assuming stability, we have $x_\infty = 0$, leading to

$$\begin{aligned} 0 - x_0^T P x_0 &= x_\infty^T P x_\infty - x_0^T P x_0 \\ &= \sum_{k=0}^{\infty}(x_{k+1}^T P x_{k+1} - x_k^T P x_k) \qquad \text{(a telescoping sum)} \end{aligned}$$

The cost therefore becomes

$$J = x_0^T P x_0 + \sum_{k=0}^{\infty}\left(x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} - x_k^T P x_k\right)$$

Substituting $x_{k+1} = Ax_k + Bu_k$ and gathering like terms

$$J = x_0^T P x_0 + \sum_{k=0}^{\infty}\left(x_k^T (Q - P) x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k)\right)$$

Expanding the latter quadratic term

$$\begin{aligned} (Ax_k + Bu_k)^T P (Ax_k + Bu_k) &= x_k^T A^T P A x_k + u_k^T B^T P B u_k + x_k^T A^T P B u_k + u_k^T B^T P A x_k \\ &= x_k^T A^T P A x_k + u_k^T B^T P B u_k + 2 x_k^T A^T P B u_k \end{aligned}$$

Thus, J becomes

$$J = x_0^T P x_0 + \sum_{k=0}^{\infty}\left(x_k^T (A^T P A - P + Q) x_k + u_k^T (R + B^T P B) u_k + 2 x_k^T A^T P B u_k\right)$$

The terms with dependence on u_k

$$u_k^T \underbrace{(R + B^T P B)}_{\tilde{R}} u_k + 2 x_k^T A^T P B u_k$$

can be rewritten by completing the square

$$\begin{aligned} &= (u_k^T + x_k^T A^T P B (\tilde{R}^{-1})^T) \tilde{R} (u_k + \tilde{R}^{-1} B^T P^T A x_k) - x_k^T A^T P B (\tilde{R}^{-1})^T \tilde{R} \tilde{R}^{-1} B^T P^T A x_k \\ &= (u_k^T + x_k^T A^T P B (\tilde{R}^{-1})^T) \tilde{R} (u_k + \tilde{R}^{-1} B^T P^T A x_k) - x_k^T (A^T P B \tilde{R}^{-1} B^T P^T A) x_k \end{aligned}$$

assuming $R = R^T \leftrightarrow (\tilde{R}^{-1})^T = \tilde{R}^{-1}$. Finally, substituting into the previous expression for J and gathering like terms, we get

$$J = x_0^T P x_0 + \sum_{k=0}^{\infty}\left(x_k^T (A^T P A - P + Q - A^T P B \tilde{R}^{-1} B^T P A) x_k + (u_k + K x_k)^T \tilde{R} (u_k + K x_k)\right)$$

where

$$\tilde{R} = (R + B^T P B); \quad K = \tilde{R}^{-1} B^T P A$$

The latter term in the sum is non-negative and minimised when set to zero

$$u_k = -K x_k$$

Similarly, the term involving x_k is eliminated by solving the Discrete Algebraic Riccati Equation (DARE)

$$A^T P A - P + Q - A^T P B (R + B^T P B)^{-1} B^T P A = 0$$

Thus, the infinite horizon cost is minimised using full state feedback, where K is set according to the above formulae.

G Riccati heuristic

Background

With the given state-space dynamics but no assumption regarding which controller is used, the DLQR cost function J can be expressed as

$$J = x_0^T P x_0 + \sum_{k=0}^{\infty} x_k^T \tilde{P} x_k + \sum_{k=0}^{\infty} (u_k + \tilde{K} x_k)^T \tilde{R} (u_k + \tilde{K} x_k)$$

where

$$\begin{aligned}
\tilde{P} &= (A^T P A - P + Q - A^T P B \tilde{R}^{-1} B^T P A) \\
\tilde{R} &= (R + B^T P B) \\
\tilde{K} &= (R + B^T P B)^{-1} B^T P A
\end{aligned}$$

and P is some symmetric positive-definite matrix. This form of the cost function is obtained as an intermediate step when deriving the solution of the standard infinite-horizon problem, shown previously. Do note that $\tilde{K}$ is not explicitly a controller feedback gain in this formula, it is some matrix determined by R,B,P,A.

First order approximation

Let's analyse local perturbations from a plant's nominal $K = \tilde{K}$. Letting P be set according to the DARE, and substituting $u_k = -(\tilde{K} + \Delta K) x_k$ gives

$$J(\tilde{K} + \Delta K) = x_0^T P x_0 + \sum_{k=0}^{\infty} x_k^T (\Delta K)^T \tilde{R} (\Delta K) x_k$$

Since this involves no first order terms, the first order approximation is

$$J(\tilde{K} + \Delta K) \approx x_0^T P x_0 \tag{55}$$

This result can also be deduced from first principles - since the controller $\tilde{K}$ is locally (and globally) optimal, then

$$\nabla J(K)|_{\tilde{K}} = 0$$

Therefore, when plant uncertainty is small enough s.t. ΔK is small and the first order approximation is valid, then the optimal controller is that which is designed for the plant with highest $x_0^T P x_0$. This leads to the *Riccati heuristic*

$$\begin{gathered}
\min_{K} \max_{(A,B) \in \Theta} J \\
\approx \\
K = (R + B^T P B)^{-1} B^T P A \\
\max_{P} \left(x_0^T P x_0 \right)
\end{gathered}$$

i.e. for each $A_i, B_i \in \Theta$

- calculate P_i which solves the associated DARE
- the cost associated with that A_i, B_i is $J_i = x_0^T P_i x_0$

and keeping track of these, record the A, B, P which maximises J, then set $K = (R + B^T P B)^{-1} B^T P A$ to get an approximately min-max optimal controller.

Second order approximation

A similar approach with the second-order approximation yields

$$J(\tilde{K}+\Delta K) \approx x_0^T \left(P + \sum_{k=0}^{\infty} (A-B\tilde{K})^{kT} (\Delta K)^T (R+B^T P B)(\Delta K)(A-B\tilde{K})^k \right) x_0$$

One may suspect that a larger $x_0^T P x_0$ indicates larger sensitivity to ΔK, so that the Riccati heuristic also holds from the perspective of second order approximations. However this does not hold in general. Analysing the infinite sum (which is second order in ΔK), one cannot in general deduce anything about whether or not this is large or small based on P.

Empirically, the contrary is more often true - the infinite sum term is usually larger with *smaller* $x_0^T P x_0$, indicating the plants which have a cheaper optimal cost are often more sensitive to ΔK. Therefore, the Riccati heuristic will not hold when the ΔK is large enough.

Demonstration of effectiveness

Despite its limitations, the effectiveness of the Riccati heuristic is often sufficient and can be much faster than solving a large LMI problem. When uncertainty is reasonably small, it may be used as a) a quick check for simultaneously stabilising K and b) a reasonable initial guess for the min-max optimal K.

Figure 14 demonstrates the effectiveness of the heuristic, using three systems $(A_1, B_1), (A_2, B_2), (A_3, B_3)$, which are random perturbations of the nominal system

$$A_0 = \begin{bmatrix} 0 & 0 & 0 & -0.048 \\ 1 & 0 & 0 & 0.676 \\ 0 & 1 & 0 & -2.21 \\ 0 & 0 & 1 & 2.6 \end{bmatrix}, \quad B_0 = \begin{bmatrix} -0.12 \\ 0.68 \\ -1.4 \\ 1 \end{bmatrix}$$

using random entries drawn from $\mathcal{N}(0, \sigma)$, with LQR costs $Q = \mathbb{I}, R = 1$.

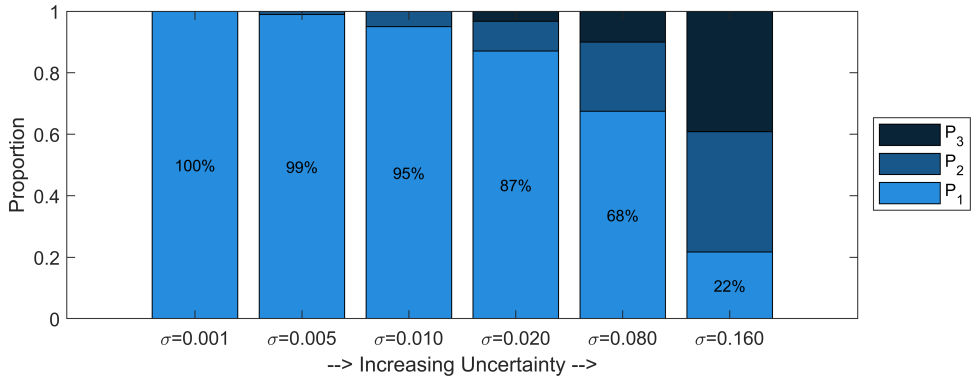


Figure 14: Performance of the Riccati heuristic. The text percentages on the bar charts indicate the number of times the heuristic held out of the $N = 100$ trials performed. The categories P_1, P_2, P_3 indicate the plant with 1st, 2nd, or 3rd highest trace(P) was the min-max optimal plant. The plot demonstrates how the heuristic is reliable when uncertainty σ is low, and performance degrades as uncertainty σ increases.

Figure 15 similarly demonstrates the effectiveness of the heuristic, using 256 perturbations $(A_1, B_1), ..., (A_{256}, B_{256})$ of the nominal system. To demonstrate the amount of uncertainty in the resulting perturbed systems, the poles are visualised in Figure 16. We see the Riccati heuristic tends to perform reasonably with modest uncertainty, though in general it cannot be relied on as the sole way of computing the min-max optimal controller.

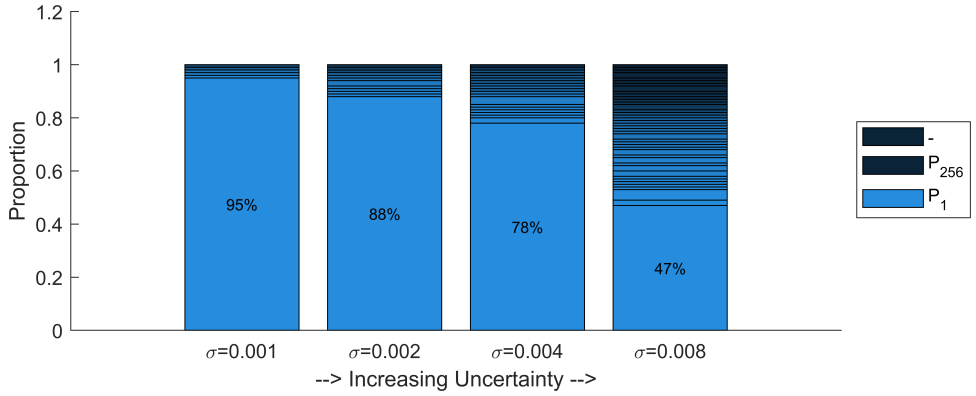


Figure 15: Performance of the Riccati heuristic. Again, the text percentages on the bar charts indicate the number of times the heuristic held out of the $N = 100$ trials performed. The categories $P_1, ..., P_{256}$ indicate the plant with 1st, …, or 256th highest trace(P) was the min-max optimal plant, while "$-$" indicates none of the 256 plants yielded a feasible controller.

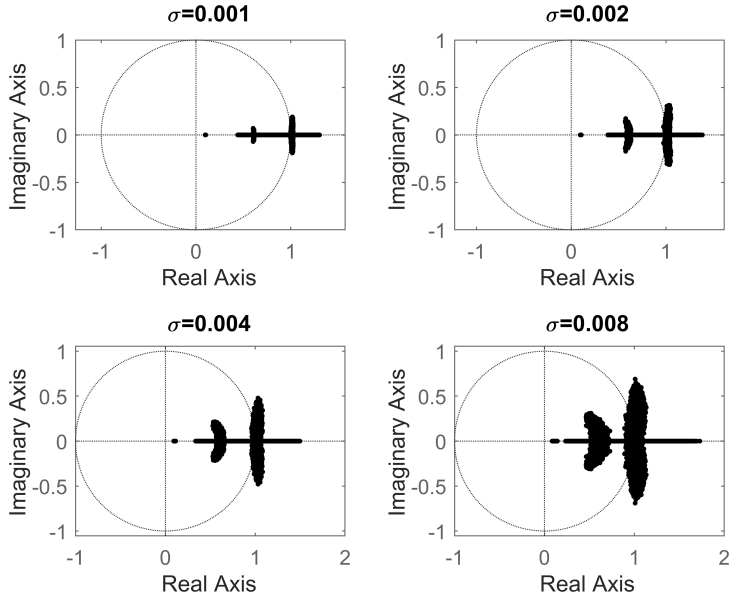


Figure 16: Possible pole locations after perturbing A_0 with $\sigma = 0.001, \ldots, 0.008$. This more clearly demonstrates how much the systems differed using the σ values in Figure 15.

H Alternative approaches to uncertainty set representation.

H.1 Alternative approaches to uncertainty set representation.

The first such approach involves decomposing the full polytope $\mathscr{A}\mathscr{B}$ into independent convex hulls $\mathscr{A}$ and $\mathscr{B}$, as considered in Oliveira 1999. This reduces the dimensionality of the convex hull computation by half; as a result, systems with up to eight states become tractable. The tradeoff is the conservatism introduced by the independence assumption. However, we still are able to use convex hull calculations to get a precise representation of $\mathscr{A}$ and $\mathscr{B}$.

The second alternative approach involves approximating the uncertainty region $\mathscr{A}\mathscr{B}$ with an ellipsoid-like polytope, bypassing convex hull computations entirely. This approach, referred to as LowResMVEE (low resolution minimum volume enclosing ellipsoid), is based on the observation from asymptotic theory that the confidence region approaches an ellipsoid as the number of datapoints $N \to \infty$. The details regarding the construction of the ellipsoid-like polytope can be found in the appendix. With this approach, we again get a small set of n vertex plants used to define and solve an LMI problem, however there is no conservativeness introduced by an independence assumption on A and B. Instead, the conservativeness is introduced by the overbounding of the confidence region by the ellipsoid-like polytope. The overbounding may be reduced by increasing the "resolution" of the polytope via a parameter m_d, however this also leads to a greater number of plants n, yielding a more complex LMI problem, again presenting another tradeoff.

Discussion of alternative uncertainty representations. The preferred method of representing the uncertainty region is via vertex plants from the Peaucelle convex hull $\mathscr{A}\mathscr{B}$; this maintains a precise representation yet simplifies the problem considerably. However, in the case where computing this convex hull is too costly, the other two approaches prevail. The choice between the two methods is application-specific.

The LowResMVEE approach allows the coupling between A and B to be maintained. Systems with highly coupled A and B with low uncertainty may benefit more from this approach, while mostly independent A and B would be best suited to the Oliveira-style approach in which precise representations of A and B can be maintained via convex hull computations.

The LowResMVEE approach is also the most scalable solution for very high-dimensional systems. With this approach, it always possible to generate a *cross-polytope*; as discussed in the appendix, a cross-polytope has $2n$ vertices in n-dimensional space, leading to a fixed number of $2n$ LMI constraints. This enables extension of the LMI synthesis procedure to beyond 8 state systems, at which point both convex hull based approaches would be intractable. However, the resulting region tends to be highly conservative due to overbounding, and it is very possible for the LMI problem to thus be infeasible, despite a simultaneously stabilising K existing for the true set of (A, B). In these cases, the mesh density parameter m_d may be increased to reduce the conservativeness, however in high dimensions this yields a polytope with a vanishingly large number of vertices n, so often only the cross-polytope generated with $m_d = 3$ is practical.

Finally, using all available plant samples captures the SPS confidence region precisely and is also available as another option, however controller synthesis quickly becomes intractable when there are a large number of plants. Since it suffices to stabilise and design for only the vertex plants of a convex polytope region, designing for every plant sample is more precise but unnecessary and inefficient.

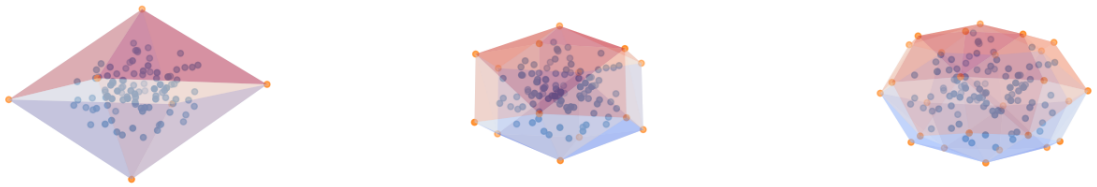
I LowResMVEE

Polytope Approximation of the SPS Confidence Region

Computing the convex hull of the sampled uncertainty set maintains an accurate representation of the true SPS confidence region, and significantly reduces the number of plants needing considered during controller synthesis, enabling tractable formulations of LMI and optimisation problems. However, the computational cost of convex hull algorithms, such as `qhull`, grows rapidly with the dimension of the data - the time complexity grows according to $\mathcal{O}(n^{\lfloor d/2 \rfloor})$ for n datapoints in d-dimensional space - making them impractical in higher dimensions.

To address this, we propose an alternative approach that scales better with dimensionality. Instead of the convex hull, we construct a convex polytope that approximates an ellipsoid enclosing the uncertainty region. This method is based on the observation from asymptotic system identification theory that the true confidence region becomes ellipsoidal as the number of data points tends to infinity. While this property does not strictly hold for finite-sample confidence regions, an ellipsoid still remains the most suitable approximation.

The end result is a systematic way of constructing a polytope approximation of the uncertainty region. Importantly, the method works in any n-dimensional setting, and the "resolution" of the approximation can be adjusted via the m_d parameter. This allows a tradeoff between a) a high number of polytope vertices, well approximating the uncertainty region but being a computationally expensive way of representing it, versus b) a low number of polytope vertices, only roughly approximating the uncertainty region but being a computationally efficient representation that scales well to high dimensional settings.



(a) (b) (c)

Figure 17: Demonstration of LowResMVEE in 3D. In (a), we show the simplest polytope enclosing the datapoints, generated using the minimum mesh density $m_d = 3$. This value of m_d always yields a cross polytope. In (b) and (c), we increase the mesh density parameter m_d to 4 and 5 respectively, yielding higher resolution polytopes. With more vertices, these more accurately characterise the uncertainty region, though this increases the computation requirements when solving the optimal control problem.

Implementation details

Overview.

Constructing such a polytope consists of the following steps:

1. Compute the Minimum Volume Enclosing Ellipsoid (MVEE) of the sampled points using Khachiyan's algorithm. Since this algorithm is approximate and may not yield a truly bounding ellipsoid, we

also apply post-processing to ensure all points are enclosed.

2. Generate a convex polytope with vertices on the MVEE surface. To do so, we start with a set of unit vectors computed via generalised spherical coordinates, with the tunable parameter m_d controlling the resolution. This gives a spherical mesh. Then, these vectors are mapped through the ellipsoid's linear transformation to obtain points on the ellipsoid boundary.
3. Scale the polytope so that it encloses the MVEE. As is, the polytope vertices are on the MVEE i.e. the MVEE encloses the polytope, whereas we require the polytope to enclose the MVEE, so that the polytope encloses the uncertainty region. This is achieved by applying an appropriate scaling factor to the ellipse.

Figure 18 visually illustrates this procedure.

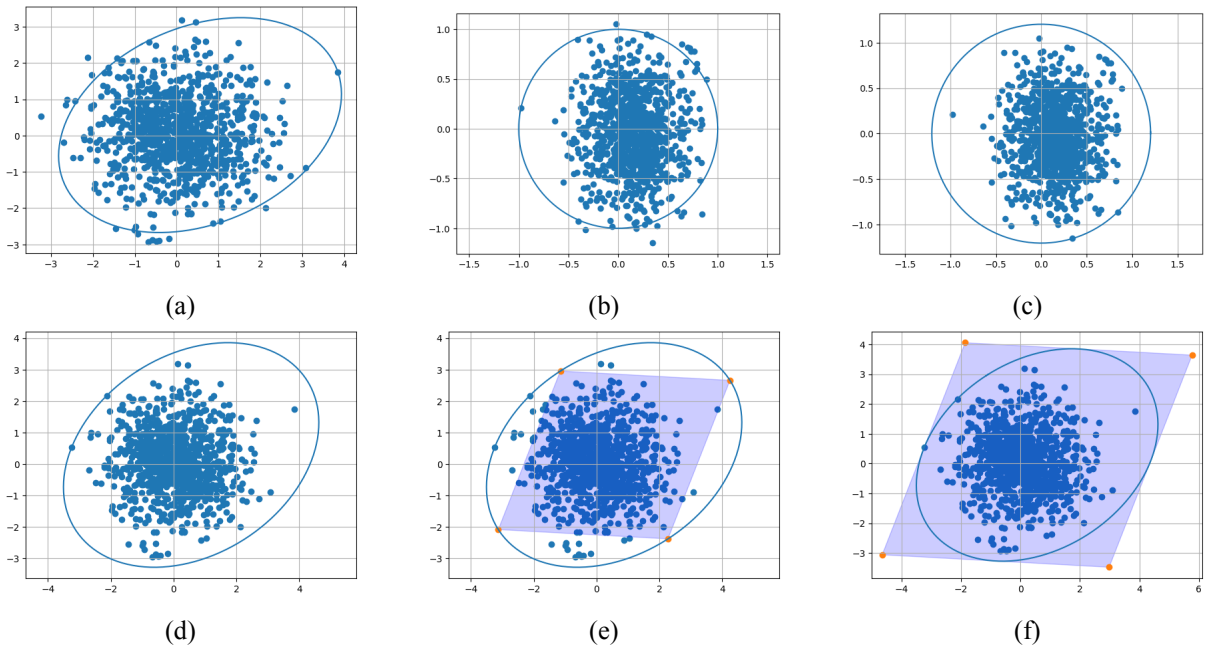


Figure 18: Demonstration of how a convex polytope enclosing the uncertainty region is computed. In (a), the MVEE is computed via Khachiyan's algorithm, yielding an approximate MVEE. The most important output of this stage is the transformation parameters A and c. In (b), a transformation is applied such that the ellipse and points are mapped back to the unit circle. In (c), we compute the maximum Euclidean distance of one of these transformed points; this determines how much the ellipse needs to be scaled to ensure it is bounding. In (d), we apply this scaling factor, demonstrating the ellipse is in fact bounding after the process. Finally, in (e) we generate the polytope with vertices on the ellipsoid boundary, and in (f) we apply the appropriate scaling factor so that the polytope encloses the MVEE.

Details.

The result of Khachaiyan's algorithm is parameters A, c defining the (approximate) MVEE

$$(x-c)^T A(x-c) \leq 1$$

Apply an eigendecomposition to get

$$A = PDP^T$$

The transformation mapping a point on the ellipsoid x_E to the corresponding point on the unit circle x_C

is

$$x_C = D^{1/2} P^T (x_E - c)$$

Applying the same transformation to the datapoints, we should get a set of points contained within the unit circle. If the MVEE is not bounding, we will get some outside the unit circle. Compute $\bar{r}$, the maximum Euclidean distance from the origin to a transformed datapoint. The crucial observation is that scaling the unit circle by a factor of $\bar{r}$ yields a circle enclosing the datapoints; similarly, scaling the ellipse by a factor of $\bar{r}$ ensures it encloses the datapoints. Thus, we can compute the correction factor $\bar{r}$ by which the ellipse must be scaled to ensure bounding.

The mesh vertices v_C are generated on the unit circle, then mapped to polytope vertices v_P using

$$v_P = (\kappa P D^{-1/2}) v_C + c$$

where $\kappa = \bar{r} \cdot s_f$. Without the scaling factor s_f, the polytope vertices would lie *on* the MVEE, i.e. the MVEE encloses the polytope, whereas we require the polytope to enclose the MVEE. This is the role of the precomputed scaling factor s_f.

Details regarding how the initial mesh vertices v_C and the scaling factor s_f are computed are deferred to the next section. Roughly speaking, the v_C are generated using generalised spherical coordinates, and s_f is computed by analysing the maximum radius sphere r_{max} that the polytope defined by v_C can enclose.

Note that in the explanation we restricted attention to the 2D case and referred to the unit circle, but the procedure is general and for 3D/nD, we'd just have the unit sphere/n-sphere.

Discussion

The main advantage of this approach is that it avoids convex hull computations, which quickly become infeasible as n and m_d increase.

The main disadvantage is how the number of vertices explodes rapidly in high dimensions. For an n-dimensional polytope generated with mesh density m_d, the number of vertices was found to satisfy the recurrence relation:

$$f(n, m_d) = f(n-1, m_d) \cdot (m_d - 2) + 2, \quad f(1, m_d) = 2$$

The following demonstrates how the mesh becomes incredibly complex as n increases and $m_d > 3$

n	$m_d = 3$	$m_d = 4$	$m_d = 5$
2	4	6	8
3	6	14	26
4	8	40	80
5	10	62	242
6	12	126	728
8	16	510	6560
10	20	2046	59048
12	24	8190	531 440
14	28	32766	4 782 968
16	32	131070	43 046 720

Fortunately, the cross polytope generated via $m_d = 3$ always remains feasible and avoids the exponential growth, scaling linearly with n. For high dimensions, we are restricted to using these simple cross-polytopes. These overbound the ellipsoid and are quite conservative, often leading to infeasible LMI problems despite a solution existing for the true SPS-identified plants. Nevertheless, this approach to characterising the uncertainty region remains suitable in some scenarios, for example when there is strong coupling between A and B and thus assuming independent A, B is too conservative. The approach also scales well with dimensionality n. Overall, this characterisation of the uncertainty region is often a practical choice and is thus included as an option in the optimal control solver.

Further details.

Computing the n-sphere mesh.

Just as polar and spherical coordinates can be used to describe 2D and 3D space, generalised spherical coordinates can be used to describe n-dimensional Euclidean space

$$\begin{aligned}
x_1 &= r\cos(\theta_1) \\
x_2 &= r\sin(\theta_1)\cos(\theta_2) \\
x_3 &= r\sin(\theta_1)\sin(\theta_2)\cos(\theta_3) \\
&\vdots \\
x_{n-1} &= r\sin(\theta_1)\cdots\sin(\theta_{n-2})\cos(\theta_{n-1}) \\
x_n &= r\sin(\theta_1)\cdots\sin(\theta_{n-2})\sin(\theta_{n-1})
\end{aligned}$$

where the angular coordinates θ_i range over

$$\begin{aligned}
\theta_1, \theta_2, \ldots, \theta_{n-2} &\in [0, \pi] \\
\theta_{n-1} &\in [0, 2\pi)
\end{aligned}$$

Consider m_d linearly spaced angles over $[0, \pi]$, and similarly for the $[0, 2\pi)$ interval

$$\begin{aligned}
\theta_1, \ldots, \theta_{n-2} &= \{0, 1, 2, \ldots, m_d - 1\} \cdot \frac{1}{m_d - 1}\pi \\
\theta_{n-1} &= \{0, 1, 2, \ldots, 2m_d - 3\} \cdot \frac{1}{m_d - 1}\pi
\end{aligned}$$

Evaluating over the combinations of these θ_i yields a discrete set of points in $\mathbb{R}^n$ approximating the n-sphere, with the parameter m_d determining the resolution of the approximation. The minimal setting $m_d = 3$ produces a cross polytope, the simplest approximation.

The end result is a straightforward, systematic way of generating a polytope approximating the n-sphere. The procedure works in any n-dimensional setting, and the polytope can be made more or less detailed via the m_d parameter.

Computing the scaling factor s_f.

Suppose we have used the procedure to generate a mesh in $\mathbb{R}^n$. To ensure that it encloses the unit n-sphere, we must apply a scaling factor s_f. The procedure for computing this is:

1. Determine the shape's facets.

2. For each facet, compute the normal vector, and thus compute the distance d from the origin to the facet.
   A general approach to computing the normal vector is as follows. Let the vertices of the facet be $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$. Form the matrix B of edge vectors:

$$B = \begin{bmatrix} (v_2 - v_1)^\top \\ (v_3 - v_2)^\top \\ \vdots \\ (v_n - v_{n-1})^\top \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}$$

   The rows of B span the hyperplane defined by the facet, while null(B) is the orthogonal space i.e. it is spanned by the facet's normal vector. Therefore, choose $n \in$ null(B) with $||n|| = 1$ to get the facet's unit normal vector.
3. Find the smallest such distance, d_{min}. The polytope can thus enclose an n-sphere of radius $r \leq d_{min}$.
4. Therefore, applying the scaling factor $s_f = 1/d_{min}$ to the polytope vertices ensures the polytope encloses the unit n-sphere.

An example demonstrating the effect of s_f is presented in Figure 18. From (e) to (f), the polytope is scaled by $s_f(n = 2, m_d = 3) \approx 1.7$, ensuring the polytope bounds the datapoints as demonstrated in (f).

J Simultaneously stabilising K is an NP-hard problem in general

Initial approach based on Lyapunov theory

Recall from Lyapunov theory that the system $x_{k+1} = Ax_k$ is stable if and only if there exists a $P > 0$ such that $A^T PA - P < 0$. Alternatively, the stability condition can be expressed as an LMI

$$\begin{bmatrix} P & PA \\ (PA)^T & P \end{bmatrix} > 0$$

which may be verified by applying a Schur complement with respect to block (1,1).

For a closed loop system under full state feedback, we have $x_{k+1} = (A + BK)x_k$, so similarly the stability condition is

$$\begin{bmatrix} P & P(A+BK) \\ (P(A+BK))^T & P \end{bmatrix} > 0$$

thus determining K via Lyapunov stability is a *bilinear matrix inequality* (BMI), since we have variables K and P appearing together in the PBK term. BMI's are not jointly convex and are NP-hard to solve, so we seek a similar approach which yields an LMI instead, as these can be solved efficiently, or concretely determined as infeasible.

Efficiency is particularly important in the context of finding a *simultaneously stabilising* K, where to ensure stability across the whole set of plants we have

$$\begin{bmatrix} P_i & P_i(A_i+B_iK) \\ (P_i(A_i+B_iK))^T & P_i \end{bmatrix} > 0$$

for all N plants, which quickly becomes infeasible to solve as a BMI.

Alternative LMI for stability

In order to reduce the problem to an LMI, first we restate a theorem from Oliveira 1999. The system $x_{k+1} = Ax_k$ is stable if and only if there exists $P = P^T > 0$ and some G such that the following LMI is feasible

$$\begin{bmatrix} P & A^T G^T \\ GA & G + G^T - P \end{bmatrix} > 0$$

An equivalent formulation which we will use is

$$\begin{bmatrix} P & AG \\ G^T A^T & G + G^T - P \end{bmatrix} > 0$$

which can be seen by using the Schur complement with respect to P for necessity, while multiplying by $T = \begin{bmatrix} I & -A \end{bmatrix}$ on the left and T' on the right shows sufficiency.

A closed-loop system yields

$$\begin{bmatrix} P & (A+BK)G \\ * & G + G^T - P \end{bmatrix} > 0$$

which is a BMI, but substituting $L = KG$ yields

$$\begin{bmatrix} P & AG + BL \\ * & G + G^T - P \end{bmatrix} > 0$$

which is now an LMI over P, G, L.

Finally, with a set of plants, we can solve a set of LMI's

$$\begin{bmatrix} P_i & A_iG + B_iL \\ * & G + G^T - P_i \end{bmatrix} > 0$$

to yield a simultaneously stabilising controller $K = LG^{-1}$.

Polytope uncertainty

The outlined approach involves forming an LMI ensuring the stability of each plant, which quickly becomes computationally expensive. As discussed in the report, we instead compute the convex polytope(s) describing the uncertainty, with each vertex of the polytope(s) defining an LMI constraint - Oliveira 1999 and Peaucelle 2000 show that this approach ensures stability for all plants inside the uncertainty region. This approach is more efficient and discussed in detail in the main report.

K G_0 and H_0 with an LQG controller

The plant dynamics are:

$$G(s) = C(sI - A)^{-1}B + D$$

The observer dynamics are:

$$\hat{x}(s) = H(s)\begin{bmatrix} u \\ y \end{bmatrix}$$

where

$$\begin{aligned} H(s) &= (sI - (A - LC))^{-1}\begin{bmatrix} B - LD & L \end{bmatrix} \\ &\doteq \begin{bmatrix} H_1(s) & H_2(s) \end{bmatrix} \end{aligned}$$

The input to the plant is:

$$u = r - K\hat{x} + N(s)n$$

$\rightarrow \ldots$ the open loop transfer functions are:

$$G_0(s) = (1 + \tilde{G}(s)KH_2(s))^{-1}\tilde{G}(s)$$

$$N_0(s) = (1 + \tilde{G}(s)KH_2(s))^{-1}N(s)$$

where

$$\tilde{G}(s) = G(s)(1 + KH_1(s))^{-1}$$

L Performance Benchmarks: ConvexHull vs LowResMVEE

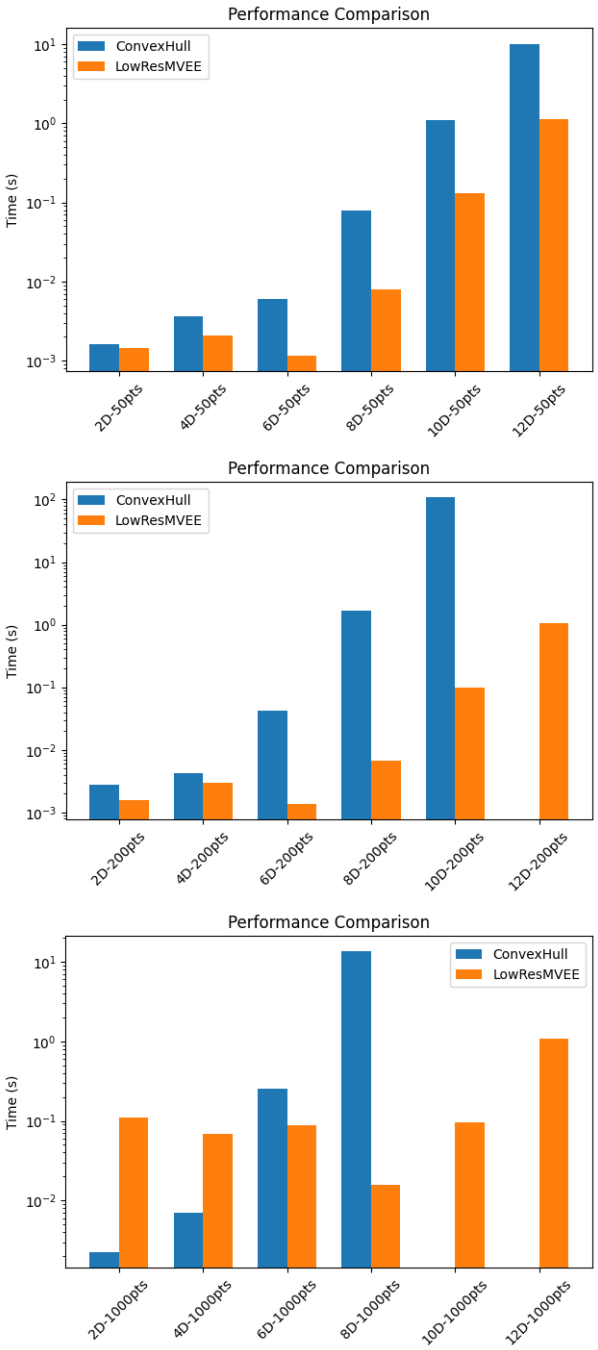


Figure 19: Performance benchmarks for the LowResMVEE algorithm. Compared to ConvexHull, the algorithm scales well with both dimensionality and the number of points. The speedup is most significant in 8D+ and when there is a large number of datapoints.

M Search Algorithms Benchmarks

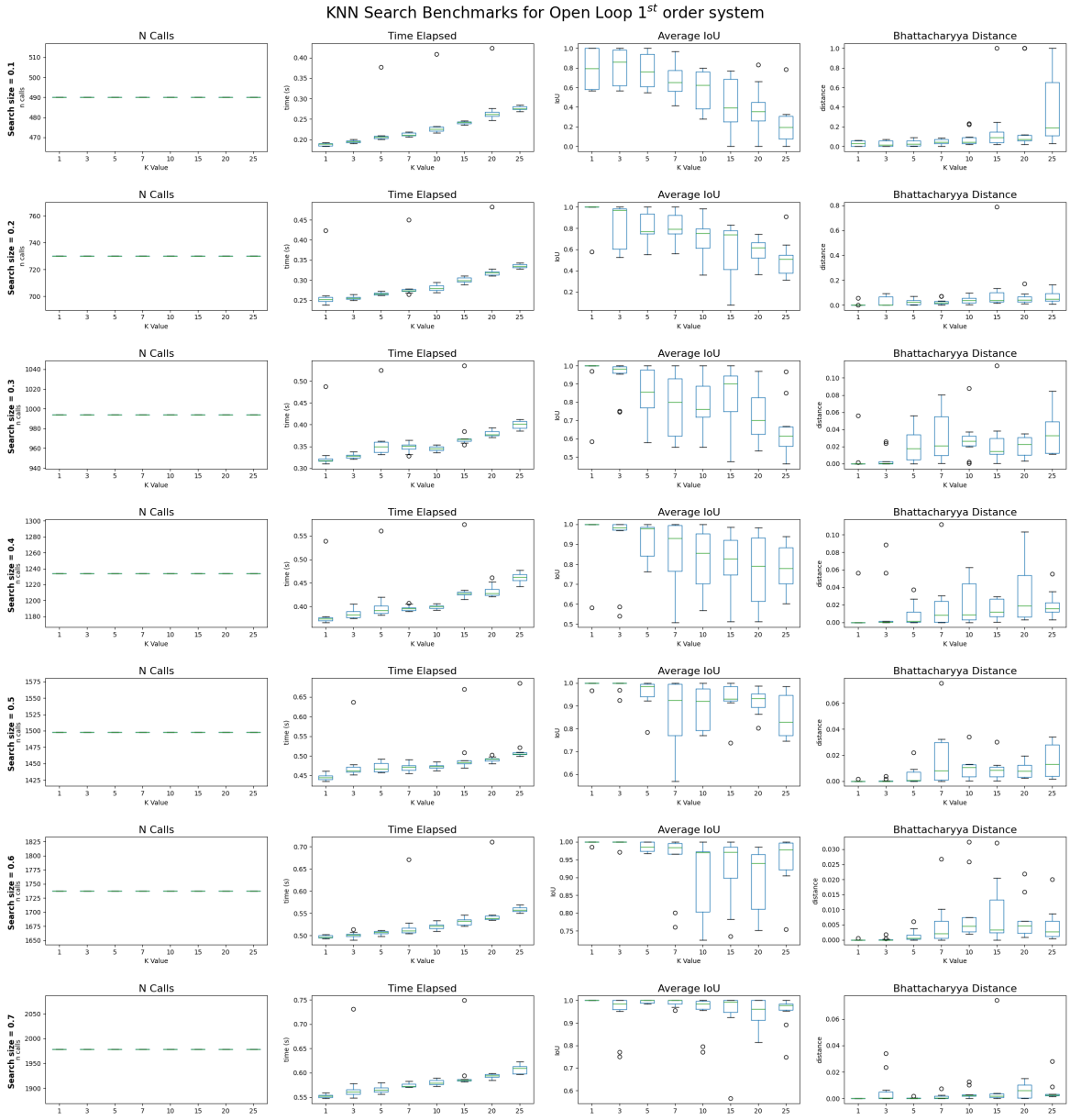


Figure 20: First Order Open Loop Benchmarks for KNN Search

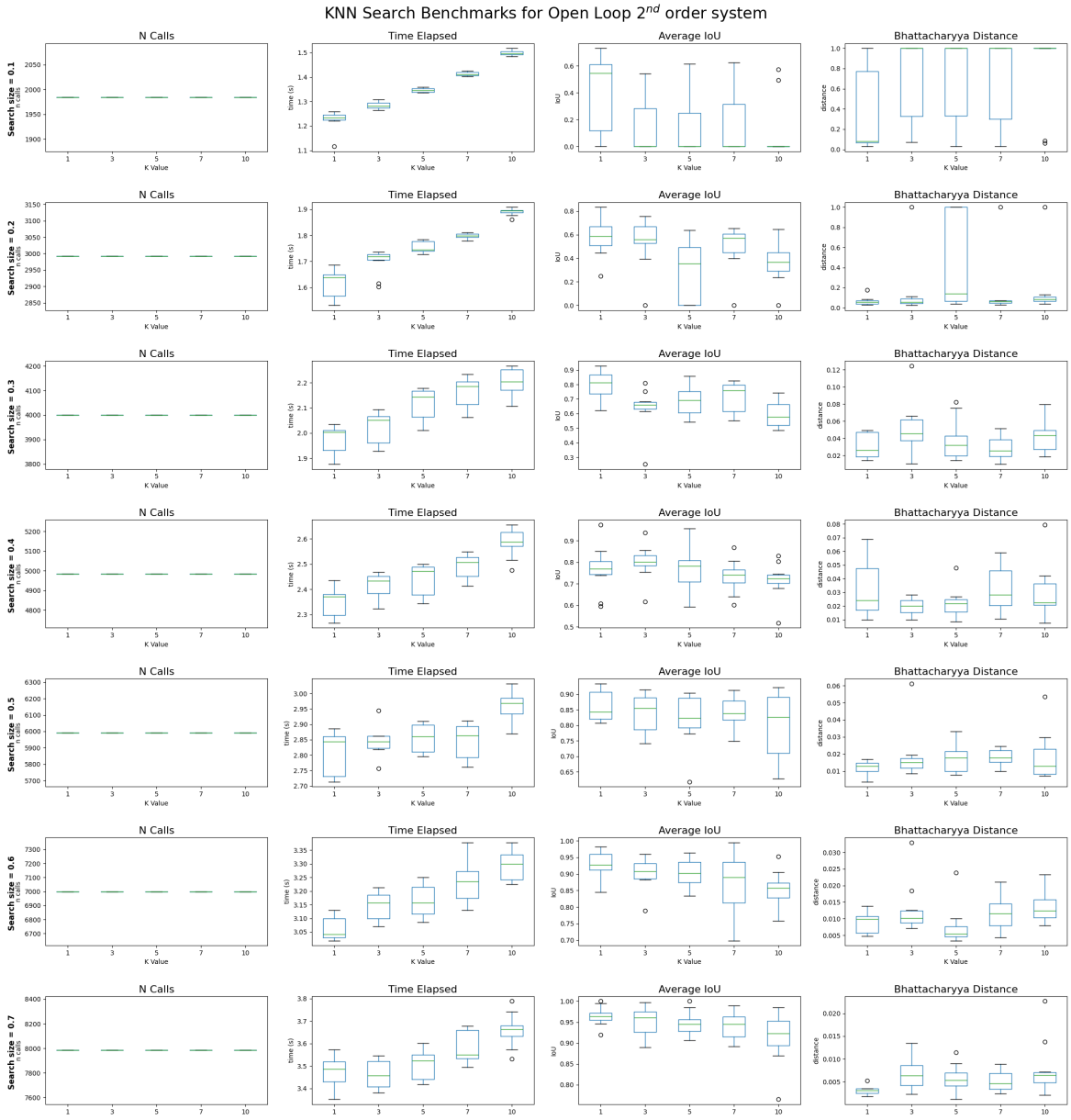


Figure 21: Second Order Open Loop Benchmarks for KNN Search (with coarser grid of n=20)

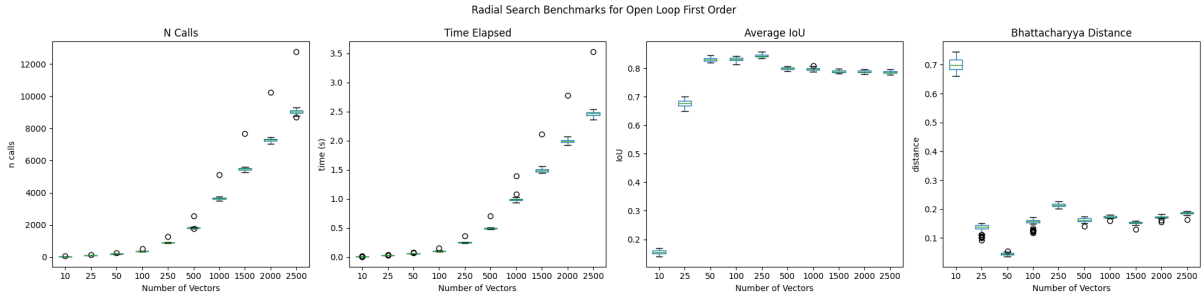


Figure 22: First Order Open Loop Benchmarks for Radial Search

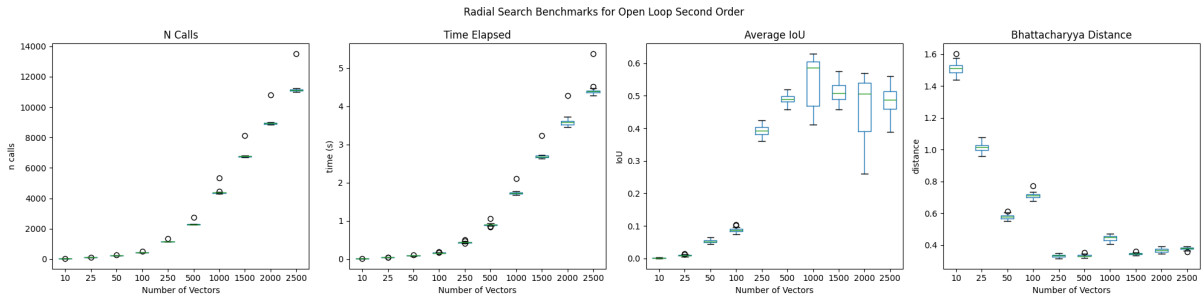


Figure 23: Second Order Open Loop Benchmarks for Radial Search

N Full Modular System Architecture

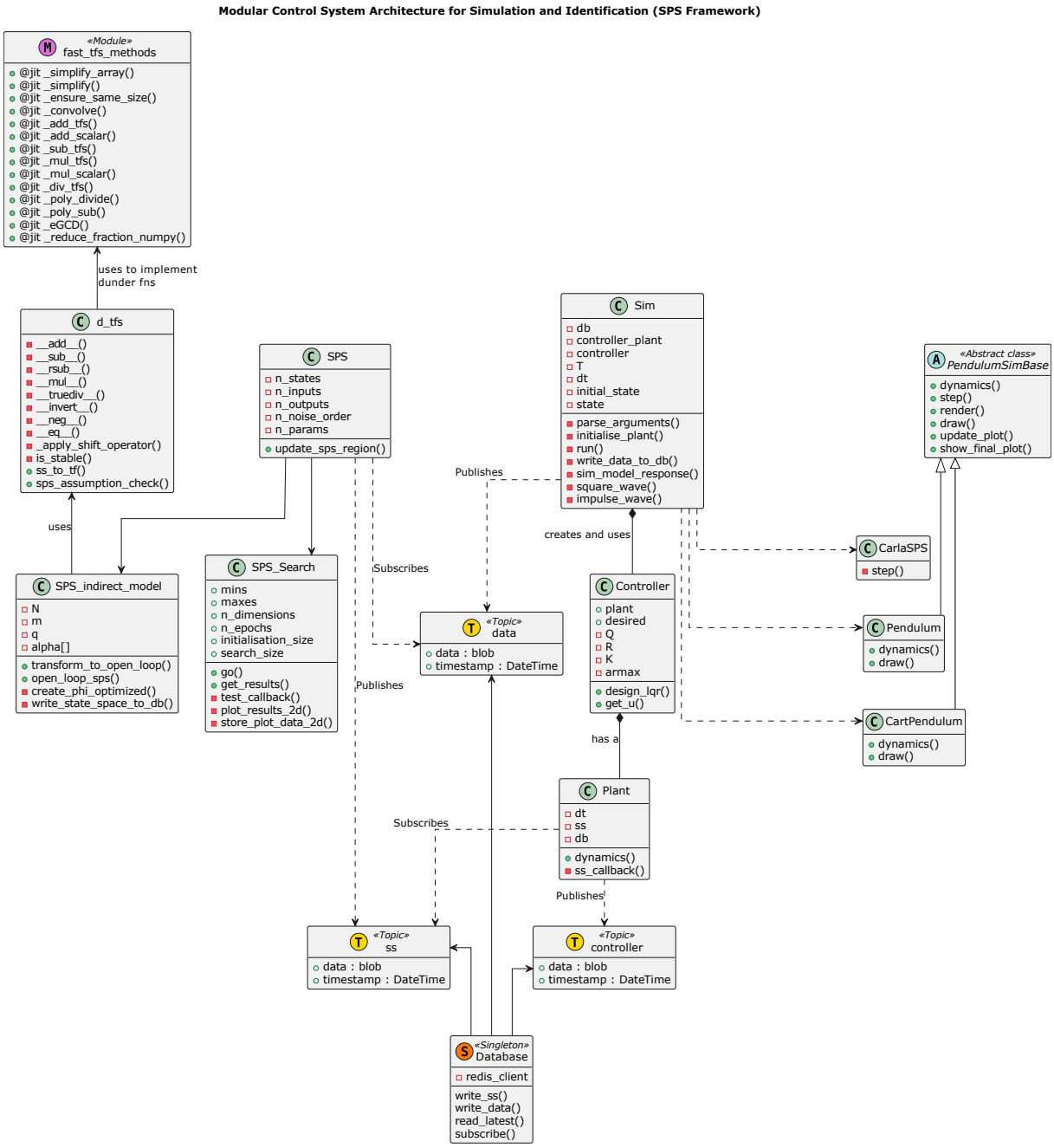


Figure 24: Modular Control System Archetecture