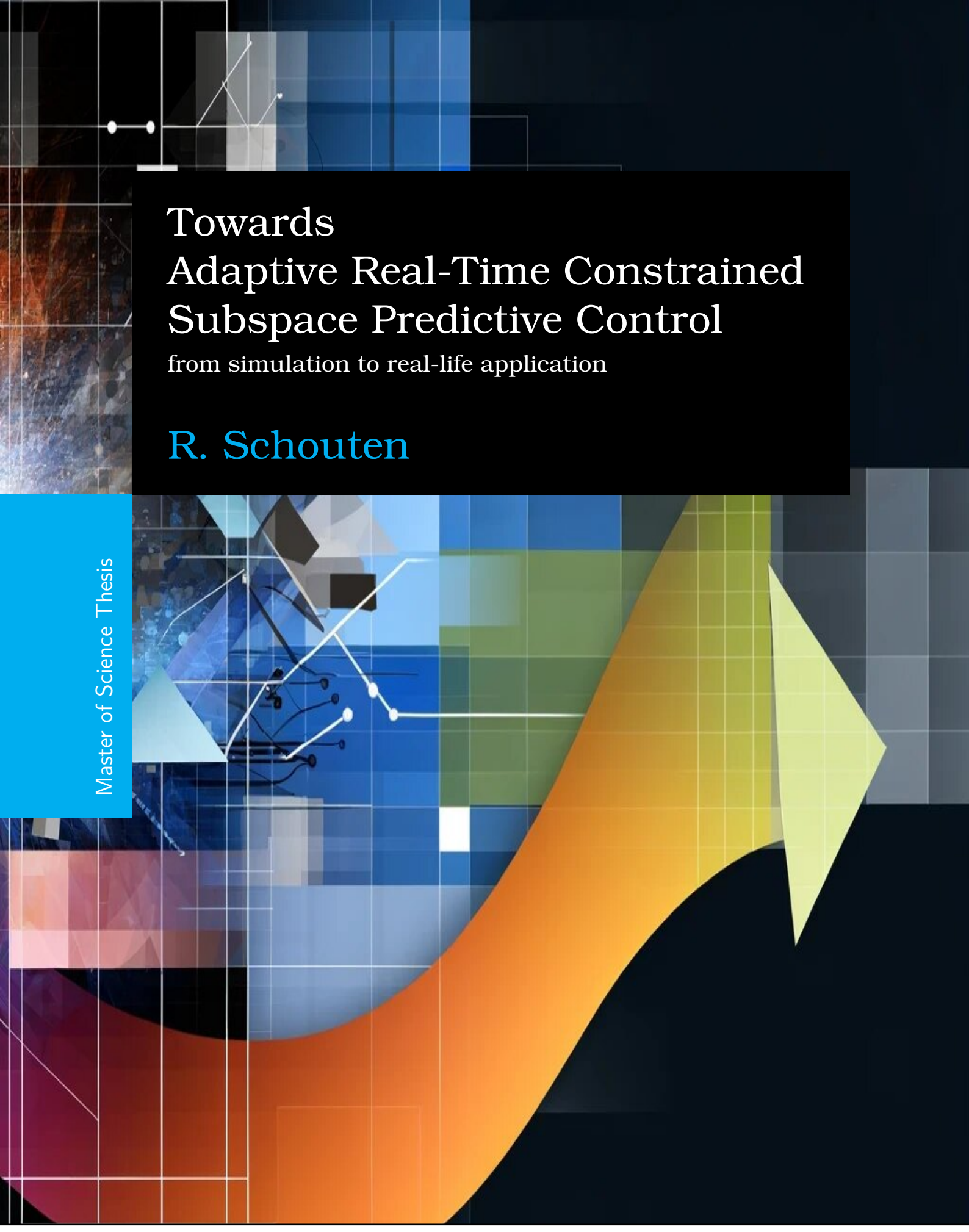


Towards Adaptive Real-Time Constrained Subspace Predictive Control

from simulation to real-life application

R. Schouten

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Towards Adaptive Real-Time Constrained Subspace Predictive Control

from simulation to real-life application

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

R. Schouten

April 12, 2025

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical Engineering (ME) for acceptance a thesis entitled

Towards
Adaptive Real-Time Constrained Subspace Predictive Control

by

R. Schouten

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: April 12, 2025

Supervisor(s):

prof.dr.ir. J.W. van Wingerden

ir. R.T.O. Dinkla

Reader(s):

dr. M.A. Zagorowska

ir. A. Ministeru

Abstract

Data-Driven Predictive Control (DDPC) has emerged as a promising alternative to Model Based Control (MBC), enabling direct control using Input-Output (I/O) data without requiring explicit model identification. This thesis advances DDPC by bridging the gap between theoretical developments and real-world implementation. The study focuses on four algorithmic variants: Subspace Predictive Control (SPC), Closed Loop Subspace Predictive Control (CL SPC), Recursive Closed Loop Subspace Predictive Control (R-CL SPC), and Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC), to enhance adaptability and ensure real-time feasibility.

A key insight in computational efficiency is the R-CL SPC algorithm, which updates system parameters online using recursive least squares estimation combined with Givens rotations. This reduces computational complexity by avoiding large-scale matrix inversions at each time step. Additionally, the CR-CL SPC variant introduces constraint handling via a Quadratic Programming (QP) solver, enabling input and output constraints.

These algorithms' performances were evaluated through simulation studies and real-time experiments on a piezo-actuated beam setup, where the control objective was to suppress the first two natural vibration modes. The R-CL SPC algorithm demonstrated a strong balance between computational efficiency and control performance, achieving execution times below 0.2 ms while maintaining effective vibration damping. Meanwhile, though at a higher computational cost, CR-CL SPC validated constraint enforcement capabilities.

This thesis demonstrates that adaptive SPC algorithms can be successfully implemented on real-world hardware. The results contribute to the growing knowledge on direct DDPC strategies and provide a foundation for their broader application in real-time, constrained control systems.

Acknowledgements

First of all, I would like to express my sincere gratitude to my daily supervisor, ir. R.T.O. Dinkla, for the valuable weekly meetings, extensive feedback, and guidance throughout this project. I have learned a great deal from our discussions and from your expertise in data-driven predictive control. I also want to thank my main supervisor, prof.dr.ir. J.W. van Wingerden, for his time and support. Additionally, I am grateful to dr. M.A. Zagorowska and ir. A. Ministeru for reading through my master thesis. Furthermore, I want to thank Luka, Eke, and Kenni (or Bram) for giving me feedback on my literature review and final thesis. I also acknowledge the use of OpenAI's ChatGPT for creating the front page figure, assistance in rewriting sentences, and providing support with coding during the development of this thesis.

Beyond academia, I would like to sincerely thank my girlfriend, Luka, for her unwavering support and patience. Despite her lack of understanding of control theory, she always made an effort to understand my work. More importantly, her help with planning and keeping me on track has been invaluable. Already working full-time, it was not always easy for her to deal with my schedule, yet she continued to support me through it all.

Finally, I want to thank my parents for their unconditional support and for always believing in me, even when my studies took a little longer than expected, and I wasn't home as much as they would have liked. Their encouragement and patience have meant a lot. I truly appreciate everything they have done for me.

Delft, University of Technology — R. Schouten
April 12, 2025

Table of Contents

Chapter 1

Introduction

This first chapter introduces Data-Driven Predictive Control (DDPC) and emphasizes its importance, obstacles, and applications. First, it highlights DDPC's contribution to the field of control systems, how it distinguishes itself by employing data directly for control, and how it moves beyond conventional Model Based Control (MBC). Afterward, the focus shifts to the challenges of implementing DDPC, aiming to provide a rounded perspective on its advantages and improvements. Lastly, DDPC's application across different fields shows the potential and what has already been done. A research question with multiple subquestions is formulated based on these challenges, potential state-of-the-art solutions, and applications.

1-1 Introduction to data-driven control

One of the most substantial steps towards what is currently known as MBC was taken due to Kalman's introduction of the parametric state-space model and optimal control in 1960 [1, 2]. With MBC, the first step of the process is modeling or identifying the plant into a parametric representation (System Identification (SysID)) and then designing a (optimal) control policy accordingly. In the upper part of Figure 1-1, the architecture of the MBC method is shown.

Identifying the plant or model can be approached in several ways. For example, by applying physical laws *(first principles modeling)*, by using observed or gathered Input-Output (I/O) data *(black box modeling)*, or through a combination of both *(grey box modeling)* [3]. The latter two methods are considered 'data-driven', due to their reliance on the path from I/O data to a resulting control policy.

Currently, several data-driven solutions exist to perform SysID, such as Prediction Error Method (PEM), Neural Network (NN), and Subspace Identification Methods (SIM). These methods will be further elaborated on in subsection 1-2-1. These algorithms help to create an accurate model but have some shortcomings, such as high computation times (especially for the NN and PEM methods) [4] or not being able to manage nonlinear systems (PEM and SIM) [5].

In MBC, a significant challenge arises when aiming to model as accurately as possible at a high model order. Although a higher-order model could yield greater accuracy, a practical need often arises to reduce the order or simplify the model for designing a feasible control system [6]. Higher-order control systems are not always suitable in practice for multiple reasons, such as high costs and poor reliability [7]. In contrast, designing a controller based on an inaccurate model may lead to poor performance or even instability in the resulting Closed Loop (CL) system [8].

Since there is an increase in the availability of data, storage capacity, and computational power, new methodologies have arisen that are mainly designed to manage large-scale data. Consequently, technological advancements have transformed the field of control theory, leading to a shift toward a more data-driven approach [3].

These technological advances eventually led to the research of less model-dependent, more direct control approaches. The more direct approach is shown in the lower part of Figure 1-1. The direct approach eliminates the explicit model creation step and achieves a control policy directly from the I/O data. This type of control allows for optimal control of complex networks with minimal knowledge of their dynamics, even with noisy data and in nonlinear environments [9]. Examples of direct DDPC methods include Subspace Predictive Control (SPC), a framework inspired by subspace identification techniques and introduced by [10], as well as Data-Enabled Predictive Control (DeePC), which is based on behavioral systems theory and formulated within a receding horizon optimal control framework, as presented in [11].

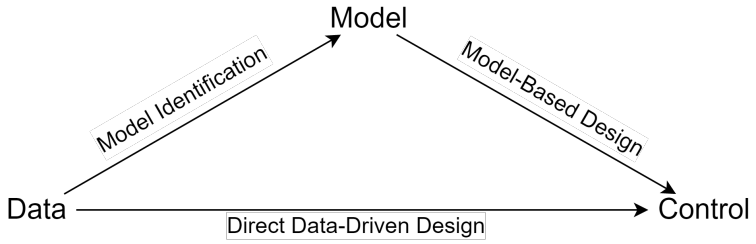


Figure 1-1: Architecture of MBC theory on the top, where the first step is to identify the system and then control is applied, and on the bottom of direct DDPC where receding horizon optimal control is achieved directly from I/O data, inspired by [3].

The term Data-Driven Predictive Control (DDPC) may be ambiguous as it encompasses all data-driven predictive control methods. A distinction is made between methods that obtain an explicit parametric model, which is referred to as an indirect method, and those that do not require such a model, which are direct methods.

Each control method, whether indirect or direct, has its advantages and disadvantages in practice. Indirect methods are mostly superior when accurate models are available because systematic design and analysis tools are widely available. At the same time, direct methods perform better when the plant models are unavailable or too complex to model. However, they have less systematic designing procedures and means of analysis available [8].

Elokda et al. [12] emphasizes the role of model accuracy in control performance. Through a quadcopter simulation, they demonstrate how the model mismatch between the assumed and actual model, caused, for example, by linearization around an incorrect operating point or by the limitations of modeling complex dynamics, can negatively impact control. Their

results show that the direct DDPC approach, specifically the DeePC algorithm, exhibits a notable degree of robustness to such model mismatches. In contrast, the Model Predictive Control (MPC) method, which relies on an explicit model, suffers from significant deviations when the model is inaccurate.

1-2 Indirect Data-Driven Predictive Control

This section presents the indirect methods of DDPC, which involve a two-step process consisting of system identification followed by optimal control. An example of such an approach is SPC, which can be interpreted as a combination of a subspace identification method—specifically SIM (e.g., N4SID [13])—and receding horizon optimal control. Together, these components form a DDPC strategy. For clarity, the components of the indirect method are discussed separately in this section, with the aim of providing an overview of various system identification techniques and optimal control strategies.

1-2-1 Identification Methods

As explained in section 1-1, indirect DDPC entails an identification step, which can be executed using different methodologies like PEM, NN, and SIM. This subsection elaborates on the strengths and weaknesses of these different identification methods.

Prediction Error Method

The first method is the Prediction Error Method (PEM), a widely used system identification approach. It minimizes the error between the measured and predicted outputs of the (to be) identified model [14]. PEM has advantages due to its flexibility in handling a lot of different model structures and its robustness when dealing with noisy data. However, it can be computationally intense, especially for large and complex systems [15]. Due to the non-convex optimization problem, the performance highly depends on the initial guess of the model parameters, meaning some model characteristics must be known to converge to the best local minimum [16].

Neural Networks

Another identification method is a Neural Network (NN), which is a powerful tool for modeling complex nonlinear systems due to its ability to approximate any continuous function by adding multiple (nonlinear) functions [17]. The main strength of a NN is the adaptability to various data types and structures and the capacity to capture nonlinear relationships without prior knowledge about the system's dynamics [18]. However, previous knowledge could improve training results [19]. On the downside, a NN is very delicate and can be prone to overfitting and underfitting, especially with limited training data [20]. Furthermore, training a NN requires careful selection of architecture, learning rate, and other hyperparameters, which can be a non-trivial and time-consuming task [21]. Lastly, creating a NN is a computationally heavy task and certainly unsuitable when it must be done at every time instant [16, 22].

Subspace Identification Methods

The third identification method involves Subspace Identification Methods (SIM), such as Numerical Algorithms for Subspace State Space System Identification (N4SID) [13] and Multivariable Output Error State Space (MOESP) [23], which offer an efficient way to identify state-space models from I/O data. SIM is particularly beneficial for complex but linear systems [24] and can provide parametric models that are easily usable for conventional (optimal) control design, such as Linear Quadratic Regulator (LQR) and MPC. The most essential advantage of SIM is the ability to use large data sets with less computation time [25]. However, the accuracy of SIM is sensitive to the selection of the subspace dimension and may be suboptimal for systems characterized by strong nonlinear behavior [26].

1-2-2 Optimal Control Methods

After identifying the parametric model, conventional optimal control is applied to complete indirect DDPC. Two widely adopted optimal control strategies are LQR and MPC, both aiming to enhance control performance, though they differ in their methodologies. LQR operates over an infinite horizon with a fixed control law, whereas MPC uses a receding horizon strategy, solving an optimization problem at each time step.

Linear Quadratic Regulator

The first optimal control method is LQR, which is an infinite-horizon control method [27] that minimizes a quadratic cost function balancing state deviation and control effort [28]. It is computationally efficient and provides robust performance against model uncertainties and disturbances [29]. However, its primary limitation is its restriction to linear systems [30].

Model Predictive Control

Another optimal control method is MPC, which optimizes control actions over a finite receding horizon [31], allowing it to handle constraints on inputs, states, and outputs [32]. This method minimizes a cost function similar to LQR but incorporates a terminal cost to ensure stability [33]. Unlike LQR, MPC is adaptable to both linear and nonlinear systems [34, 35]. However, it remains computationally demanding, especially for nonlinear systems, making real-time implementation challenging [36]. Additionally, its performance depends on the accuracy of the identified model [37].

1-3 Direct Data-Driven Predictive Control

Having addressed indirect methods in the previous section, this section now turns to direct methods. As explained in section 1-1, direct DDPC skips traditional SysID by directly utilizing system I/O data for control action [38]. This method, with methodologies like SPC and DeePC, focuses on predicting and optimizing control inputs and outputs without an explicit parametric model. This bypass of the parametric model allows it to enhance

adaptability to nonlinear or time-varying environments [39]. The benefit of merging this identification step with an optimal control method is that optimizing for the best model, state, and control policy at every timestep is possible [11]. This makes it a highly promising approach for real-time and adaptive control applications. The remaining part of this section explains the two methodologies for direct DDPC, which are SPC and DeePC and their CL variants.

1-3-1 Subspace Predictive Control (SPC)

The first direct DDPC method is Subspace Predictive Control, which integrates concepts from SIM, specifically N4SID [13], and a receding horizon optimal control method (MPC). As previously explained, direct DDPC relies solely on I/O data without explicitly identifying a parametric model of the system dynamics.

Favoreel introduced SPC and provided his algorithm [10]. One could argue that this version of SPC is indirect, as it implicitly estimates key system matrices, such as the observability matrix Γ, the reversed extended controllability matrix $\mathcal{K}$, and the block-Toeplitz matrix $H_{(B,D)}$. However, since these matrices are not explicitly parameterized, but rather estimated as composite terms—namely $\hat{\Gamma\mathcal{K}}$ and $\hat{H_{(B,D)}}$—which do not necessarily conform to a parametric state-space model, SPC is classified as a direct method.

A key limitation of Favoreel's [10] algorithm is the lack of explicit constraints on inputs, states, and outputs—features that are typically incorporated in a MPC scheme. Instead of relying on an analytical optimal solution as done in this paper, introducing an optimization-based approach could enable the integration of such constraints within SPC.

Furthermore, SPC assumes that the collected data is Persistance of Excitation (PE), ensuring that it contains sufficient information to capture the full system dynamics [40]. A data set is said to be *PE of order N* if the data matrix constructed from input-output pairs over a time span of N steps has full row rank, allowing for the identification of the system's dynamics up to that order. The effectiveness of SPC thus heavily depends on the quality and richness of the data used, as inadequate excitation could lead to incomplete system representation and suboptimal control performance [41].

Closed Loop Subspace Predictive Control

A different version of the standard SPC algorithm is the Closed Loop Subspace Predictive Control (CL SPC) algorithm. A key advantage of CL SPC as described by Dong et al. [42] is its efficient implementation, which involves estimating predictor-form Markov parameters and using them to compute the innovation-form Markov matrices required efficiently. The one-step-ahead predictor also allows CL SPC to avoid problems associated with CL identification bias, making it quick to implement and effectively manage dynamic systems with possible real-time adaptability requirements [43].

1-3-2 Data-Enabled Predictive Control (DeePC)

Another direct method is Data-Enabled Predictive Control, which takes a more direct approach than SPC by utilizing I/O data to compute control actions without explicitly modeling

system dynamics (i.e., $\Gamma\mathcal{K}$ and $H_{(B,D)}$). Instead, DeePC relies solely on the slack variable g to determine control inputs. The algorithm is based on behavioral systems theory, which approaches dynamic systems without requiring a predefined parametric model [11].

A key advantage of DeePC is its ability to incorporate input and output constraints naturally within its optimization framework, which also solves for the best possible system. As shown in Equation 1-1, the optimization problem is formulated to minimize tracking error and control effort while ensuring feasibility constraints on the system outputs and inputs.

$$\begin{aligned} \underset{g,u,y}{\text{minimize}} \quad & \textstyle\sum_{k=\hat{i}_p}^{\hat{i}_p+f} \left(\|y_k - (r_f)_k\|_Q^2 + \|u_k\|_R^2 \right) \\ \text{subject to} \quad & \begin{pmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,f,\bar{N}} \\ Y_{i_p,f,\bar{N}} \end{pmatrix} g = \begin{pmatrix} U_{\hat{i},p,1} \\ Y_{\hat{i},p,1} \\ U_{\hat{i}_p,f,1} \\ Y_{\hat{i}_p,f,1} \end{pmatrix}, \\ & u_k \in \mathcal{U}, \forall k \in \{\hat{i}_p, \ldots, \hat{i}_p + f\}, \\ & y_k \in \mathcal{Y}, \forall k \in \{\hat{i}_p, \ldots, \hat{i}_p + f\}. \end{aligned} \tag{1-1}$$

Similar to SPC, DeePC relies on sufficiently PE data to ensure that the collected I/O data fully captures the system dynamics [11]. This requirement is essential for the optimization problem to generate effective control inputs.

Despite its advantages, DeePC can be computationally expensive since the optimization problem must be solved with a Quadratic Programming (QP) solver at each time step, and there are more decision variables [44].

Equivalence SPC and DeePC

In Fiedler et al. [45] and van Wingerden et al. [46], a relationship and equivalence between SPC and DeePC is found. The investigation of Fiedler et al. [45] reveals that at their core, SPC and DeePC employ the same implicit linear least squares techniques to inform predictive control. The research of van Wingerden et al. [46] shows that DeePC, when augmented with Instrumental Variables (IV), shares a direct equivalence with SPC. Although these investigations highlight apparent methodological differences, they also reveal a fundamental equivalence in how both approaches utilize data to inform control decisions.

Closed Loop-Data-Enabled Predictive Control

The analysis of CL-DeePC by Dinkla et al. [47] is an extension to the original DeePC algorithm. The usage of sequential one-step-ahead predictors gives a possibility to use DeePC in CL. The significant advantage of CL-DeePC is that it avoids problems associated with CL identification bias [48].

1-4 Challenges of direct data-driven control

Several challenges arise when looking at the previously described direct DDPC methods. This section elaborates on these challenges and potential solutions, which include dealing

with nonlinear or time-varying systems, ensuring persistency of excitation, noise mitigation, and real-time feasibility.

1-4-1 Dealing with nonlinear or time-varying systems

One challenge related to direct DDPC is dealing with nonlinear or time-varying systems. Many DDPC methods in literature construct block Hankel data matrices $Y_{i_p,f,\bar{N}}$, $U_{i_p,f,\bar{N}}$, $U_{i,p,\bar{N}}$, $Y_{i,p,\bar{N}}$ offline and then use them to control the system online (see Figure 1-2). To improve robustness in time-varying or strongly nonlinear systems, Wegner [49] introduced the idea of using adaptive Hankel matrices. An example timeline of the data used for adaptive Hankel data matrices is given in Figure 1-3.

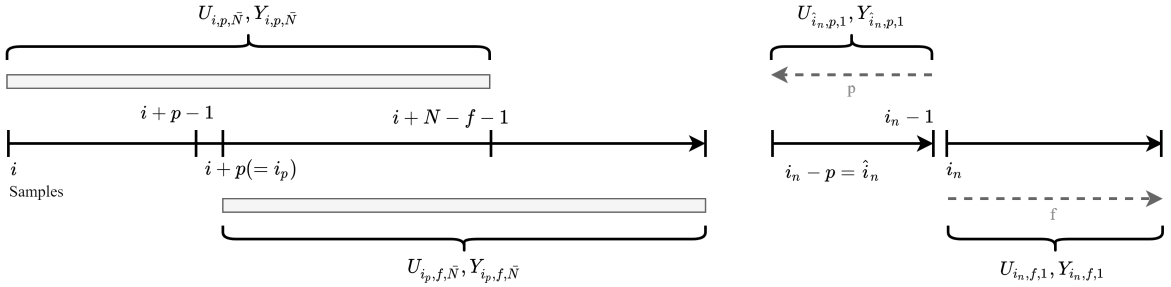


Figure 1-2: Visualisation of data of non-adaptive method, where the data on the left is computed offline, and the data on the right online, inspired by [46]

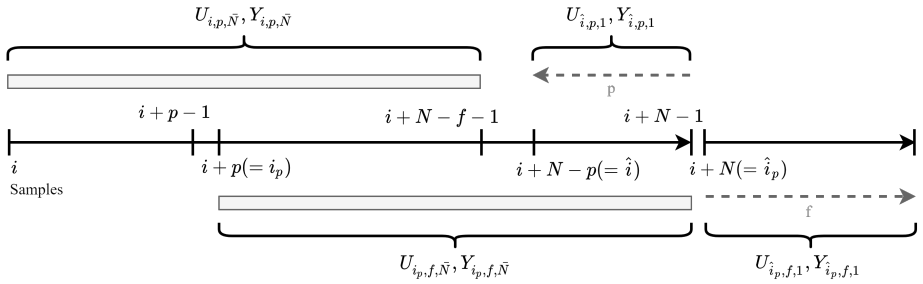


Figure 1-3: Visualisation of data of adaptive method where all data is utilized online, inspired by [46]

To address the inefficiencies inherent in the use of adaptive Hankel matrices, van der Veen [50] proposes a potential solution. This research introduces a recursive least squares scheme to update the SysID implicitly. This recursive CL SPC method is initiated with the potential to enhance the model's fault tolerance. The methodology shows promising results in overcoming the non-linearities and time-varying dynamics of, in this case, wind turbines by recursively updating the control strategy using real-time data to reflect the wind turbine's current state.

1-4-2 Ensuring persistency of excitation

One of the most significant issues when applying adaptive Hankel matrices, as explained in subsection 1-4-1, lies in addressing the challenge of PE, which is needed to identify the whole

dynamics of the model [40].

To overcome this challenge, the system can be excited with additional perturbations, which may lead to undesired output responses. These perturbations must be kept to a minimum while still ensuring PE. Van der Veen [50] addresses this by proposing a square-root covariance filter with directional forgetting of past information. This method provides a safer alternative to traditional recursive least-squares algorithms in terms of PE, as it selectively forgets only the information orthogonal to the current excitation, preserving the parallel components. Directional forgetting maintains the boundedness of the covariance matrix while retaining the algorithm's adaptivity. A recursive CL SPC framework with this directional forgetting algorithm was applied to two realistic experimental setups, where one involves a piezo-actuated beam.

1-4-3 Noise mitigation strategies

In the context of DDPC, handling noise in the data used for control design and implementation is another critical challenge. Noise can significantly degrade control performance, making employing effective noise mitigation strategies essential. This subsection discusses some regularization and decomposition techniques based on insights from the literature.

Regularization One strategy to mitigate noise is regularization. Regularization techniques add penalties to the control problem formulation, helping attenuate noise's effects on the control action. The application of weighted Tikhonov regularization [51] and Lasso regularization [52] facilitates the incorporation of prior knowledge or assumptions about the system or control action, thereby enhancing the robustness of the predictive control solution against noise [53]. Regularization can be particularly useful when the data is highly noisy or the system dynamics are poorly understood by imposing some prior knowledge [54].

Decomposition Techniques Another strategy for noise mitigation is the use of decomposition methods, such as Dynamic Mode Decomposition (DMD) and Singular Value Decomposition (SVD). These methods offer a way to handle noise by decomposing the data into modes associated with specific dynamic behaviors [55]. This technique allows for identifying and separating noise-dominated modes from those capturing the essential dynamics of the system [56]. Implementing decomposition within DDPC, with an emphasis on the relevant dynamic modes, can significantly improve control performance, even in the presence of noise [57].

In conclusion, effectively handling noise in DDPC requires a multifaceted approach, leveraging one or a combination of methods such as regularization and decomposition techniques. Sasella's work discusses more strategies [53, 55]. They provide a comprehensive toolkit to enhance the performance of direct DDPC systems in noisy environments.

1-4-4 Real-time feasibility

Another key challenge associated with direct DDPC is ensuring real-time feasibility. While direct DDPC strategies present promising solutions for managing complex systems through the

direct use of I/O data, their practical deployment in real-time scenarios is often constrained by substantial computational demands—particularly when online adaptation of data matrices is required. Both SPC and DeePC can involve large-scale computationally demanding optimizations, whereas DeePC even has more optimization variables because slack variable g is also completely free. Real-time application of these controls requires efficient computational strategies to handle the extensive data. In this subsection, possible solutions are introduced to reduce the computational times.

Matrix Decomposition Techniques One solution to enhance real-time feasibility is the introduction of matrix decomposition techniques. Firstly, DMD and SVD decompositions, as described earlier, are used to mitigate noise. Still, other large datasets decomposition techniques, such as an RQ/LQ decomposition [58], can be applied to break the optimization problems into smaller parts that can be solved with less computational effort [59]. Although the decomposition process is computationally expensive due to the requirement of row-wise processing of the original data matrix [58], efficient updates can be achieved using Givens rotations [60]. This improves efficiency and scalability, enabling the application of direct DDPC methods in more extensive and complex systems.

An example of the application of matrix decomposition is found in γ-DDPC, which utilizes an LQ decomposition of the Hankel data matrices to reduce the size of the optimization variable [61].

Potential MPC-inspired solution Real-time feasibility can also be obtained by using an MPC-inspired solution. As explained in section 1-2-2, MPC also faces computational challenges due to solving an optimization problem at each time step [62]. To address this, various strategies, such as sparse dynamic programming and explicit MPC have been developed. Sparse dynamic programming reduces computational complexity by exploiting the sparsity of the optimization problem, allowing for more efficient solutions by breaking the problem into smaller, computationally manageable subproblems [34]. Explicit MPC, on the other hand, pre-computes control laws offline and stores them as piecewise affine functions, enabling real-time implementation without solving an optimization problem online [35]. These techniques, initially developed for MPC, can also be extended and applied to direct DDPC methods.

1-5 Applications of Direct Data-Driven Predictive Control

Indirect DDPC has been widely applied across various domains, including Power Electronics [63], Heating and Ventilation [64], and Agriculture [65]. In contrast, real-life implementations of direct DDPC remain relatively scarce [50].

This section presents several examples of direct DDPC applications, illustrating both their practical potential and inherent limitations. These cases highlight successful implementations while also revealing the remaining challenges in real-world deployment.

1-5-1 Nuclear Reactor

In Vajpayee et al. [66], a SPC methodology for a nuclear reactor is introduced. This paper considers time variations in the process by recursively updating the control parameters with the arrival of new data (assuming that this new data is persistently exciting). To minimize computational cost, an efficient recursive updating procedure is used from [67]. This approach efficiently leverages an algorithm that avoids computing a full QR decomposition at each time step, a process that is computationally intensive, by instead employing a weighted scheme based on the Givens rotations method [60]. Limitations of this study are its exclusive reliance on simulations, without validation through real-world applications, and the lack of reported computational times.

1-5-2 Piezo-Actuated Beam

In van der Veen's work [50], an experiment was performed on a physical beam structure equipped with piezoelectric transducers. The goal of the control algorithm was to actively dampen the first two vibration modes while remaining responsive to changes in the system dynamics. To evaluate its adaptability, a structural modification was introduced midway through the experiment by clamping the free end of the beam, requiring the algorithm to adjust to the altered dynamic behavior.

A key challenge in this experiment was ensuring that the collected data remained persistently exciting (PE) even after the system dynamics changed. The experiment was implemented on a real-life setup, with the control algorithm running at a rate of 200 Hz, corresponding to a computational time of less than 5 ms per iteration.

1-5-3 Excavator

The results of Wegner [49] confirm the ability to control strongly nonlinear mechanical real-world systems with DeePC using a high-fidelity simulation of a 2 degrees of freedom and 4 degrees of freedom Excavator. They state that data collection is crucial, especially for real-world applications with substantial noise and delay.

The block Hankel matrices ($Y_{i_p,f,\bar{N}}, U_{i_p,f,\bar{N}}$, $U_{i,p,\bar{N}}$ and $Y_{i,p,\bar{N}}$) are computed offline, and control is performed online. An implementation with adaptive Hankel matrices was tried but was unsuccessful due to the inability to achieve real-time capability at 20 Hz.

Because of the significant presence of noise, future work is suggested to look into a method to quantify data fitness to speed up the workflow in collecting data and approaches that can reduce solving time.

1-5-4 Autonomous Vehicle Path Following

In Hromatko et al. [68], a comparison between a DeePC and a classic MPC with the true system model is made. The controllers were tested by performing a maneuver (double lane change) at an 80 km/h speed as quickly as possible. The results indicate that the nominal MPC and the DeePC perform similarly, with a slightly larger solve time for DeePC with a

mean and a maximum solver time of, respectively, 4 ms and 8 ms. This means that real-time control (under 50 ms) is still achieved.

1-5-5 Quadcopter

The quadcopter application in Elokda et al. [12] demonstrates that real-life real-time control can be achieved using DeePC, with performance comparable to MPC when a linearized first-principles model is available. In particular, DeePC shows potential for applications where obtaining an accurate parametric model is either impractical or infeasible. However, when a well-identified model is available, MPC still outperforms DeePC regarding control precision and efficiency.

The study also highlights that the optimization problem in DeePC is solved efficiently, with computation times of 4.14 ms for three outputs and 6.66 ms for five outputs. The real-time implementation was executed at 25 Hz, corresponding to a 40 ms control period.

1-5-6 Overview of applications

Table 1-1 provides an overview of the discussed applications, specifically indicating the type of direct DDPC method employed, whether a real-life application is included, whether an adaptive approach is used, and what the reported computational times are.

Table 1-1: An overview of selected direct DDPC applications, looking at theDDPC algorithm used, whether the experiment was conducted on a real-life setup, if an adaptive method was used, and the computational times.

	DDPC	Real-Life	Adaptive method	Computation Time
Nuclear Reactor	SPC		✓	-
Piezo Actuated Beam	SPC	✓	✓	< 5 ms
Excavator	DeePC			< 50 ms
Vehicle Path Following	DeePC			< 8 ms
Quadcopter	DeePC	✓		4.14 - 6.66 ms

This overview supports existing observations in the literature, which conclude that DDPC is rarely applied to real-life systems [50, 69], particularly in the context of adaptive methods. However, the computation times reported for other applications are sufficiently low to control a real-life system.

1-6 Research Question

This chapter initially presented the foundational concepts and methodologies associated with indirect control approaches. Multiple identification and optimal control methods were given to create an overview of the possibilities of the known methodologies. It then transitioned to a detailed examination of direct DDPC, with particular emphasis on the comparison to

indirect control methods. As part of this discussion, two of the most widely adopted direct DDPC strategies, SPC and DeePC, along with their CL variants, were introduced.

Subsequently, the chapter elaborated on several critical challenges associated with direct DDPC, such as managing strongly nonlinear or time-varying systems, ensuring PE, mitigation noise, and achieving real-time feasibility. Potential solutions to these issues were provided, including the use of adaptive Hankel matrices, various noise mitigation strategies, and computational techniques such as matrix decomposition methods and approaches derived from the MPC framework.

Next, an overview of existing applications, summarized in Table 1-1, revealed that only a limited number of real-world implementations successfully address all the identified challenges. Given the availability of experimental infrastructure at the DCSC laboratory, the piezo-actuated beam setup has been selected as the basis for this research.

This research primarily focuses on SPC-based techniques, as they support handling constraints with the receding horizon optimal control method. In contrast, DeePC often entails a significantly larger number of optimization variables, complicating real-time implementation [44]. Accordingly, SPC is selected in this work for its advantageous balance between computational efficiency and practical feasibility.

This all resulted in the following research question:

> *How can an adaptive direct data-driven predictive controller be designed to enable constrained, real-time vibration control of a real-world piezo-actuated beam setup?*

To address this research question, the following subquestions are formulated:

1. How can the adaptivity of the algorithm be incorporated?

   The theoretical framework for adaptive implementations is considered in this research. However, the piezo-actuated beam setup is not suitable for fully demonstrating the benefits of adaptivity due to hardware limitations. The added value of an adaptive approach is expected to be more evident in systems that are not already approximately Linear Time-Invariant (LTI).

2. How can computational time be reduced to achieve real-time feasibility?

   Achieving real-time feasibility is a key objective, particularly when constraints are included. The impact of decreasing optimization variables by implementing CL SPC and matrix decomposition techniques to be able to update the SysID recursively is investigated while also analyzing which parts of the algorithms are the bottleneck theoretically and practically.

3. How do the proposed controllers perform compared to each other?

   The effectiveness of the controllers is evaluated by comparing their performance against each other. The advantages and disadvantages of the proposed approach are analyzed, considering computational efficiency and control performance.

4. How can vibration control be effectively implemented?

   The research is focused on integrating hardware and software for vibration control. The controller is designed to dampen the first two resonance modes of the piezo-actuated beam while ensuring system stability.

The structure of this thesis is as follows. Chapter 2 introduces and analyzes four adaptive direct DDPC algorithms, which are standard SPC, CL SPC, Recursive Closed Loop Subspace Predictive Control (R-CL SPC), and Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC). Chapter 2 furthermore presents the theoretical formulation of each method and discusses their computational properties, particularly focusing on real-time feasibility. Chapter 3 outlines the simulation methodology used to evaluate the algorithms, while Chapter 4 presents and interprets the simulation results, including performance comparisons and computational times. Following the simulation-based evaluation, Chapter 5 describes the implementation of the control strategies on a real-world piezo-actuated beam setup, covering both hardware and software aspects, and also examines the starting point for the baseline model. The stability analysis of the internal controller and the CL system, the methodology, and the results of the experimental tests are presented in Chapter 6, where the real-life performance is assessed. Finally, Chapter 7 summarizes the findings, answers the research questions, discusses limitations, and provides recommendations for future work.

Chapter 2

Theoretical Background of Subspace Predictive Control Algorithms

This chapter lays the theoretical foundation for all direct Data-Driven Predictive Control (DDPC) algorithms discussed in this thesis, with a particular emphasis on their formulation and computational efficiency expressed in terms of Floating Point Operations (FLOPs). The chapter begins by introducing the necessary preliminaries and mathematical background, providing the foundation for the subsequent methods. This is followed by a detailed exposition of the standard Subspace Predictive Control (SPC) algorithm. The Closed Loop Subspace Predictive Control (CL SPC) variant distinguishes itself by leveraging a parametric model in predictor-form to improve performance. The Recursive Closed Loop Subspace Predictive Control (R-CL SPC) method further reduces the computational burden by employing recursive updates. Finally, the Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC) approach extends this framework by incorporating constraint handling through the use of a Quadratic Programming (QP) solver.

These algorithmic developments directly contribute to answering the research questions by introducing adaptiveness across the SPC variants and evaluating real-time feasibility through a FLOPs-based analysis. A Floating Point Operation is a single arithmetic computation, such as addition, multiplication, or division, performed on floating point numbers [70]. As a machine-independent metric of computational complexity, FLOPs serve as a suitable relative performance indicator for assessing computational time requirements [71].

2-1 Preliminaries

This section begins by elaborating on the necessary preliminaries required to understand the subsequent algorithms. A Single-Input Single-Output (SISO) Linear Time-Invariant (LTI) system is considered in both innovation-form and predictor-form, as defined in Equation 2-1 to 2-2 and Equation 2-3 to 2-7, respectively.

The innovation-form state-space equations are defined as:

$$x_{k+1} = Ax_k + Bu_k + Ke_k \tag{2-1}$$

$$y_k = Cx_k + Du_k + e_k \tag{2-2}$$

where $u_k \in \mathbb{R}$ represents the system input, $y_k \in \mathbb{R}$ the system output, and $x_k \in \mathbb{R}^n$ the state vector. The term $e_k \in \mathbb{R}$ is a zero-mean white noise process, and $K \in \mathbb{R}^{n\times 1}$ is the Kalman gain. The system matrices are defined as follows: $A \in \mathbb{R}^{n\times n}$ is the system state transition matrix, $B \in \mathbb{R}^{n\times 1}$ maps the control input to the states, $C \in \mathbb{R}^{1\times n}$ maps the states to the output, and $D \in \mathbb{R}$ represents the direct feedthrough term from input to output.

The predictor-form state-space equations are given as:

$$x_{k+1} = Ax_k + Bu_k + K(y_k - Cx_k - Du_k) \tag{2-3}$$

$$y_k = Cx_k + Du_k + (y_k - Cx_k - Du_k) \tag{2-4}$$

Rearranging these equations, the system can be rewritten as:

$$x_{k+1} = (A - KC)x_k + (B - KD)u_k + Ky_k \tag{2-5}$$

$$x_{k+1} = \tilde{A}x_k + \tilde{B}u_k + Ky_k \tag{2-6}$$

$$y_k = Cx_k + Du_k \tag{2-7}$$

the system is now expressed in measured outputs rather than explicitly modeling the noise terms.

2-2 Subspace Predictive Control (SPC)

The initial algorithm employed in this study is the standard SPC algorithm with the incorporation of adaptive features. This algorithm serves as the baseline for subsequent extensions. As explained earlier, this approach was chosen over Data-Enabled Predictive Control (DeePC) due to its reduced number of optimization variables and the potential for a complete analytical solution. By eliminating the need for an explicit optimization scheme and a QP solver in the initial stages, SPC provides a foundation, allowing for a gradual increase in complexity if needed.

To enhance its adaptability, a modification is introduced compared to the standard SPC formulation as shown in [10]. This modification includes dynamically updating the Hankel matrices at each time step and thereby avoiding the offline approximation of the system. As a result, the algorithm now consists of four main steps that are executed at every control step:

1. Constructing the block Hankel matrices.
2. Solving a least squares problem.
3. Constructing the controller vector.
4. Analytically calculating the first input.

This section explores the modified standard SPC algorithm and identifies its computationally demanding components. The computational complexity is estimated using FLOPs to provide a theoretical quantitative assessment.

2-2-1 Mathematical Foundation

The observability matrix Γ and the reversed extended controllability matrix $\mathcal{K}$ are given in Equation 2-8, and the block Toeplitz matrices $H_{(B,D)}$ and $H_{(K,I)}$ are defined as shown in Equation 2-9.

$$\Gamma = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{f-1} \end{pmatrix}, \quad \mathcal{K} = \begin{bmatrix} A^{p-1}\bar{B}, & A^{p-2}\bar{B}, & \dots, & \bar{B} \end{bmatrix} \tag{2-8}$$

With $\bar{B} = \begin{bmatrix} B & K \end{bmatrix}$

$$H_{(B,D)} = \begin{pmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{f-2}B & CA^{f-3}B & \cdots & D \end{pmatrix} \tag{2-9}$$

Where $p \in \mathbb{N}_0$ and $f \in \mathbb{N}_0$. The observability matrix $\Gamma \in \mathbb{R}^{f \times n}$, where n is the system order. The reversed extended controllability matrix $\mathcal{K} \in \mathbb{R}^{n \times 2p}$ represents the system's response to past inputs and disturbances. The matrices $H_{(B,D)} \in \mathbb{R}^{f \times f}$ and $H_{(K,I)} \in \mathbb{R}^{f \times f}$ are block Toeplitz matrices, representing the effect of future inputs and disturbances on the system.

The block Hankel matrices are defined as presented in Equation 2-10, while the past data matrix W_i is specified in Equation 2-11.

$$U_{i,p,\bar{N}} = \begin{bmatrix} u_i & u_{i+1} & \cdots & u_{i+\bar{N}-1} \\ u_{i+1} & u_{i+2} & \cdots & u_{i+\bar{N}} \\ \vdots & \vdots & \ddots & \vdots \\ u_{i+p-1} & u_{i+p} & \cdots & u_{i+\bar{N}+p-2} \end{bmatrix}, \tag{2-10}$$

$$W_i = \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \tag{2-11}$$

Using these matrices, the following expression is obtained:

$$Y_{i_p,f,\bar{N}} = \Gamma \mathcal{K} W_i + H_{(B,D)} U_{i_p,f,\bar{N}} + H_{(K,I)} E_{i_p,f,\bar{N}} \tag{2-12}$$

Where $Y_{i_p,f,\bar{N}} \in \mathbb{R}^{f \times \bar{N}}$ is the output matrix over f time steps, $W_i \in \mathbb{R}^{2p \times \bar{N}}$ is the past input-output data matrix, $U_{i_p,f,\bar{N}} \in \mathbb{R}^{f \times \bar{N}}$ is the input matrix over f time steps, and $E_{i_p,f,\bar{N}} \in \mathbb{R}^{f \times \bar{N}}$ represents the effect of process noise over f steps.

Since the noise is assumed to be zero-mean white noise, $E_{i_p,f,\bar{N}}$ is approximated as a zero matrix. This approximation allows the term $H_{(K,I)} E_{i_p,f,\bar{N}}$ to be omitted from the equation, resulting in Equation 2-13.

$$Y_{i_p,f,\bar{N}} = \Gamma\mathcal{K}W_i + H_{(B,D)}U_{i_p,f,\bar{N}} \tag{2-13}$$

This simplification reduces the system equation to an expression where the future output data ($Y_{i_p,f,\bar{N}}$) are modeled as a function of past Input-Output (I/O) data ($U_{i,p,\bar{N}}$ and $Y_{i,p,\bar{N}}$) and future input data ($U_{i_p,f,\bar{N}}$) which are all known.

2-2-2 Block Hankel Matrices

The first step involves constructing the block Hankel matrices from the data sequence, as described in [13], and considering the variables shown in Figure 1-3.

Ensure that $U_{i,p,\bar{N}}$ and $Y_{i,p,\bar{N}}$ have u_i and y_i as their first elements, respectively, as shown in Equation 2-10. Similarly, $U_{i_p,f,\bar{N}}$ and $Y_{i_p,f,\bar{N}}$ should have u_{i_p} and y_{i_p} as their first elements, where $i_p = i + p$, as shown in Figure 1-3. The block Hankel matrices consist of either p or f block rows and $\bar{N}$ columns.

Since there are no matrix multiplications in creating the Hankel matrices, it is difficult to express the amount of FLOPs. However, since every multiplication or addition is a FLOP, it is assumed that the time required to create the matrix is correlated with the size of the created Hankel Matrices. This means two times a $p \times \bar{N}$ matrix and two times a $f \times \bar{N}$ matrix, which adds up to approximately $(2(p+f)\bar{N})$ FLOPs.

2-2-3 Least Squares

The second step involves solving a least-squares problem. As shown in Equation 2-13, this formulation enables the least-squares problem defined in Equation 2-14 to be solved for the estimation of the variables $\Gamma\mathcal{K}$ and $H_{(B,D)}$. This problem can be solved analytically using a pseudo-inverse, as presented in Equation 2-15, and yields a unique solution if the Persistance of Excitation (PE) conditions are satisfied, such that $\begin{pmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \\ U_{i_p,f,\bar{N}} \end{pmatrix}$ is full row rank.

$$\min_{\Gamma\mathcal{K},H_{(B,D)}} \left\| Y_{i_p,f,\bar{N}} - \begin{pmatrix} \Gamma\mathcal{K} & H_{(B,D)} \end{pmatrix} \begin{pmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \\ U_{i_p,f,\bar{N}} \end{pmatrix} \right\|_F^2. \tag{2-14}$$

$$\begin{pmatrix} \Gamma\mathcal{K} & H_{(B,D)} \end{pmatrix} = Y_{i_p,f,\bar{N}} \begin{pmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \\ U_{i_p,f,\bar{N}} \end{pmatrix}^{\dagger} \tag{2-15}$$

The computation of this Pseudo-inverse takes a certain amount of FLOPs, which are explained below:

- **Step 1: Matrix Pseudo-inverse using Singular Value Decomposition (SVD)** $\left(\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,f,\bar{N}} \end{bmatrix} \right)^{\dagger}$: The matrix is decomposed as $A = U\Sigma V^T$ where U and V are orthogonal matrices with size $\bar{N} \times \bar{N}$ and $(2p+f) \times (2p+f)$, and Σ is a diagonal rectangular matrix of $\bar{N} \times (2p+f)$ containing $2p+f$ singular values. The pseudo-inverse is obtained as $A^{\dagger} = V\Sigma^{\dagger}U^T$ where $\Sigma^{\dagger}$ is computed by inverting its nonzero singular values. The computational cost of SVD is approximately $(\bar{N}^2(2p+f) + (2p+f)^3)$ FLOPs, while inverting the nonzero singular values also adds an additional approximately $(2p+f)$ FLOPs. Furthermore, a sparse matrix product of $V\Sigma^{\dagger}$ of approximately $(2(2p+f))$ FLOPs and a matrix matrix product of this result times U^T of approximately $(\bar{N}(2p+f)^2)$ FLOPs. This totals the whole pseudo-inverse to approximately be $((2p+f)(\bar{N}^2 + \bar{N}(2p+f) + (2p+f)^2 + 3))$ FLOPs.

- **Step 2: Matrix Multiplication** $Y_{i_p,f,\bar{N}} \left(\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,f,\bar{N}} \end{bmatrix} \right)^{\dagger}$: The matrix multiplication of an $f \times \bar{N}$ matrix with a $\bar{N} \times (2p+f)$ matrix costs approximately $(2f\bar{N}(2p+f) - f(2p+f))$ FLOPs

In total this equation takes approximately $((2p+f)(\bar{N}^2+\bar{N}(2p+f)+(2p+f)^2+2f\bar{N}-f+3))$ FLOPs.

2-2-4 Constructing Controller Vector

The third step involves constructing the controller vector w_p. This requires first forming the two vectors of size p, $U_{\hat{i},p,1}$ and $Y_{\hat{i},p,1}$, as defined in Equation 2-16 and illustrated in Figure 1-3. These vectors are structured as such due to the SISO assumption made earlier. The first elements of these vectors are $u_{\hat{i}}$ and $y_{\hat{i}}$, respectively. With $\hat{i} = i + N - p$. The final controller vector w_p is obtained by vertically concatenating $U_{\hat{i},p,1}$ and $Y_{\hat{i},p,1}$, as shown in Equation 2-17.

$$U_{\hat{i},p,1} = \begin{pmatrix} u_{\hat{i}} & u_{\hat{i}+1} & \cdots & u_{\hat{i}+p-1} \end{pmatrix}^T, \tag{2-16}$$

$$w_p = \begin{pmatrix} U_{\hat{i},p,1}^T & Y_{\hat{i},p,1}^T \end{pmatrix}^T \tag{2-17}$$

2-2-5 Calculate the First Input

Finally, the future input sequence $U_{\hat{i}_p,f,1}$ can be computed with all the constructed variables. The calculation of this input depends on the choice of the cost function. Two different cost functions are discussed in the following subsections.

Simple Cost Function

The first cost function considered is the Simple cost function, as defined in Equation 2-18. In this formulation, Q typically represents the weight matrix applied to the tracking error, penalizing deviations of the state x from the reference of the states r_x, with $Q \geq 0$. However, since only the output y of a SISO system is considered in this context, Q reduces to a scalar weight. Similarly, R is the weight matrix applied to the control input u, penalizing the magnitude of control actions, with $R > 0$ typically assumed. Given the SISO setup, R also becomes a scalar in this case.

$$J(y,u) = (y-r)^T Q(y-r) + u^T R u \tag{2-18}$$

Minimizing this cost function is the same as minimizing Equation 2-20 at every timestep. This ensures that the system output follows the reference signal while keeping the control inputs from going to extreme values. This cost function ultimately leads to the expression in Equation 2-21, which provides an analytical solution for calculating the future input sequence. The analytical solution introduces diagonal matrices $\tilde{R}$ and $\tilde{Q}$, with $\tilde{R} \in \mathbb{R}^{f \times f}$ and $\tilde{Q} \in \mathbb{R}^{f \times f}$ with the scalars R and Q on the diagonal, respectively. Additionally, Equation 2-20 and Equation 2-21 involve the reference vector r_f shown in Equation 2-19, with $\hat{i}_p = i + N$ as shown in Figure 1-3.

$$r_f = \begin{pmatrix} r_{\hat{i}_p} & r_{\hat{i}_p+1} & \cdots & r_{\hat{i}_p+f-1} \end{pmatrix}^T \tag{2-19}$$

$$J(Y_{\hat{i}_p,f,1}, U_{\hat{i}_p,f,1}) = (Y_{\hat{i}_p,f,1} - r_f)^T \tilde{Q}(Y_{\hat{i}_p,f,1} - r_f) + U_{\hat{i}_p,f,1}^T \tilde{R} U_{\hat{i}_p,f,1} \tag{2-20}$$

$$U_{\hat{i}_p,f,1} = \left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q} \left(r_f - \Gamma \mathcal{K} w_p\right) \tag{2-21}$$

The computation of $U_{\hat{i}_p,f,1}$ involves several steps, each contributing to the total number of FLOPs:

- **Step 1: Matrix Multiplication** $H_{(B,D)}^T \tilde{Q}$: This step is a $f \times f$ times a diagonal $f \times f$ matrix multiplication which requires a approximately $(0.5f^2 + 0.5f)$ FLOPs.
- **Step 2: Matrix Multiplication** $H_{(B,D)}^T \tilde{Q} H_{(B,D)}$: Here the created $f \times f$ matrix is multiplied with a $f \times f$ matrix and requires a approximately $(2f^3 + f^2)$ FLOPs.
- **Step 3: Matrix Inversion** $\left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1}$: The inversion of a $f \times f$ matrix typically costs approximately $(f^3 + 2f)$ FLOPs.
- **Step 4: Matrix-Vector Multiplication** $\Gamma \mathcal{K} w_p$: The multiplication of a $f \times 2p$ matrix is multiplied with a $2p$ vector and requires a approximately $((8fp - f)$ FLOPs.
- **Step 5: Matrix-Vector Multiplication** $H_{(B,D)}^T \tilde{Q} \left(r_f - \Gamma \mathcal{K} w_p\right)$: The created $f \times f$ matrix times a f vector requires approximately $(2f^2 - f)$ FLOPs.

- **Step 6: Final Matrix-Vector Multiplication**: The final $f \times f$ on the left side multiplicated with f vector costs approximately $(2f^2 - f)$ FLOPs.

Summing these contributions, the total computational cost is dominated by approximately $(3f^3 + 5.5f^2 + (8p - 0.5)f)$ FLOPs.

Adapted Cost Function

The second cost function, as shown in Equation 2-22, is an adapted formulation that modifies the input penalty term to penalize changes in the control input, Δu, rather than the input itself. The matrix (or scalar) Q continues to weight the tracking error, penalizing the output y deviations from the reference r. In contrast to the previous formulation, a new weighting term R_Δ is introduced to penalize the rate of change in the control input, Δu.

$$J(y, u) = (y - r)^T Q(y - r) + \Delta u^T R_\Delta \Delta u \tag{2-22}$$

Minimizing this adapted cost function ensures that the system output closely follows the reference signal while keeping the control input change small by adding integral action [37]. Furthermore, it discourages large fluctuations in the control inputs over time.

To find the analytical solution to this problem, two matrices, S_Δ and S_0, are introduced, as shown in Equation 2-23. These matrices define the change in control inputs, $\Delta U_{\hat{i}_p,f,1}$. The specific method for obtaining $\Delta U_{\hat{i}_p,f,1}$ is given in Equation 2-24 [72].

$$S_\Delta = \begin{bmatrix} I & & & \\ -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{bmatrix}, \quad S_0 = \begin{bmatrix} 0 & \cdots & 0 & I & 0 & \cdots & 0 \\ \vdots & \ddots & \cdots & 0 & \cdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \tag{2-23}$$

With $I \in \mathbb{R}$, $S_\Delta \in \mathbb{R}^{f \times f}$, and $S_0 \in \mathbb{R}^{f \times 2p}$, where S_0 contains a single I at the p-th position in the first row.

$$\Delta U_{\hat{i}_p,f,1} = S_0 w_p + S_\Delta U_{\hat{i}_p,f,1} \tag{2-24}$$

Integrating this equation eventually leads to a more rigorous cost function Equation 2-25 and to the analytical solution in Equation 2-26 [72].

$$J(Y_{\hat{i}_p,f,1}, U_{\hat{i}_p,f,1}) = (Y_{\hat{i}_p,f,1} - r_f)^T \tilde{Q}(Y_{\hat{i}_p,f,1} - r_f) + (S_0 w_p + S_\Delta U_{\hat{i}_p,f,1})^T \tilde{R}_\Delta (S_0 w_p + S_\Delta U_{\hat{i}_p,f,1}) \tag{2-25}$$

$$U_{\hat{i}_p,f,1} = \left(S_\Delta^T \tilde{R}_\Delta S_\Delta + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} \left(H_{(B,D)}^T \tilde{Q} \left(r_f - \Gamma \mathcal{K} w_p\right) - S_\Delta^T \tilde{R}_\Delta S_0 w_p\right) \tag{2-26}$$

Since the added matrices are sparse, quantifying the impact of these matrix multiplications in terms of FLOPs is challenging, but their computational cost remains minimal, due to this sparsity. Similarly, $S_\Delta^T \tilde{R}_\Delta S_\Delta$ contain the two sparse matrices (S_Δ^T and S_Δ) and a diagonal matrix ($\tilde{R}_\Delta$) and the same holds for $S_\Delta^T \tilde{R}_\Delta S_0 w_p$ with two sparse matrices (S_Δ^T and S_0) and a diagonal matrix $\tilde{R}_\Delta$.

Similar to the simple cost function, the dominant cost remains in matrix inversion and multiplications. Therefore, the adapted cost function is slightly more computationally heavy, but the difference may be negligible due to the sparse matrices, depending on the specific problem size and context.

2-2-6 Improvements

As shown in the amount of FLOPs, the (pseudo) inverse computation is identified as a significant bottleneck in the process, as illustrated in Equation 2-15. Reducing reliance on this operation could lead to notable efficiency improvements.

One possible approach to reducing computational complexity is leveraging the known structure of the system, which allows for estimating fewer parameters and decreasing the overall computational burden. CL SPC exploits this structural insight by utilizing the predictor-form Markov parameter matrix, under the assumption that $\tilde{A} = A - KC$ is stable, leading to $C\tilde{A}^p = 0$, as shown in Equation 2-27 [73].

This assumption reduces the size of the (pseudo)inverse calculation, as only the first row of the matrix needs to be estimated. The remaining rows can be constructed using the values of the first row, significantly reducing the computational effort required to solve the system [73].

Furthermore, the CL SPC has an improvement to the standard SPC algorithm because it prevents the Closed Loop (CL) identification bias by only estimating only a single-step-ahead predictor [47]. The next section continues on this CL SPC.

$$\begin{bmatrix} \hat{\Xi}_0 \\ \hat{\Xi}_1 \\ \vdots \\ \hat{\Xi}_{f-1} \end{bmatrix} = \begin{bmatrix} C\tilde{A}^{p-1}\bar{\tilde{B}} & C\tilde{A}^{p-2}\bar{\tilde{B}} & \dots & C\tilde{A}^{p-f}\bar{\tilde{B}} & \dots & C\bar{\tilde{B}} \\ 0 & C\tilde{A}^{p-1}\bar{\tilde{B}} & \dots & C\tilde{A}^{p-f+1}\bar{\tilde{B}} & \dots & C\tilde{A}\bar{\tilde{B}} \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & C\tilde{A}^{p-1}\bar{\tilde{B}} & \dots & C\tilde{A}^{f-1}\bar{\tilde{B}} \end{bmatrix} \tag{2-27}$$

With $\bar{\tilde{B}} = \begin{bmatrix} \tilde{B} & K \end{bmatrix}$.

2-3 Closed-Loop Subspace Predictive Control (CL SPC)

While the previous section described the standard SPC algorithm, this section presents an enhanced version: the CL SPC algorithm. This approach improves upon the standard SPC by exploiting a parametric model of the system. The CL approach enhances performance by leveraging the predictor-form Markov parameters to approximate the system dynamics more effectively. This reduces the number of optimization variables that must be estimated in the Least Squares step, improving computational efficiency [73].

To implement the CL approach, several modifications to the SPC algorithm, as described in section 2-2, are required. The first two steps, which involve constructing the Hankel matrices and solving a reduced Least Squares problem, remain largely unchanged. However, the key distinction of the CL SPC method lies in transforming the system's predictor-form Markov parameters back into the needed innovation-form Markov parameters. This transformation is based on the methodology outlined in [74, 75]. Once the system's Markov parameters are updated, the final step of computing the first input remains identical to that in the standard SPC approach. By integrating these enhancements, CL SPC improves computational efficiency while also being able to work with the CL identification bias. The remainder of this section outlines the modifications and similarities across the four main steps, followed by a discussion of potential improvements.

2-3-1 Block Hankel Matrices

In the first step for the CL variant of the SPC algorithm, the matrices $U_{i,p,\bar{N}}$ and $Y_{i,p,\bar{N}}$ as defined in Equation 2-10 are still required. However, different from the standard SPC, two vectors are needed instead of matrices: $U_{i_p,1,\bar{N}}$ and $Y_{i_p,1,\bar{N}}$, as shown in Equation 2-28. Since the same two $p \times \bar{N}$ matrices are used, but the two $f \times \bar{N}$ matrices are replaced with two $\bar{N}$-dimensional vectors, this step results in a direct reducting in computational complexity compared to the standard SPC. Specifically, it requires $(2(p+2)\bar{N})$ FLOPs instead of $(2(p+f)\bar{N})$ FLOPs.

$$U_{i_p,1,\bar{N}} = \begin{bmatrix} u_{i_p} & u_{i_p+1} & \cdots & u_{i_p+\bar{N}-1} \end{bmatrix}, \tag{2-28}$$

2-3-2 Least Squares

Following Dong et al. [73], the relation between $Y_{i_p,1,\bar{N}}$ and $\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}$ in predictor-form is given by Equation 2-29. When $\tilde{A} = A - KC$ is stable and p is sufficiently large, it can be assumed that $C\tilde{A}^p = 0$. Based on this assumption, Equation 2-31 (and Equation 2-33 for $D \neq 0$) are derived, which can be solved using the Least Squares approach as shown in Equation 2-30 and 2-32.

$$Y_{i_p,1,\bar{N}} = C\tilde{A}^p X_i + \Xi_0 \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \tag{2-29}$$

Assuming $D = 0$

When the assumption is made that $D = 0$, it holds that $\tilde{B} = B - KD = B$. Therefore, Equation 2-30 can be used to estimate the Predictor-form Markov Parameters:
$\Xi_0 \triangleq \begin{bmatrix} C\tilde{A}^{p-1}B & \cdots & CB & C\tilde{A}^{p-1}K & \cdots & CK \end{bmatrix}$.

$$\min_{\Xi_0} \left\| Y_{i_p,1,\bar{N}} - \Xi_0 \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \right\|_F^2. \tag{2-30}$$

This least squares problem can be solved using a pseudo-inverse, as shown in Equation 2-31, and has a unique solution if the PE conditions hold, such that $\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}$ is full row rank.

$$\hat{\Xi}_0 = Y_{i_p,1,\bar{N}} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}^{\dagger} \tag{2-31}$$

This least squares algorithm is calculated using the same previous steps (subsection 2-2-3) and is approximately $((2p)(\bar{N}^2+\bar{N}(2p)+(2p)^2+2\bar{N}p+3))$ FLOPs. The complete calculation of FLOPs is shown in Appendix A-2-1.

When $D \neq 0$

In addition to $D \neq 0$, an estimation for D needs to be found. This is accomplished by appending $U_{i_p,1,\bar{N}}$ to the existing matrix $\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}$, as formulated in Equation 2-32. Consequently, the Predictor-form Markov Parameters are defined:
$\overline{\Xi}_0 \triangleq \begin{bmatrix} C\tilde{A}^{p-1}\tilde{B} & \cdots & C\tilde{B} & C\tilde{A}^{p-1}K & \cdots & CK & D \end{bmatrix}$
This also incorporates the estimation of D.

$$\min_{\overline{\Xi}_0} \left\| Y_{i_p,1,\bar{N}} - \overline{\Xi}_0 \begin{pmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \\ U_{i_p,1,\bar{N}} \end{pmatrix} \right\|_F^2 . \tag{2-32}$$

This least squares problem can be solved with Equation 2-33, and has a unique solution if the PE conditions hold, such that $\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,1,\bar{N}} \end{bmatrix}$ is full row rank.

$$\hat{\overline{\Xi}}_0 = Y_{i_p,1,\bar{N}} \begin{pmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \\ U_{i_p,1,\bar{N}} \end{pmatrix}^{\dagger} \tag{2-33}$$

This formulation introduces additional complexity due to the augmented row in the pseudo inverse calculation and totals up to approximately $((2p+1)(\bar{N}^2+\bar{N}(2p+1)+(2p+1)^2+2\bar{N}(p+1)+3))$ FLOPs. The complete FLOPs calculation is provided in Appendix A-2-2.

2-3-3 Building the Innovation Markov parameters

The next step is building the innovation-form Markov parameters. The predictor-form Markov parameters obtained in the previous step must be converted to their innovation-form counterparts. This transformation can be achieved either through a matrix inversion, as described in Houtzager et al. [76], or via a recursive formulation found in van der Veen [74] and Dong [75]. Given the high computational burden associated with matrix inversion, the latter approach was adopted for implementation.

Assuming $D = 0$

Assuming $D = 0$, the matrix in Equation 2-27 can be constructed, which enables the computation of Ψ_i and Λ_i, as defined in Equation 2-34 and Equation 2-35, respectively [74, 75]. Given that the system is SISO, the overlined components reduce to scalar values that are already known, as they correspond to entries of Ξ_0 or $\overline{\Xi}_0$ obtained in the preceding step.

$$\Psi_i = \hat{\Xi}_i + \sum_{\tau=0}^{i-1} \overline{C\tilde{A}^{i-\tau-1}K} \cdot \Psi_\tau, \quad \Psi_0 = \hat{\Xi}_0, \tag{2-34}$$

$$\Lambda_j = \overline{C\tilde{A}^{j-1}B} + \sum_{\tau=1}^{j-1} \overline{C\tilde{A}^{j-\tau-1}K} \cdot \Lambda_\tau, \quad \Lambda_1 = \overline{CB}, \quad \Lambda_0 = 0 \tag{2-35}$$

The number of FLOPs needed to calculate the full Ψ and Λ is approximately $(2pf^2 + 2pf)$ and $\left(\frac{f(f+1)}{2}\right)$, respectively. The calculations of these numbers are shown in Appendix A-2-3 and A-2-4, respectively. These two combined lead to a total of approximately $\left(\frac{(4p+1)f(f+1)}{2}\right)$.

When $D \neq 0$

When the assumption that $D \neq 0$ holds, $\hat{\overline{\Xi}}_0$ needs to be split into two parts, as shown in Equation 2-36. After $\hat{\Xi}_0$ and D are obtained, a matrix similar to Equation 2-27 (Now $B \neq \tilde{B}$) can be constructed, allowing for the computation of Ψ_i and Λ_j using the slightly adjusted formulas Equation 2-37 and 2-38.

$$\hat{\overline{\Xi}}_0 = [\hat{\Xi}_0 \quad \hat{D}] = \left[C\tilde{A}^{s-1}\tilde{B} \ \cdots \ C\tilde{B} \quad C\tilde{A}^{s-1}K \ \cdots \ CK \quad D\right], \tag{2-36}$$

$$\Psi_i = \hat{\Xi}_i + \sum_{\tau=0}^{i-1} \overline{C\tilde{A}^{i-\tau-1}K} \cdot \Psi_\tau, \quad \Psi_0 = \hat{\Xi}_0, \tag{2-37}$$

$$\Lambda_j = \overline{C\tilde{A}^{j-1}\tilde{B}} + \overline{C\tilde{A}^{j-1}K}D + \sum_{\tau=1}^{j-1} \overline{C\tilde{A}^{j-\tau-1}K} \cdot \Lambda_\tau, \quad \Lambda_1 = \overline{C\tilde{B}} + \overline{CK}D, \quad \Lambda_0 = D \tag{2-38}$$

In comparison with Equation 2-34 and 2-35, the only addition is $\overline{C\tilde{A}^{j-1}K}D$ and the initial condition $\overline{C\tilde{B}} + \overline{CK}D$ which are needed to go from predictor-form $\tilde{B}$ to innovation-form B. This also introduces a slight increase in computational complexity, as demonstrated in Appendix A-2-5. The total cost of constructing Λ amounts to approximately $\left(\frac{f(f+1)}{2} + 2f\right)$. Consequently, the overall amount of FLOPs increases to approximately $\left(\frac{(4p+1)f^2+(2p+5)f}{2}\right)$.

Construction of Ψ and Λ

The complete form of the matrix Ψ is provided in Equation 2-39:

$$\Psi = \begin{bmatrix} \Psi_0^T & \Psi_1^T & \Psi_2^T & \cdots & \Psi_{f-1}^T \end{bmatrix}^T \approx \Gamma \mathcal{K} \tag{2-39}$$

Once all components Λ_j have been determined, the matrix Λ can be assembled as shown in Equation 2-40:

$$\Lambda = \begin{bmatrix} \Lambda_0 & & & \\ \Lambda_1 & \Lambda_0 & & \\ \vdots & \vdots & \ddots & \\ \Lambda_{f-1} & \Lambda_{f-2} & \cdots & \Lambda_0 \end{bmatrix} \approx H_{(B,D)} \tag{2-40}$$

2-3-4 Calculating the first input

The final step is calculating the first input. This step is identical to that of the standard SPC algorithm, as detailed in subsection 2-2-5, and is therefore not further elaborated in this subsection.

2-3-5 Improvements

Although the computational burden of the least-squares problem has been reduced in the CL SPC algorithm compared to the standard SPC formulation, it remains the primary bottleneck in terms of FLOPs. Because of this, the next chapter considers a recursive option. This approach utilizes a recursive least squares algorithm, which updates the predictor-form Markov Parameters with the new incoming data instead of estimating these parameters at every control step. Which will decrease computational complexity even more.

2-4 Recursive Closed-Loop Subspace Predictive Control (R-CL SPC)

As discussed in section 2-2 and section 2-3, the computational cost of calculating the large (pseudo)inverse remains a significant bottleneck. To address this, a recursive method is introduced to update Ξ_0, reducing the computational burden.

In contrast to the standard SPC and CL SPC algorithms, the R-CL SPC method does not directly construct Block Hankel matrices, as described in subsection 2-2-2 and subsection 2-3-1. Instead, it recursively updates the least squares estimate using newly collected data vectors, thereby reducing computational complexity. However, although this approach significantly reduces computational overhead, it may introduce performance trade-offs due to its reliance on incremental updates rather than full recomputation.

Despite these modifications, constructing the innovation-form Markov parameters and calculating the first control input remains identical to the process outlined in section 2-3. Meanwhile, the least squares step is completely removed and replaced by the recursive update.

2-4-1 New Data

A new data timeframe is introduced as shown in Figure 2-1. Within this time frame, a new vector γ_k is constructed (see Equation 2-42), comprising the vectors $U_{k_{-p},p,1}$ and $Y_{k_{-p},p,1}$, which represent the previous p data points (with $k_{-p} = k - p$), as illustrated in Figure 2-1, along with scalar u_k, corresponding to input of the current time.

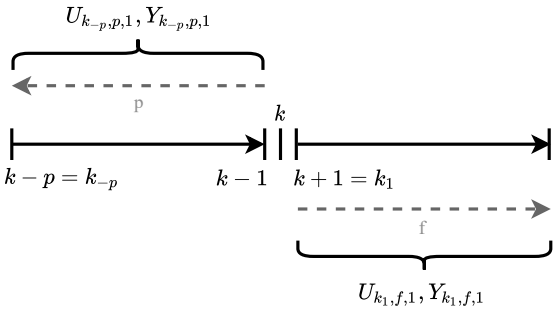


Figure 2-1: Visualization of data for the recursive method, inspired by [?]

$$U_{k_p,p,1} = \begin{pmatrix} u_{k_{-p}} & u_{k_{-p+1}} & \cdots & u_{k_{-p+p-1}} \end{pmatrix}^T \tag{2-41}$$

$$\gamma_k = \begin{pmatrix} U_{k_{-p},p,1}^T & Y_{k_{-p},p,1}^T & u_k \end{pmatrix}^T \tag{2-42}$$

2-4-2 Recursive solution

The most straightforward approach to deriving a Recursive Least Squares (RLS) algorithm with exponential forgetting involves the use of the covariance matrix P_k at k and the exponential forgetting factor λ_{exp}, which discounts past data by a factor of $1 - \lambda_{\text{exp}}$ at every time step, where $\lambda_{\text{exp}} \leq 1$ [77]. Equation 2-43 is used to calculate P_k and afterwards the new $\hat{\Xi}_{0_k}$ is calculated using Equation 2-44.

$$P_k = \frac{1}{\lambda_{\text{exp}}} \left(P_{k-1} - P_{k-1}\gamma_k \left(\lambda_{\text{exp}} + \gamma_k^T P_{k-1} \gamma_k \right)^{-1} \gamma_k^T P_{k-1} \right) \tag{2-43}$$

$$\hat{\Xi}_{0_k} = \hat{\Xi}_{0_{k-1}} - P_k \gamma_k \left(y_k - \gamma_k^T \hat{\Xi}_{0_{k-1}} \right) \tag{2-44}$$

The normal RLS algorithm, as previously described, is seldom used in practice due to numerical instability. Instead, a more robust alternative, commonly referred to as square root RLS, or the inverse QR algorithm, is often employed [50]. This approach utilizes the lower triangular Cholesky factor R_k of the covariance matrix, such that $P_k = R_k R_k^T$. A sequence of Givens rotations is then applied, as illustrated in Figure 2-2 and further detailed in Appendix B-2 [60]. In this context, the Givens rotations effectively perform an inverse QR (or LQ) matrix decomposition.

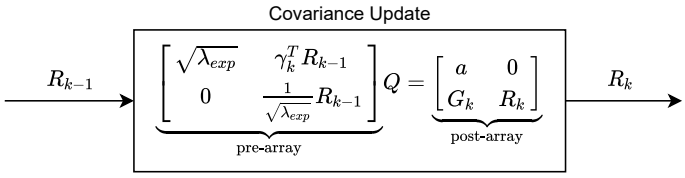


Figure 2-2: Performed Givens rotations for exponential forgetting algorithm, inspired by [50]

From the found values by means of the Givens rotations, Equation 2-45 is used to calculate the new value for $\hat{\Xi}_{0_k}$ [50].

$$\hat{\Xi}_{0_k} = \hat{\Xi}_{0_{k-1}} + \sqrt{\lambda_{\exp}} G_k^T a^{-1} (y_k - \gamma_k^T \hat{\Xi}_{0_{k-1}}) \tag{2-45}$$

The calculations for the amount of FLOPs can be found in Appendix A-2-6 and is approximately $(16p^3 + 48p^2 + 60p + 23)$ FLOPs.

To estimate the amount of historical data used, Equation 2-46 provides an approximation of the effective window length using $\lambda_{\exp}$ [50]. This value is of the same order of magnitude as the Hankel matrix width $\bar{N}$. Therefore, $\hat{\bar{N}}$ is introduced for future comparison purposes.

$$\hat{\bar{N}} = \frac{1}{1 - \lambda_{\exp}} \tag{2-46}$$

2-4-3 Markov Parameters and First Input

The calculation of the Markov Parameters and the first input follows almost the same procedure as the CL SPC, as discussed earlier in section 2-3. However, there is a slight differentiation in the definition of w_p due to the introduction of the new time frame as shown in Equation 2-47 and 2-48.

$$U_{k_{-p+1},p,1} = \begin{pmatrix} u_{k_{-p+1}} & u_{k_{-p+2}} & \cdots & u_k \end{pmatrix}^T, \tag{2-47}$$

$$w_p = \begin{pmatrix} U^T_{k_{-p+1},p,1} & Y^T_{k_{-p+1},p,1} \end{pmatrix}^T \tag{2-48}$$

2-4-4 Improvements

Although the computational burden is significantly reduced for R-CL SPC, there could still be some improvements. Since the analytical solution is used, system constraints cannot be explicitly accounted for. To account for this, a QP solver must be introduced, which uses an optimization scheme to find the best possible input. This enhancement is further described in the next chapter.

2-5 Constrained Recursive Closed-Loop Subspace Predictive Control (CR-CL SPC)

This chapter extends the R-CL SPC approach from section 2-4 by incorporating a QP solver to handle system constraints. The solver used in this implementation is ForcesPro [78, 79] because it works with the available hardware. While this integration enables constraint enforcement, it also introduces additional computational complexity.

The CR-CL SPC algorithm largely follows the structure of the R-CL SPC approach outlined in section 2-4, with a key modification: the analytical method previously used to compute the first input is replaced by an optimization-based approach. This change allows the system to enforce the suggested setup's possible input and output constraints, ensuring feasibility within operational limits. However, introducing a QP solver also affects real-time feasibility.

By integrating constraint handling into the R-CL SPC algorithm, CR-CL SPC seeks to create a viable approach for real-time applications where constraint satisfaction is critical.

2-5-1 First Input

The only difference from the R-CL SPC algorithm is the calculation of the first input with the help op an optimisation scheme. The optimization scheme uses the same time horizon as shown in Figure 2-1, incorporating the vector w_p, which corresponds to the historical data vector introduced in Equation 2-48. The predicted future inputs and outputs are denoted as $U_{k_1,f,1}$ and $Y_{k_1,f,1}$, respectively, and are defined in Equation 2-49:

$$U_{k_1,f,1} = \begin{pmatrix} u_{k_1} & u_{k_1+1} & \cdots & u_{k_1+f-1} \end{pmatrix}^T, \tag{2-49}$$

The optimization problem is presented in Equation 2-50. The problem minimizes a cost function that combines tracking performance, input magnitude, and input smoothness by integrating both the simple and the adapted cost functions introduced earlier in Equation 2-18 and 2-22. The objective is to follow a reference trajectory while respecting system constraints and minimizing unnecessary or abrupt control actions.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=0}^{f-2} \left(0.5\|y_{k_1+i} - r_{f,i+1}\|_Q^2 + 0.5\|u_{k_1+i}\|_R^2 + 0.5\|u_{k_1+i+1} - u_{k_1+i}\|_{R_\Delta}^2 \right) \\ & \qquad + 0.5\|y_{k_1+f-1} - r_{f,f}\|_Q^2 + 0.5\|u_{k_1+f-1}\|_R^2, \\ \text{subject to} \quad & Y_{k_1,f,1} = \Psi w_p + \Lambda U_{k_1,f,1}, \\ & u_{\min} \leq U_{k_1,f,1} \leq u_{\max}, \\ & y_{\min} \leq Y_{k_1,f,1} \leq y_{\max}. \end{aligned} \tag{2-50}$$

In this optimization scheme, the term Q assigns a penalty to the tracking error, encouraging the output y to follow the reference trajectory r closely. The term R penalizes the magnitude of the control input u, discouraging excessive actuation effort. The term R_Δ penalizes changes in the control input. The matrices Ψ and Λ define the data-driven model, which maps past

information w_p and future control inputs $U_{k_1,f,1}$ to the predicted output sequence $Y_{k_1,f,1}$. The first constraint enforces adherence to this model, while the remaining constraints ensure that both control inputs and predicted outputs stay within specified bounds throughout the prediction horizon.

To solve this optimization problem, a QP solver called ForcesPro [78, 79] is employed. The use of this solver introduces nonlinear dependencies and additional computational complexity. As a result, it is not feasible to explicitly express the number of FLOPs required for this computation.

Chapter 3

Simulation Methodology

Whereas the previous chapter provided a theoretical description of the algorithms, this chapter presents the simulation methodology used to evaluate the performance and computational efficiency of the various Subspace Predictive Control (SPC) algorithms. It begins with a description of the simulated system and its parameters, followed by the implementation of the SPC algorithms across several simulation scenarios. All simulations were performed on a laptop with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80Ghz processor.

The initial set of simulations focused on tuning key algorithmic parameters to enable a performance comparison and assess computational complexity while comparing the computational times with the estimated Floating Point Operations (FLOPs). Additional simulations investigated the effect of different cost functions, analyzed the influence of including the feedthrough matrix D, and introduced a methodology to assess the role of the exponential forgetting factor λ_{exp} under varying noise conditions and nonlinear system behavior.

These simulations aimed to validate the earlier theoretical developments while offering practical insights into how adaptivity and computational constraints impact controller performance. They directly address the research questions concerning real-time feasibility, adaptiveness, and control effectiveness.

3-1 System Description and Parameters

This section describes the system used for the simulations and other possible parameters. All simulations were performed on a simple second-order system, which is given in Equation 3-1 (from [50]). The system was selected for its simplicity, allowing the focus to remain primarily on evaluating the control methodology. The noise added was band-limited white noise with variance ${\sigma_v}^2 = 0.01$. The chosen sampling time was 0.1s.

$$y(z) = \frac{0.5z^2}{z^2 + 0.3z + 0.4}u(z) + v(z) \tag{3-1}$$

All SPC variables, such as the future and past window (p and f) and the width of the Hankel matrices ($\bar{N}$) or the exponential forgetting factor (λ_{exp}), which implicitly defines the effective window length ($\hat{\bar{N}}$), are considered tunable variables. The (near) optimal values for these parameters were determined in the initial set of simulations.

3-2 Implementation of SPC Algorithms in Simulation

Now that the simulated system is known, this section aims to describe all the simulations performed. First, a structured tuning procedure was applied to determine the optimal past and future window sizes (i.e., p and f), Hankel matrix width ($\bar{N}$), and exponential forgetting factor (λ_{exp}). The computational complexity of each SPC variant was analyzed by comparing simulation results to estimated FLOPs. Additional simulations investigated the impact of cost function choices, the role of the feedthrough matrix D, and the influence of the exponential forgetting factor λ_{exp} under varying conditions.

The following subsections detail the methodology for parameter tuning and other performance evaluations.

3-2-1 Parameter tuning

Because of there are a lot of possible combination with three tuning parameters, a variation of a greedy algorithm was applied to determine a near-optimal combination of tunable parameters p, f, and $\bar{N}$ (or λ_{exp}). This significantly reduces the number of simulations that must be performed while still being able to find near-optimal values [80].

The process began with a simulation where $p = f$ was varied, while $\bar{N}$ (or λ_{exp}) were kept constant at a sufficiently high value, in this case: $\bar{N} = 400$ and $\lambda_{\text{exp}} = 0.9975$ ($\hat{\bar{N}} = \frac{1}{1-\lambda_{\text{exp}}} = 400$). Once the optimal value for $p = f$ was identified, another simulation was conducted in which this value was used as a fixed future window f, and p was varied independently.

After determining the optimal value of p, a simulation was performed where p was held constant at the found optimal value, and f was varied, determining the optimal value for f. Finally, after the values of p and f were found, another simulation was performed to optimize $\bar{N}$ (or $\hat{\bar{N}}$) by varying its value. This iterative procedure yields a near-optimal combination of the parameters p, f, and $\bar{N}$ (or $\hat{\bar{N}}$). The mean of 10 simulations with different noise seeds was taken to mitigate the outliers.

The cumulative stage cost is the performance metric to determine the near-optimal variables. It represents the sum of the costs calculated at each time step concerning the current input and output based on the cost function Equation 2-22.

A 300-seconds reference trajectory was followed in all simulations, as illustrated in Figure 3-1. During the initial five seconds, white noise with a variance ($\sigma^2 = 5$) was applied at a sampling interval of 1 second to initialize the SPC controller. Following this initialization phase, the controller became active. To allow the controller sufficient time to stabilize, data collection for performance evaluation began at 50 seconds.

As previously mentioned, the system operated with a sample time of 0.1 seconds. The penalty on output deviation, denoted by Q, was set to 1. The penalty on the absolute control input R was set to 0, and the penalty on the rate of change in the control input R_Δ was set to 1. These parameter values were selected based on their ability to yield robust and reliable simulation performance.

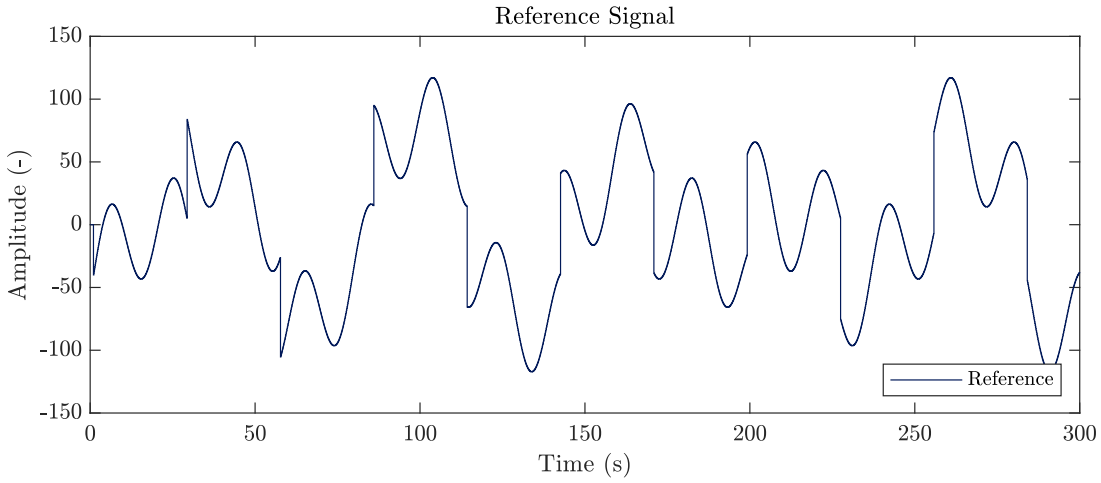


Figure 3-1: The Reference signal used for simulations consists of two sine waves and a square wave. Including the square wave ensures that the signal possesses a broad frequency spectrum, which is needed to ensure Persistance of Excitation (PE).

Furthermore, the results of all simulations were utilized to analyze the computational times associated with the individual components of the different SPC algorithms. These components include the construction of the Hankel Matrices, the solution of the least-squares problem, the update of the covariance matrix, the construction of the innovation-form Markov Parameters, and the calculation of the first input.

These computational times were subsequently compared to the estimated computational complexity, expressed in terms of FLOPs, as previously calculated in chapter 2. For clarity and reference, an overview of these estimates is provided in Table 3-1, in terms of p, f and $\bar{N}$.

Table 3-1: Computational Complexity of different steps of different SPC Variants where the not used steps are indicated with a '-' and the calculation of the first input of the Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC) is indicated with '?' due to the nonlinear way it is calculated.

	Create Hankel Matrices	Solving Least Squares	Update Covariance	Constructing Parameters	Calculate Input
SPC	$2(p+f)N$	$(2p+f)N^2+(4fp+2f^2)N+(8p^3+12p^2f+6pf^2+f^3+2p+f-1)$	-	-	$3f^3+5.5f^2+(8p-0.5)f$
CL SPC	$2(p+2)\bar{N}$	$(2p+1)\bar{N}^2+(4p^2+6p+2)\bar{N}+(8p^3+12p^2+6p+1)$	-	$\frac{(4p+1)f^2+(2p+5)f}{2}$	$3f^3+5.5f^2+(8p-0.5)f$
R-CL SPC	-	-	$16p^3+48p^2+60p+23$	$\frac{(4p+1)f^2+(2p+5)f}{2}$	$3f^3+5.5f^2+(8p-0.5)f$
CR-CL SPC	-	-	$16p^3+48p^2+60p+23$	$\frac{(4p+1)f^2+(2p+5)f}{2}$	?

3-2-2 Cost function comparison

To compare the simple cost function with the adapted cost function, a simulation was conducted on the system in Equation 3-1 with $\sigma_v^2 = 0.01$ to evaluate the performance of the

simple cost function against the 'Adapted' cost function. The standard SPC algorithm was used. The simulation parameters were set as follows: past window size (p) and future window size (f) both equal to 10, sample time at 0.1 seconds, penalty on state deviation (Q) at 10, and penalty on control effort (R or R_Δ) at 1 and the width of the Hankel Matrices are capped at $\bar{N} = 400$. These values were chosen as they provided robust and reliable simulation performance. In contrast, the higher value for Q compared with previous simulations gave a more reasonable steady-state error for the simple cost function.

During the first phase of five seconds, white noise with a variance (σ^2) of 5 was given at a sampling time of 1 second to activate the SPC controller. Following this phase, the controller was engaged, and a step reference 20 was introduced at the 20-second mark.

3-2-3 Use of Feedthrough Matrix D

To find if the feedthrough matrix can be assumed to be zero or not, three simulations were performed on the system in Equation 3-1 using $\sigma_v^2 = 0.01$ and a sampling time of 0.1s to confirm the performance using the feedthrough matrix. The Closed Loop Subspace Predictive Control (CL SPC) algorithm was analyzed both with the feedthrough matrix D and for the case where $D = 0$.

In the simulations, a reference signal of 600 seconds was followed, utilizing the same functions as previously shown in Figure 3-1. During the initial phase of five seconds, white noise with a variance (σ^2) of 5 was applied at a sampling time of 1 second to initialize the SPC controller. After this initialization phase, the controller was engaged. As previously mentioned, the sampling time was set to 0.1 seconds, the penalty on output deviation (Q) was set to 1, the penalty on absolute control input (R) was set to 0, and the penalty on the change in control input (R_Δ) was set to 1. These values were chosen as they provided robust and reliable simulation performance. To reduce ambiguity, the test was performed using three different noise seeds.

3-2-4 Influence of λ_{exp}

To get some initial understanding of how λ_{exp} influences the simulation results, two simulations were performed to investigate further the influence of λ_{exp}.

The first simulation examined the effect of λ_{exp} in the presence of a higher noise level. This simulation was conducted with the same kind of simulation as previously in subsection 3-2-1, but with the system in Equation 3-1 and using $\sigma_v^2 = 5$. In this scenario, the Recursive Closed Loop Subspace Predictive Control (R-CL SPC) algorithm was employed with $p = 2$ and $f = 23$, attempting to follow the reference trajectory shown in Figure 3-1. The sample time was set to 0.1 seconds, the penalty on output deviation (Q) was set to 1, the penalty on absolute control input (R) was set to 0, and the penalty on change in control input (R_Δ) was also set to 1. These values were chosen as they provided robust and reliable simulation performance.

The second simulation investigated the effect of λ_{exp} when the system exhibits nonlinear behavior. In this case, the system switched from the system shown in Equation 3-1 to the system shown in Equation 3-2 at $t = 150s$, with a noise variance of $\sigma_v^2 = 0.01$ and a sampling

time of 0.1s. The system followed a 600s reference trajectory, as shown in Figure 3-1. The penalty parameters were maintained as follows: $Q = 1$, $R = 0$, and $R_\Delta = 1$. These values were chosen as they provided robust and reliable simulation performance.

$$f_2(z) = \frac{-0.5z^2}{z^2 + 0.9z + 0.9}u(z) + v(z). \tag{3-2}$$

This chapter outlines the methodologies used. The results of the simulation methodologies described are presented in the next chapter.

Chapter 4

Simulation Results

This chapter presents the simulation results used to evaluate the performance and computational efficiency of the implemented Subspace Predictive Control (SPC) algorithms. It begins with a parameter tuning procedure, where key variables such as the past and future window sizes (p and f), the Hankel matrix width ($\bar{N}$), and the exponential forgetting factor (λ_{exp}) were optimized for each algorithm. To assess real-time feasibility, the computational complexity of the different SPC variants has been evaluated by comparing measured computational times with the corresponding theoretical Floating Point Operations (FLOPs) estimates.

Subsequent simulations investigated the influence of different cost function formulations, the inclusion of the feedthrough matrix D, and the behavior of the algorithms under varying noise levels and nonlinear dynamics through the role of λ_{exp}.

In the last section of this chapter, some concluding remarks on the simulation results are given, including the found optima, FLOPs accuracy, and a discussion on the choice of cost function, feedthrough matrix, and exponential forgetting factor.

These results provide quantitative insights into the adaptability, computational demands, and control performance of each SPC variant, directly supporting the research questions regarding real-time feasibility, adaptive capability, and comparative controller performance.

4-1 Parameter Tuning

To ensure optimal performance, the key parameters of the SPC algorithms were systematically tuned. These include the past window size p, future window size f, and either the Hankel matrix width $\bar{N}$ or the exponential forgetting factor $\lambda_{\mathbf{exp}}$. The tuning process was performed iteratively, aiming to improve control accuracy based on simulation results.

This approach also enabled the analysis of computational complexity in relation to the theoretical FLOPs estimations shown in Table 3-1. In each subsection, the analysis is simplified to isolate and highlight the effect of the varied parameter.

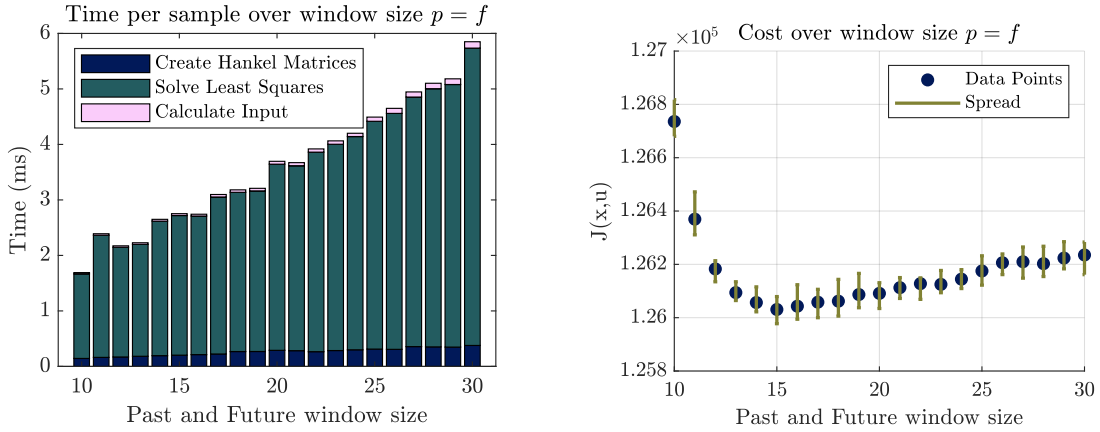
4-1-1 Varying past and future window

The first step in identifying near-optimal parameters involved determining the optimal value when the past and future window lengths were equal, that is, $p = f = \kappa$. A sufficiently large value for $\bar{N}$ (or $\hat{\bar{N}}$) was selected and set to 400 to avoid limitations due to insufficient data. Table 4-1 provides an overview of the estimated number of FLOPs required under the condition $p = f = \kappa$ with $\bar{N} = 400$.

Table 4-1: Computational Complexity of different steps of different SPC Variants for $p = f = \kappa$ and $\bar{N} = 400$. The not used steps are indicated with a '-' and the calculation of the first input of the Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC) is indicated with '?' due to the nonlinear way it is calculated.

	Create Hankel Matrices	Solving Least Squares	Update Covariance	Constructing Parameters	Calculate Input
SPC	1600κ	$8\kappa^3 + 4012\kappa^2 + 800007\kappa - 1$	-	-	$3\kappa^3 + 5.5\kappa^2 + 7.5\kappa$
CL SPC	$800\kappa + 1600$	$8\kappa^3 + 1612\kappa^2 + 322406\kappa + 160801$	-	$2\kappa^3 + 2.5\kappa^2 + 2.5\kappa$	$3\kappa^3 + 5.5\kappa^2 + 7.5\kappa$
R-CL SPC	-	-	$16\kappa^3 + 48\kappa^2 + 60\kappa + 23$	$2\kappa^3 + 2.5\kappa^2 + 2.5\kappa$	$3\kappa^3 + 5.5\kappa^2 + 7.5\kappa$
CR-CL SPC	-	-	$16\kappa^3 + 48\kappa^2 + 60\kappa + 23$	$2\kappa^3 + 2.5\kappa^2 + 2.5\kappa$	?

For the standard SPC algorithm, it was observed that when κ ranged between 10 and 30, the creation of Hankel matrices increased linearly, requiring approximately 10^5 FLOPs, as indicated by the term 1600κ in Table 4-1. In the least-squares step, although the expression contains cubic and quadratic components, the large coefficient of the linear term made it numerically dominant in this range, resulting in an estimated cost around 10^7 FLOPs. The computation of the first input was governed primarily by the cubic term $3\kappa^3$, but remained relatively low in cost, at approximately 10^4 FLOPs.



(a) Mean computational time of different algorithm components per sample as a function of window size.

(b) Cumulative stage costs as a function of window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-1: Impact of varying window size ($p = f$) on simulation time and cumulative stage costs for the Simple SPC.

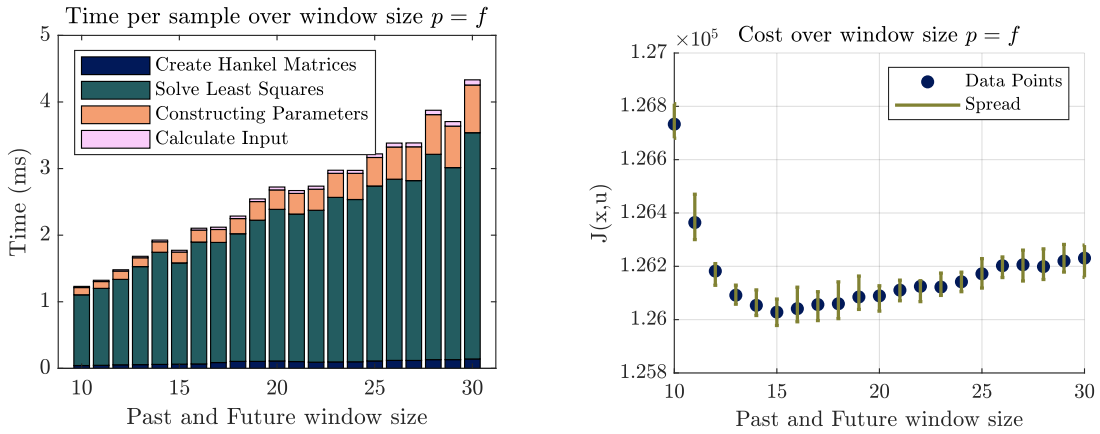
These FLOPs estimates align with the trends shown in Figure 4-1a, where a linear increase

is visible in the computational effort for matrix construction and solving the least-squares problem. A smaller yet steeper increase was observed in the input computation step.

Regarding control performance, as shown in Figure 4-1b, the best results were achieved at $p = f = 15$. Beyond this point, further increases in window size led to a gradual decline in performance.

For the Closed Loop Subspace Predictive Control (CL SPC) algorithm, when κ ranged between 10 and 30, the computational effort of the creation of Hankel matrices increased linearly, requiring approximately 10^5 FLOPs. This was consistent with the expression $800\kappa + 1600$ in Table 4-1. The dominant computational step remained the solution of the least squares problem, which incurred between 10^6 and 10^7 FLOPs. However, this cost was slightly reduced compared to the standard SPC due to a smaller problem formulation and the presence of lower-order terms in the expression.

The computation of the first input was governed by a cubic expression but remained relatively inexpensive, requiring approximately 10^4 FLOPs. An additional step in CL SPC involved the construction of innovation-form Markov parameters, which introduced further computational load. This step scaled with κ^3 in theory and contributed to the order of 10^5 operations within the tested range.



(a) Mean computational time of different algorithm components per sample as a function of window size.

(b) Cumulative stage costs as a function of window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-2: Impact of varying window size ($p = f$) on simulation time and cumulative stage costs for the CL SPC.

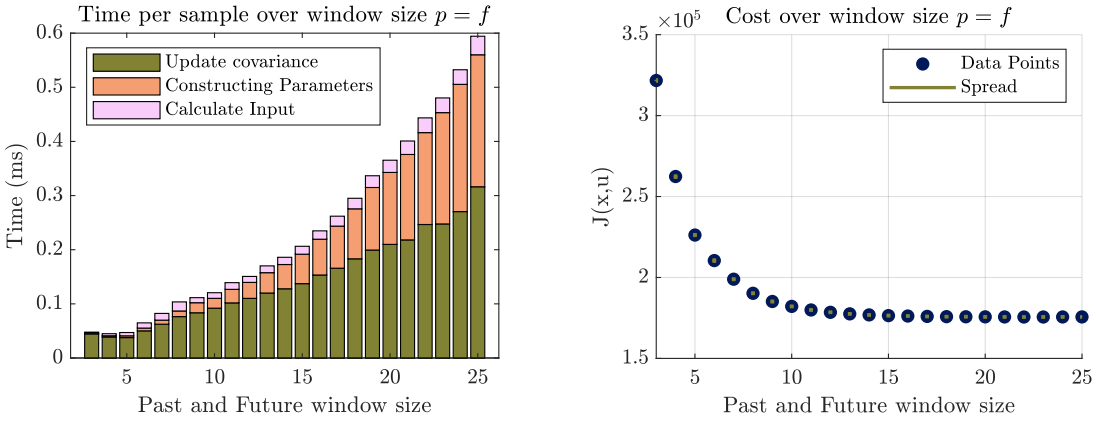
These FLOPs trends are clearly reflected in Figure 4-2a, where both Hankel matrix creation and least squares solving show a linear increase with window size. The cost of computing the first input exhibits a steeper rise, and the added parameter construction step also contributes noticeably to the total computation time, although not in a strictly cubic manner.

As shown in Figure 4-2b, the best control performance was achieved at $p = f = 15$, where the cumulative stage cost was minimized. Beyond this point, increasing the window size led to a

gradual performance decline. Overall, the behavior of CL SPC closely resembled that of the standard SPC, with modest improvements in computational efficiency and a more structured model formulation.

For the Recursive Closed Loop Subspace Predictive Control (R-CL SPC) algorithm, when κ was between 10 and 30, the computational complexity changed significantly compared to standard SPC and CL SPC. The creation of Hankel matrices was no longer required, eliminating an estimated 10^5 FLOPs. Likewise, the least squares problem was not solved at each time step, removing the dominant computational burden of approximately 10^7 FLOPs.

Instead, the algorithm relied on a recursive covariance matrix update, which became one of the most computationally demanding steps, requiring between 10^5 and 10^6 FLOPs, depending on the window size. The construction of innovation-form Markov parameters was also retained and contributed a similar computational load. Meanwhile, the calculation of the first input remained comparable in complexity to previous variants, at approximately 10^4 FLOPs.



(a) Mean computational time of different algorithm components per sample as a function of window size.

(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

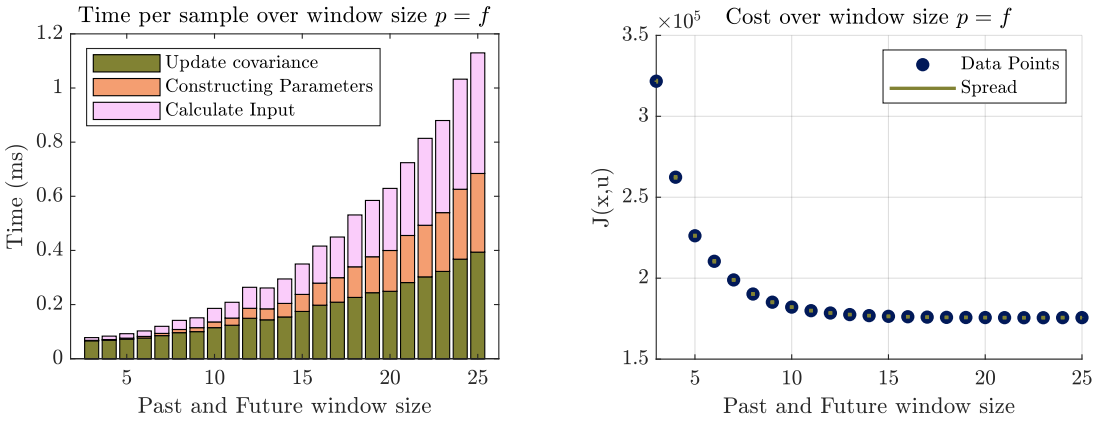
Figure 4-3: Impact of varying window size ($p = f$) on simulation time and cumulative stage costs for the R-CL SPC.

As shown in Figure 4-3a, this restructuring for the R-CL SPC algorithm leads to a clear reduction in computational cost. Removing the explicit least squares step significantly decreased overall execution time, and the recursive update steps dominated the computational profile. Input computation remained the least expensive component, though it increased more sharply with larger window sizes.

Regarding control performance, Figure 4-3b indicates that the optimal performance occurred at $p = f = 23$, which is higher than in the previous algorithms. In contrast to SPC and CL SPC, increasing the window size beyond this point did not lead to a degradation in performance.

The computational complexity of the CR-CL SPC algorithm closely followed that of the R-CL SPC algorithm, with one key distinction: the incorporation of a Quadratic Program-

ming (QP) solver to handle input and output constraints. While the core structure remained unchanged, eliminating the need for explicit Hankel matrix construction and least squares solving, adding the QP optimization step introduced a potential computational burden.



(a) Mean computational time of different algorithm components per sample as a function of window size.

(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-4: Impact of varying window size ($p = f$) on simulation time and cumulative stage costs for the CR-CL SPC.

As shown in Figure 4-4a, the QP solver quickly became one of the dominant contributors to computation time as the window size $p = f$ increased. In contrast to the R-CL SPC algorithm, where the input calculation step was relatively inexpensive, the constraint handling in CR-CL SPC substantially increased the computational time of this component.

Regarding performance, Figure 4-4b shows that the optimal window size remains consistent with that of the R-CL SPC algorithm, with $p = f = 23$ yielding the lowest cumulative stage cost. Notably, no performance degradation was observed for larger window sizes.

Overall, CR-CL SPC retained most of the computational advantages of its recursive counterpart while enabling the handling of constraints, making it well-suited for real-world applications with critical input and output limits.

4-1-2 Varying Past Window Size

The next step in the parameter tuning process involved determining the optimal value for the past window size p, while keeping the future window size fixed. For the SPC and CL SPC algorithms, the future window size was set to $f = 15$, whereas for the R-CL SPC and CR-CL SPC algorithms, it was set to $f = 23$, as established in the previous section. A sufficiently large Hankel matrix width and effective window length were maintained, with $\bar{N} = \hat{\bar{N}} = 400$.

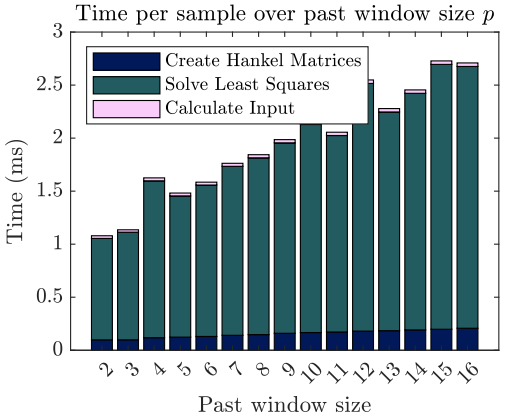
Table 4-2 provides a detailed overview of the estimated computational complexity across different algorithm components for various values of p. This allows for an informed analysis

of how increasing the past window size influences the overall computational cost in each SPC variant.

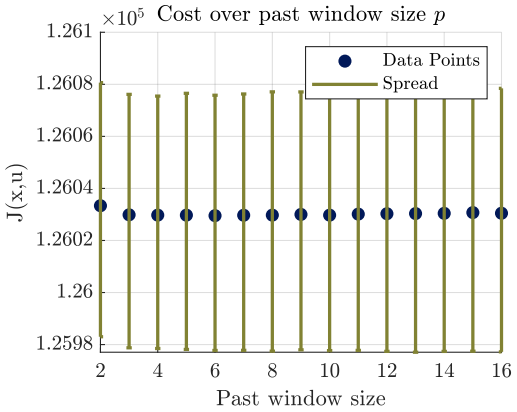
Table 4-2: Computational Complexity of different steps of different SPC Variants for $\bar{N} = 400$, where $f = 15$ for SPC and CL SPC, and $f = 23$ for R-CL SPC and CR-CL SPC. The not used steps are indicated with a '-' and the calculation of the first input of the CR-CL SPC is indicated with '?' due to the nonlinear way it is calculated.

	Create Hankel Matrices	Solving Least Squares	Update Covariance	Constructing Parameters	Calculate Input
SPC	$1600p + 12000$	$8p^3 + 180p^2 + 345352p + 2583389$	-	-	$120p + 10125$
CL SPC	$1600p + 3200$	$8p^3 + 1612p^2 + 320406p + 160801$	-	$450p^2 + 275p + 135$	$120p + 10125$
R-CL SPC	-	-	$16p^3 + 48p^2 + 60p + 23$	$2116p^2 + 1046p + 264.5$	$176p + 38151$
CR-CL SPC	-	-	$16p^3 + 48p^2 + 60p + 23$	$2116p^2 + 1046p + 264.5$	?

For the standard SPC algorithm, Table 4-2 shows that when p ranged from 2 to 16, a linear increase in computational cost was expected for creating the Hankel matrices, described by the expression $1600p + 12000$, with a total of approximately 10^5 FLOPs. The most computationally intensive step remained solving the least squares problem, which involves a cubic expression in p. However, due to the large coefficient of the linear term in the expression $8p^3 + 180p^2 + 345352p + 2583389$, the growth appeared nearly linear in this range, resulting in a total cost between 10^6 and 10^7 FLOPs. The calculation of the first input grew linearly with p and remained relatively inexpensive, at around 10^4 FLOPs.



(a) Mean computational time of different algorithm components per sample as a function of past window size.



(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-5: Impact of past window sizes on simulation time and cumulative stage costs for the Simple SPC.

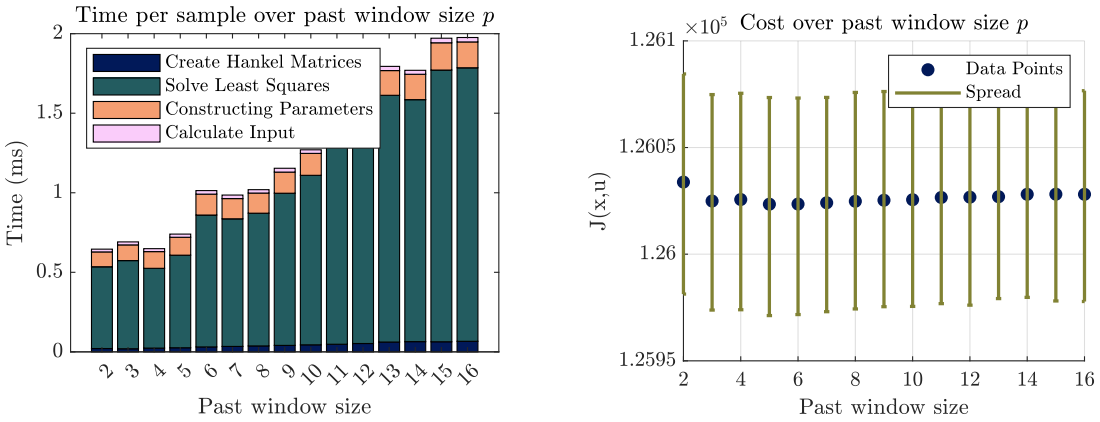
As seen in Figure 4-5a, the measured computational time aligned with these FLOPs expectations. The creation of Hankel matrices exhibited a steady linear increase, and solving the least squares problem dominated the total time, increasing predictably with p. In contrast, the input calculation remained the least demanding among the three components.

Figure 4-5b further illustrates the improvement in control performance as p increases. A sharp decrease in cumulative stage cost is observed up to $p = 3$, after which performance

trend becomes more gradual and steady. Beyond $p = 6$, additional increases in p yield no further performance gains.

For the CL SPC algorithm, Table 4-2 showed computational trends similar to those of standard SPC. The creation of Hankel matrices grew linearly with p, and the calculation of the first input remained relatively small, with a linear cost of approximately 10^4 FLOPs. The least squares step, while structurally similar to SPC, contained a slightly lower constant and linear term, resulting in a marginally lower computational cost in the tested range of p. However, the difference was not substantial, and both algorithms incurred between 10^6 and 10^7 FLOPs for this step.

An additional step in CL SPC was the construction of the innovation-form Markov parameters, which introduced a quadratic cost with respect to p, amounting to approximately 10^5 FLOPs.



(a) Mean computational time of different algorithm components per sample as a function of past window size.

(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-6: Impact of past sizes on simulation time and cumulative stage costs for the CL SPC.

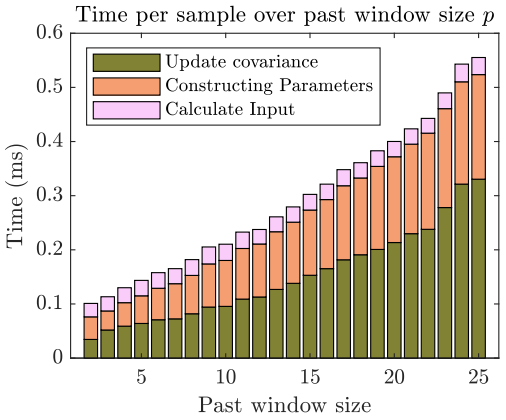
As shown in Figure 4-6a, the observed computational times aligned with the theoretical expectations. The total computational time remained slightly lower than that of standard SPC, largely due to a lower overall constant offset in the least squares computation.

In Figure 4-6b, the cumulative stage cost follows a similar trend to SPC, reaching an optimum at $p = 5$. Beyond this point, increasing p yielded no improvements in performance.

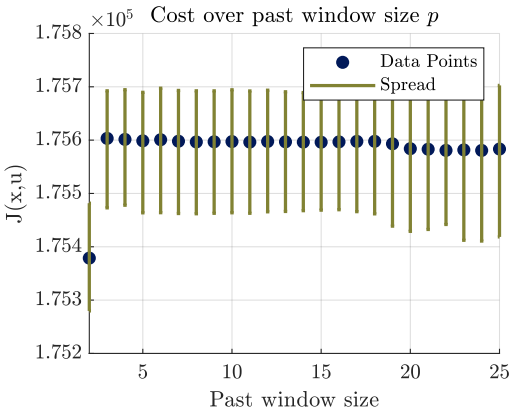
When examining Table 4-2, the R-CL SPC algorithm exhibited a cubic increase in computational complexity for the covariance matrix update, with FLOP counts starting in the 10^4 range and approaching 10^5 as p increased. The construction of the innovation-form Markov parameters contributed an additional quadratic cost, growing into the 10^5 FLOP range for typical values of p. The calculation of the first input remained relatively minor, with a linear dependence on p and a lower overall cost than the other components.

As shown in Figure 4-7a, the covariance update dominated the computational time, increasing

rapidly with p, while the Markov parameter construction also required significant computational effort, though not as steeply. These observations confirmed the trends predicted by the table. The input calculation continued to contribute only a small portion of the total computational load.



(a) Mean computational time of different algorithm components per sample as a function of past window size.



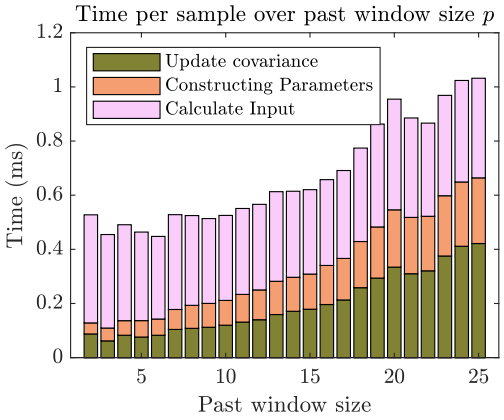
(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-7: Impact of past sizes on simulation time and cumulative stage costs for the R-CL SPC.

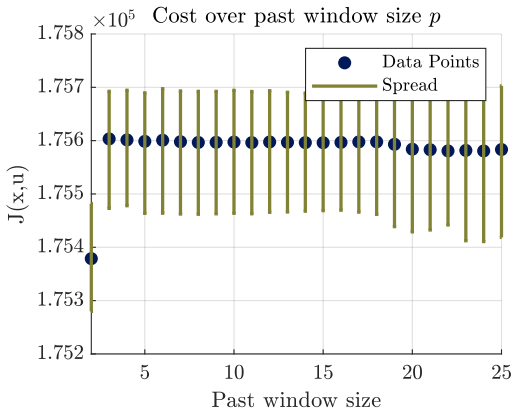
From a performance perspective, Figure 4-7b shows that the optimal past window size was smaller than in previous algorithms, with $p = 2$ yielding the best results. This outcome was consistent with expectations, as the value of p implicitly defines the model order, and the simulated system had an order of two. Increasing p beyond this point led to a noticeable degradation in performance, which then remained relatively stable at a lower level.

The results in Table 4-2 for the CR-CL SPC algorithm were largely comparable to those of the R-CL SPC algorithm, with the notable exception that the number of FLOPs required for the input calculation step could not be specified due to the use of a QP solver.

As shown in Figure 4-8a, the introduction of constraints significantly changes the computational profile. The QP solver becomes the dominant contributor to total runtime. Although its theoretical complexity is not constant, it appeared relatively flat over the tested range of p. The recursive covariance update and the construction of innovation-form Markov parameters contributed similarly to their roles in the R-CL SPC algorithm. However, their relative impact was reduced due to the added overhead introduced by the QP solver.



(a) Mean computational time of different algorithm components per sample as a function of past window size.



(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-8: Impact of past window sizes on simulation time and cumulative stage costs for the CR-CL SPC.

Despite the increased computational cost, Figure 4-8b confirms that the optimal past window size remains at $p = 2$, as in the unconstrained case.

4-1-3 Varying Future Window Size

The next step in the parameter tuning process focused on determining the optimal value for the future window size f, while keeping the past window size p fixed. Based on the previous analysis, the past window was set to $p = 6$ for the SPC algorithm, $p = 5$ for CL SPC, and $p = 2$ for both the R-CL SPC and CR-CL SPC algorithms. A sufficiently large Hankel matrix width and effective window length were maintained, with $\bar{N} = \hat{\bar{N}} = 400$.

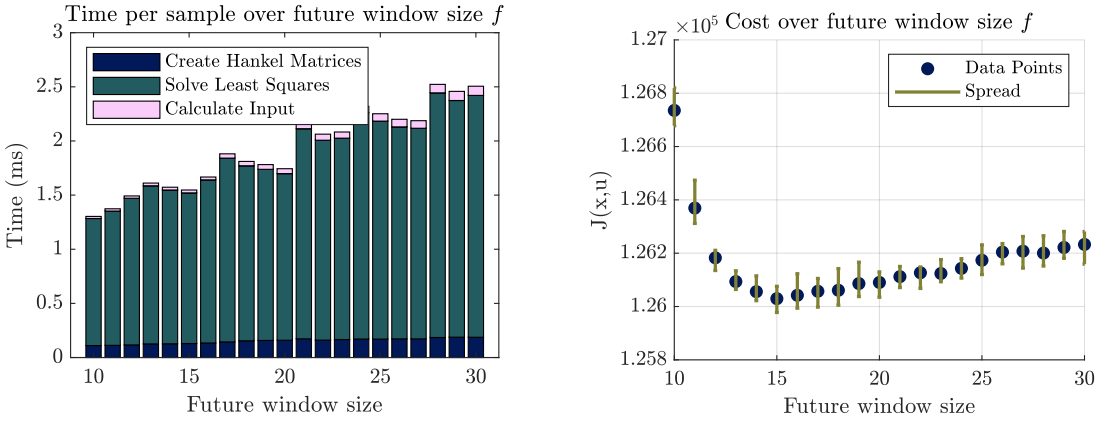
Table 4-3 summarizes the estimated computational complexity for each SPC variant across varying values of f. This table provided insight into how the prediction horizon was expected to impact the computational components in each algorithm during the simulation results analysis.

Table 4-3: Computational Complexity of different steps of different SPC Variants for $\bar{N} = 400$, where $p = 6$ for SPC, $p = 5$ for CL SPC, and $p = 2$ for R-CL SPC and CR-CL SPC. The future horizon f remains free. The not used steps are indicated with a '-' and the calculation of the first input of the CR-CL SPC is indicated with '?' due to the nonlinear way it is calculated.

	Create Hankel Matrices	Solving Least Squares	Update Covariance	Constructing Parameters	Calculate Input
SPC	$1600f + 9600$	$4f^3 + 132f^2 + 6640f + 960000$	-	-	$3f^3 + 5.5f^2 + 47.5f$
CL SPC	$1600f + 8000$	$4f^3 + 106f^2 + 6600f + 800000$	-	$10f^2 + 30f$	$3f^3 + 5.5f^2 + 39.5f$
R-CL SPC	-	-	152	$4f^2 + 9f$	$3f^3 + 5.5f^2 + 15.5f$
CR-CL SPC	-	-	152	$4f^2 + 9f$	?

In the range $10 < f < 30$, a small linear increase of approximately 10^4 FLOPs was expected for the SPC algorithm in terms of Hankel matrix creation, as given by the expression $1600f +$

9600. Solving the least squares problem, governed by a cubic expression $4f^3 + 132f^2 + 6640f + 960000$, was theoretically expected to grow—dominated by the f^3 and f^2 terms—but remained numerically around the 10^6 FLOPs level due to the large constant term. The computation of the first input was primarily driven by the f^3 term and grew rapidly with increasing f, though it remained relatively minor compared to the least squares step.



(a) Mean computational time of different algorithm components per sample as a function of future window size.

(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-9: Impact of future window size on simulation time and cumulative stage costs for the Simple SPC.

As seen in Figure 4-9a, the expected linear increase in Hankel matrix creation is clearly visible. While the least squares step was theoretically cubic, its growth trend appeared closer to linear within the tested range, likely due to the dominance of constant and linear terms. The input computation remained a smaller component but increased sharply at higher values of f.

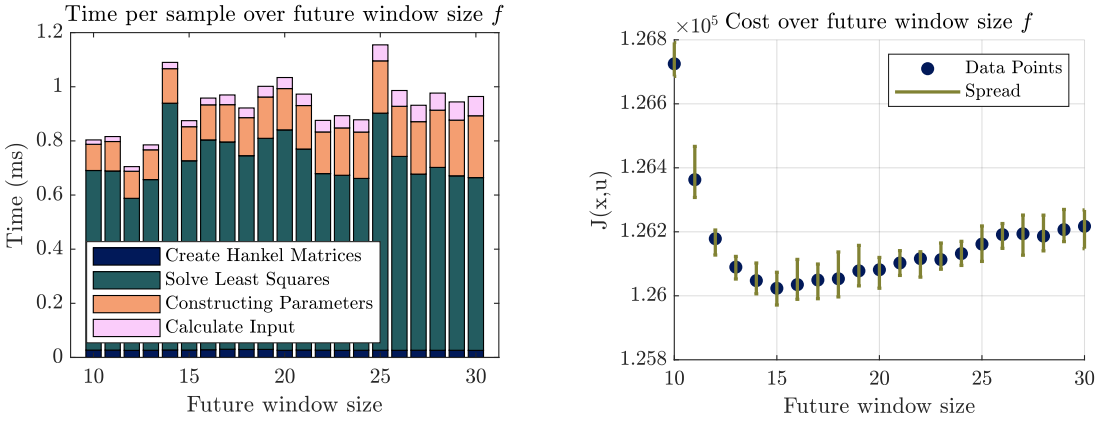
From a performance standpoint, Figure 4-9b follows a trend similar to that observed when $p = f$, with optimal performance achieved at $f = 15$. Beyond this point, performance gradually declined, suggesting that longer prediction horizons do not necessarily improve control quality.

When examining Table 4-3, in the range $10 < f < 30$, a small linear increase of approximately 10^4 FLOPs was expected for the CL SPC algorithm in terms of Hankel matrix creation, similar to the trend seen in standard SPC. The solving of the least squares problem remained the dominant computational step, with theoretical complexity scaling with f^3, and total FLOPs in the order of 10^6, this step was expected to grow rapidly. In contrast to SPC, CL SPC included an additional step for constructing innovation-form Markov parameters, which increased quadratically with f but contributed a much smaller share of the total cost. A cubic term primarily governed the computation of the first input, and while it grew quickly with f, it remained less significant than the least squares step in this range.

As shown in Figure 4-10a, the observed computational trends did not fully align with the theoretical expectations, particularly regarding the least squares component. While the Hankel

matrix creation step increased only slightly, as expected from its linear complexity, and the Markov parameter construction showed a gradual rise consistent with its quadratic scaling, the input computation exhibited a sharp increase at higher values of f, in line with its cubic dependence. However, the least squares step, despite being theoretically dominated by a cubic term, appeared to remain relatively constant across the evaluated range. This suggested that the large constant term in the expression was possibly masking the expected growth within the tested values of f.

The optimal performance achieved at $f = 15$ was similar for both SPC and CL SPC, as can be observed in Figure 4-10b and Figure 4-2b. Beyond this point, performance steadily declined. This behavior was also observed in the SPC variant. It confirmed that the added complexity of Markov parameter construction in CL SPC had a limited impact on the optimal future window size.



(a) Mean computational time of different algorithm components per sample as a function of future window size.

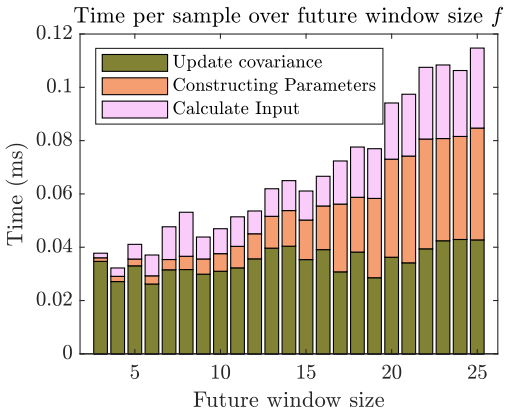
(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-10: Impact of future window size on simulation time and cumulative stage costs for the CL SPC.

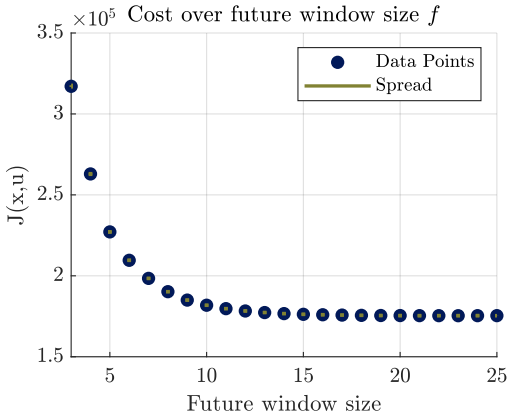
In the range $3 < f < 25$, the computational complexity of the R-CL SPC algorithm differed significantly from that of the standard SPC and CL SPC algorithms. Since recursive updating replaces explicit least squares solving, the dominant computational step is avoided. Instead, the recursive covariance update remained constant in cost, requiring only 152 FLOPs, regardless of the future horizon length. The construction of innovation-form Markov parameters increased with f^2. The computation of the first input remained driven by a cubic term in f, making its impact more prominent now that other steps were computationally more lightweight.

Figure 4-11a shows that these improvements were clearly observed. The recursive covariance update contributed to a fixed but still significant computational cost. Markov parameter construction increased quadratically with f, consistent with theoretical expectations. The cost of computing the first input grew steeply with f and became more significant in the total runtime.

Figure 4-11b reveals that the optimal future window size shifted to $f = 23$, in contrast to the fixed $f = 15$ observed in the non-recursive variants. This shift suggests that R-CL SPC benefits from longer prediction horizons.

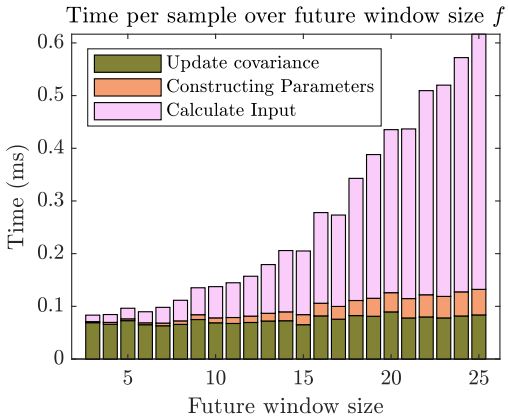


(a) Mean computational time of different algorithm components per sample as a function of future window size.

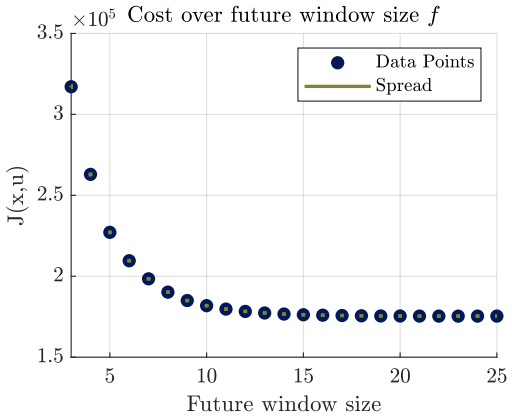


(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-11: Impact of future window size on simulation time and cumulative stage costs for the R-CL SPC.



(a) Mean computational time of different algorithm components per sample as a function of future window size.



(b) Cumulative stage costs as a function of future window size, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-12: Impact of future window size on simulation time and cumulative stage costs for the CR-CL SPC.

The computational complexity of the CR-CL SPC algorithm with respect to the future hori-

zon f closely followed that of the R-CL SPC algorithm. The recursive covariance update remained constant at 152 FLOPs, and the construction of innovation-form Markov parameters grew quadratically with f, following the expression $4f^2 + 9f$. The key difference was observed in the additional computational burden introduced by the use of a QP solver to enforce input and output constraints.

As shown in Figure 4-12a, the impact of the QP solver became increasingly evident as f grew, dominating the overall computational time. While the complexity of this step was not explicitly defined in Table 4-3, it clearly outweighed the cost of other components for larger horizons. Despite this, the recursive structure still offered substantial efficiency gains over SPC and CL SPC, which involved costly least squares computations.

From a performance perspective, Figure 4-12b shows similar behavior as the R-CL SPC algorithm, with the optimal future window size again found at $f = 23$.

4-1-4 Varying Hankel Matrix Width or Effective Window Length

The final parameter to be analyzed is the Hankel matrix width $\bar{N}$, which determines the number of past data points available for constructing the Hankel matrices. In the case of R-CL SPC and CR-CL SPC, $\bar{N}$ was not used, but instead, the values for the effective window length $\hat{\bar{N}}$ are compared.

Table 4-4 presents the estimated computational complexity for varying values of $\bar{N}$. This table showed that for both SPC and CL SPC, the dominant computational cost lay in solving the least squares problem, which scales quadratically with $\bar{N}$. In contrast, the cost of constructing Hankel matrices was expected to increase linearly with $\bar{N}$, thus contributing a significantly smaller portion to the overall complexity.

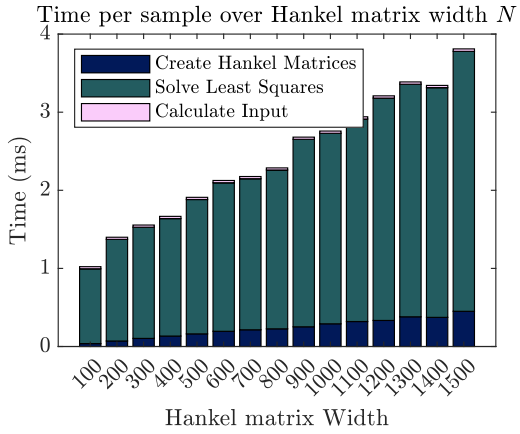
For the recursive variants, R-CL SPC and CR-CL SPC, the computational complexity was expected to remain constant with respect to $\hat{\bar{N}}$, as these methods do not rely on explicit Hankel matrices. Instead, their complexity was fixed in this comparison.

Table 4-4: Computational Complexity of different steps of different SPC Variants where $p = 6, f = 15$ for SPC, $p = 5, f = 15$ for CL SPC, and $p = 2, f = 23$ for R-CL SPC and CR-CL SPC. The Hankel matrix width $\bar{N}$ remains free. The not-used steps are indicated with '-', and the calculation of the first input of the CR-CL SPC is indicated with '?' due to the nonlinear way it is calculated.

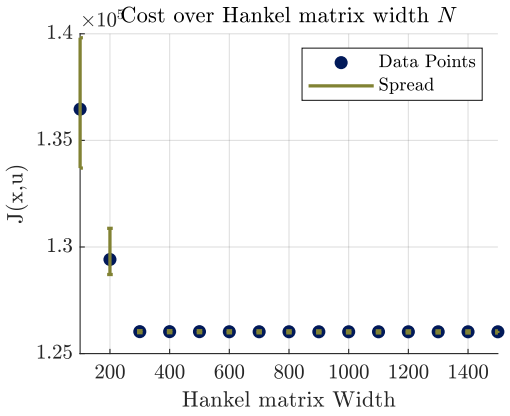
	Create Hankel Matrices	Solving Least Squares	Update Covariance	Constructing Parameters	Calculate Input
SPC	$42N$	$27N^2 + 1980N + 17208$	-	-	10845
CL SPC	$14N$	$11N^2 + 1080N + 10101$	-	1155	10687.5
R-CL SPC	-	-	344	2212.5	38503
CR-CL SPC	-	-	344	2212.5	?

When analyzing Figure 4-13a, the predicted quadratic increase in solving the least squares problem was not distinctly evident. Instead, the growth appeared to be closer to linear. The linear increase in Hankel matrix creation was also observed, although it contributed only a relatively minor portion of the overall computational cost compared to the least-squares step. Similar trends can be seen in Figure 4-14a. This figure shows that adding the Markov parameter construction introduced a modest computational overhead. Nevertheless, solving the least squares problem remained the dominant contributor to total complexity.

In terms of performance, Figure 4-13b and Figure 4-14b show that increasing $\bar{N}$ beyond a certain threshold yields only small improvements. A moderate Hankel matrix width was sufficient to capture the relevant system dynamics while avoiding unnecessary computational cost. In both SPC and CL SPC, the optimal value in terms of cumulative stage costs was found at $\bar{N} = 1400$.

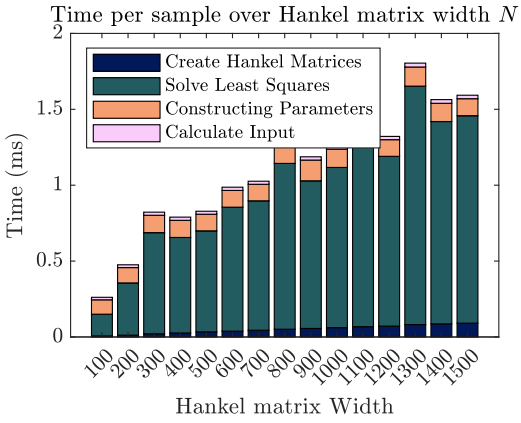


(a) Mean computational time of different algorithm components per sample as a function of Hankel matrix width $\bar{N}$.

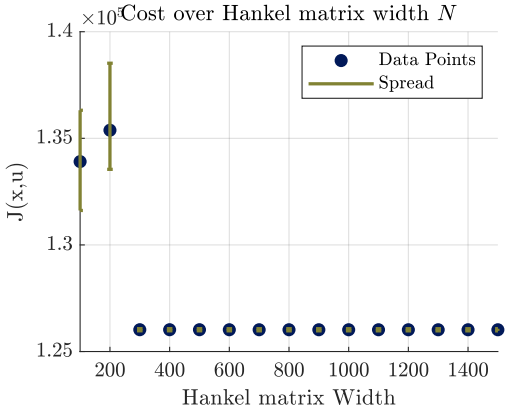


(b) Cumulative stage costs as a function of Hankel matrix width $\bar{N}$, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-13: Impact of Hankel matrix width $\bar{N}$ on simulation time and cumulative stage costs for the Simple SPC.



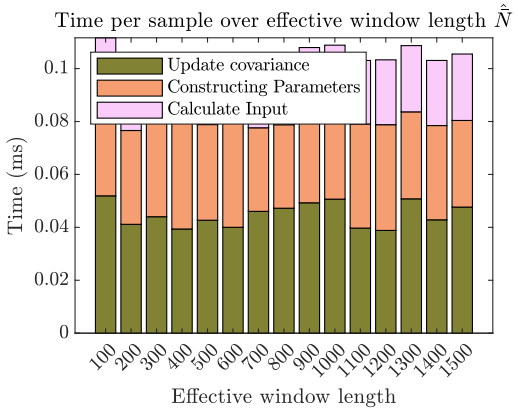
(a) Mean computational time of different algorithm components per sample as a function of Hankel matrix width $\bar{N}$.



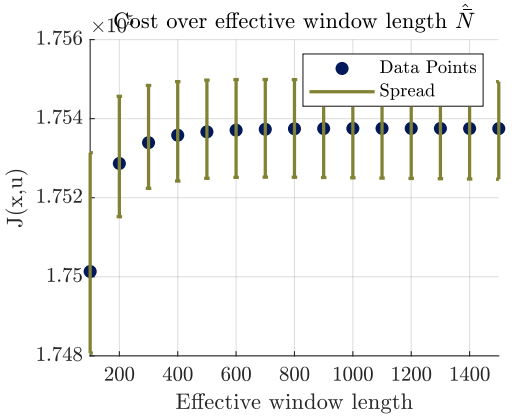
(b) Cumulative stage costs as a function of Hankel matrix width $\bar{N}$, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-14: Impact of Hankel matrix width $\bar{N}$ on simulation time and cumulative stage costs for the CL SPC.

For the recursive methods, Figure 4-15a and Figure 4-16a confirm that computational time remained nearly constant across different values of the effective window length $\hat{\hat{N}}$, as was expected from the theoretical expressions.

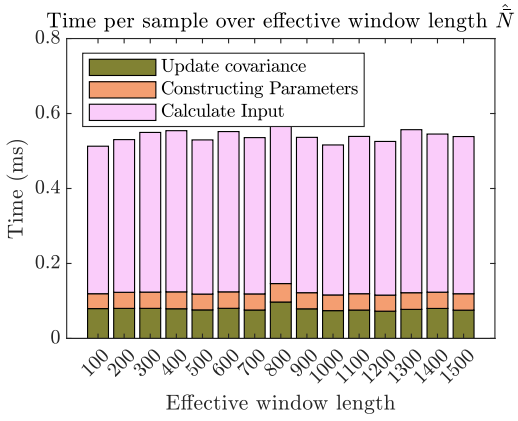


(a) Mean computational time of different algorithm components per sample as a function of effective window length $\hat{\hat{N}}$.

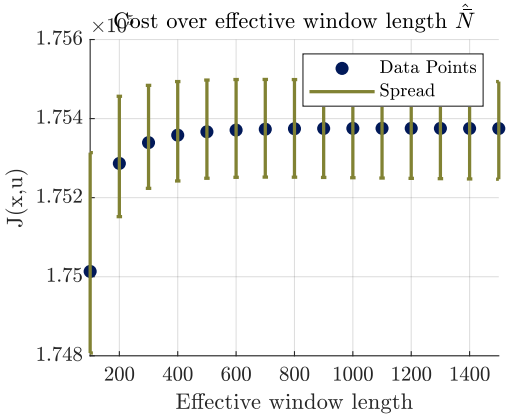


(b) Cumulative stage costs as a function of effective window length $\hat{\hat{N}}$, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-15: Impact of effective window length $\hat{\hat{N}}$ on simulation time and cumulative stage costs for the R-CL SPC.



(a) Mean computational time of different algorithm components per sample as a function of effective window length $\hat{\hat{N}}$.



(b) Cumulative stage costs as a function of effective window length $\hat{\hat{N}}$, where the spread shows the minimum and maximum value of the 10 simulations.

Figure 4-16: Impact of effective window length $\hat{\hat{N}}$ on simulation time and cumulative stage costs for the CR-CL SPC.

From a performance standpoint, the results for R-CL SPC and CR-CL SPC, shown in Fig-

ure 4-15b and Figure 4-16b, revealed an interesting and somewhat counter-intuitive trend: lower values of $\hat{N}$ resulted in better control performance.

Further concluding remarks on these results are presented at the end of this chapter in section 4-5. That section provides a summary and interpretation of the identified optimal parameter values and a discussion explaining why theoretical FLOP estimates do not always perfectly match observed computational times. Furthermore, additional findings, such as the near-equivalence in control performance between the R-CL SPC and CR-CL SPC algorithms and the significant peak in cumulative stage cost for CL SPC for lower values of $\bar{N}$ are discussed at the end of this chapter. The next section shows the comparison results between the different cost functions.

4-2 Cost function comparison

This section describes the results of the comparison between the simple and adapted cost functions. As shown in Figure 4-17, the simple cost function resulted in faster settling, reaching a steady state in approximately 1 second. In contrast, the adapted cost function required around 2 to 3 seconds to settle. However, the adapted cost function demonstrated superior steady-state behavior.

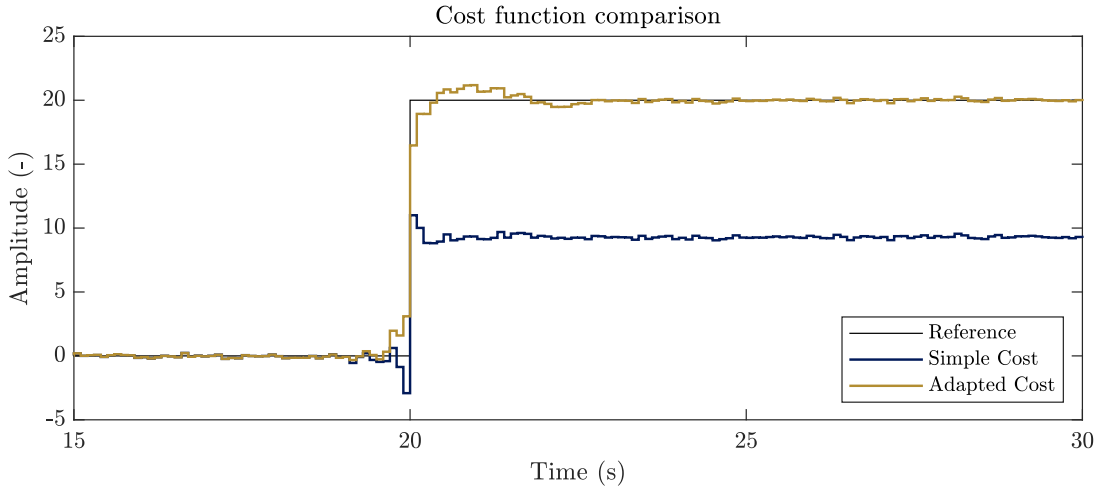
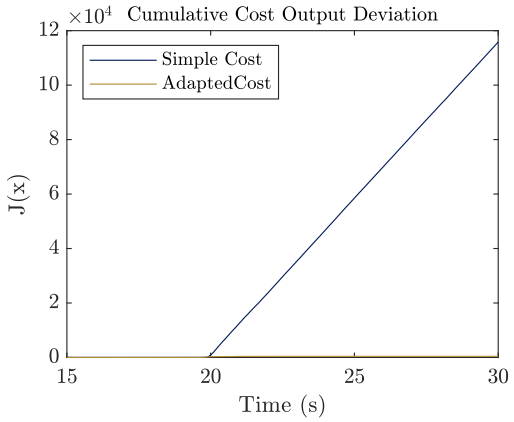
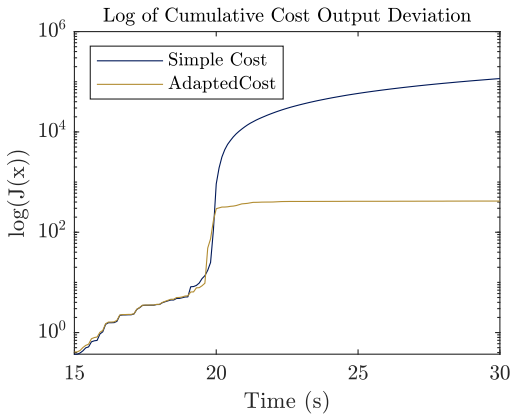


Figure 4-17: Simulation with step reference of two different cost functions, where the controller is engaged at 5s and the measurements and plot starts at 15s

The cumulative stage cost of output deviation, shown in Figure 4-18, indicated that the adapted cost function stabilized over time, whereas the simple cost function continued to increase after the step input.



(a) Cumulative cost of output deviation for the 'Simple' and 'Adapted' cost functions.



(b) Log of cumulative cost of output deviation for the 'Simple' and 'Adapted' cost functions.

Figure 4-18: Comparison of cumulative output deviation costs for the 'Simple' and 'Adapted' cost functions using (a) linear scale and (b) logarithmic scale.

Further analysis and remarks on the choice of the cost function are provided in the final section of this chapter (section 4-5). The next section, section 4-3 provides the results related to the use of the feedthrough matrix.

4-3 Use of feedthrough matrix

To show the importance of the feedthrough matrix Figure 4-19 shows that when $D = 0$ was assumed, the simulation initially struggled, then performed reasonably well during the mid-phase, and deteriorated again toward the end. In contrast, the version with $D \neq 0$ demonstrated a consistent and more robust performance. This was probably caused by the fact that the simulated system (Equation 3-1) had a direct feedthrough of $D = 0.5$. The full state space matrices are shown in Equation 4-1.

$$A = \begin{bmatrix} -0.3 & -0.4 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} -0.15 & -0.2 \end{bmatrix}, \quad D = 0.5 \tag{4-1}$$

For this reason, the configuration with $D \neq 0$ was selected for further research. Additional simulations—although yielding better results than the one presented here—also supported this decision and are presented in Appendix C-1.

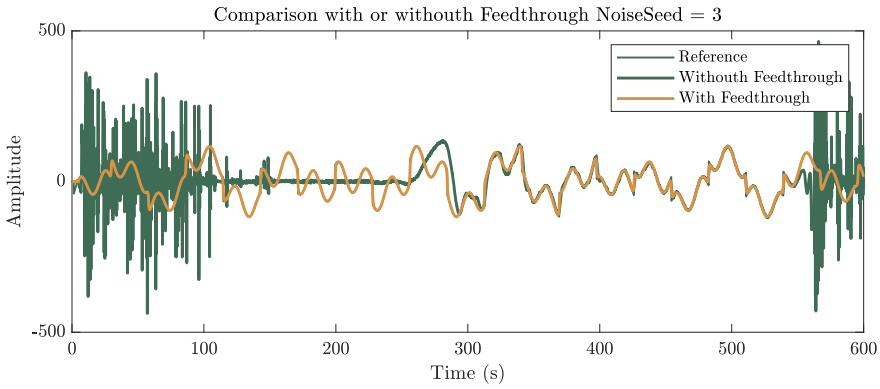


Figure 4-19: Visualization of output data for the R-CL SPC algorithm for the version where $D \neq 0$ in yellow and $D = 0$ in green.

Further analysis and remarks on the choice of using the feedthrough matrix D are provided in section 4-5.

4-4 Influence of λ_{exp}

This chapter has shown the results of parameter tuning, cost function comparison, and the use of a feedthrough matrix. The final part displays the results related to the influence of λ_{exp}. Previously, subsection 4-1-4 showed that lower values of λ_{exp} led to lower mean cumulative cost. However, Figure 4-20, which is the same simulation under increased noise with $\sigma^2 = 5$, shows that lower values of λ_{exp} were unable to converge to the correct system as quickly as previously observed, due to the increased influence of noise. The results indicate that when higher levels of noise are present, a greater amount of historical data (i.e., a higher λ_{exp}) is required to achieve accurate convergence.

In the nonlinear system simulation, as visualized in Figure 4-21, two different values of the forgetting factor, $\lambda_{\text{exp}} = 0.99$ and $\lambda_{\text{exp}} = 1$, were tested. The results indicate that after an initial phase where the system does not immediately recognize the switch, the simulation with a lower forgetting factor exhibits significantly faster reference tracking recovery than the case where $\lambda_{\text{exp}} = 1$. However, once sufficient new data was accumulated, the system with $\lambda_{\text{exp}} = 1$ converged to the reference.

Further analysis and remarks on the choice of λ_{exp} can be found in the next section.

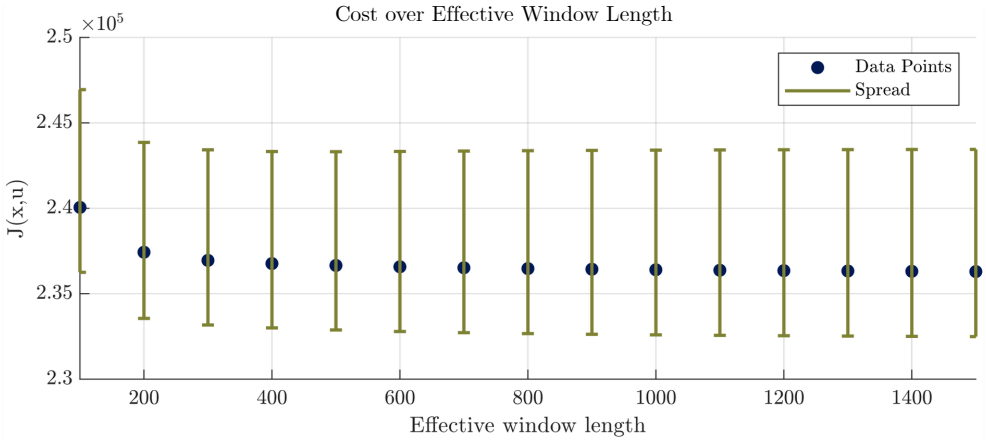


Figure 4-20: Scatter plot which shows how the effective window length influences the cumulative stage costs of the total simulation, with a higher $\sigma_v^2 = 5$, where the spread shows the minimum and maximum value of the 10 simulations.

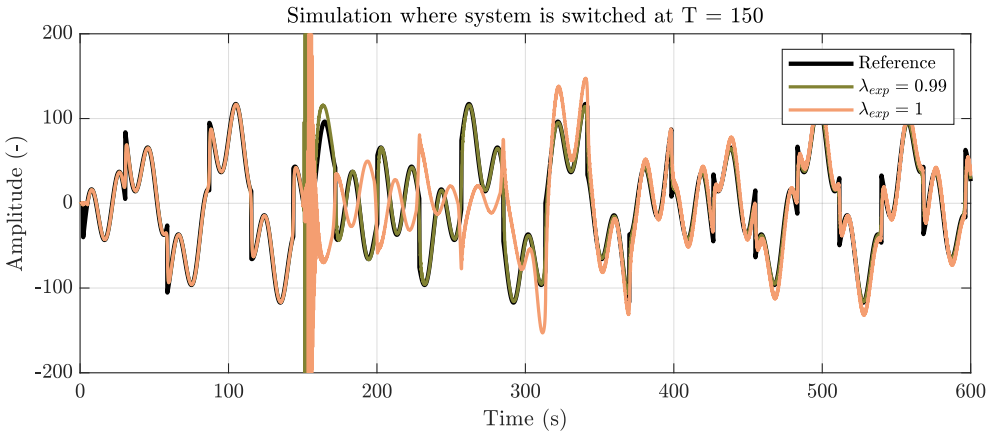


Figure 4-21: Simulation of R-CL SPC algorithm where the system transitions from $f_1(z)$ to $f_2(z)$ at $t = 150s$, where the performance for two λ_{exp} values are compared.

4-5 Concluding remarks on simulation results

This section discusses the simulation results in detail, highlighting key insights into parameter optimization, computational efficiency, and performance trade-offs across the implemented SPC algorithms. The discussion begins with an evaluation of the optimized parameters, analyzing how different choices for p, f, and $\bar{N}$ influence both control performance and computational cost.

Further analysis investigated the role of the cost function, illustrating trade-offs between reference tracking performance and control effort. The influence of the feedthrough matrix D was assessed, as well as the adaptability enabled by the exponential forgetting factor λ_{exp} in the presence of noise and nonlinearity.

Finally, this section addresses several notable observations, such as unexpected trends in the Hankel matrix width and the effect of constraint activation in CR-CL SPC. These findings offer practical insights into the implementation, adaptability, and real-time feasibility of the proposed SPC variants, reinforcing their relevance to the overarching research questions.

4-5-1 Found Optima

Table 4-5 provides an overview of the identified (near) optimal values for the SPC, CL SPC, R-CL SPC, and CR-CL SPC algorithms. However, several key observations must be highlighted when analyzing these results.

Table 4-5: Comparison of controllers with their best-performing parameter combinations, including performance metrics in terms of computational time and cumulative cost for the different SPC algorithms.

Metric	**SPC**	**CL SPC**	**R-CL SPC**	**CR-CL SPC**
Optimal p	6	5	2	2
Optimal f	15	15	23	23
Optimal $\bar{N}$ or $\hat{\bar{N}}$	1400	1400	100	100
Minimum Mean Computational Time per Sample	3.33 ms	1.40 ms	0.11 ms	0.51 ms
Minimum Mean Cumulative Cost per Simulation	1.2602×10^5	1.2602×10^5	1.7501×10^5	1.7501×10^5

The first observation drawn for this table is that simulations where the prediction horizon p was set equal to the future horizon f generally followed the same trend as those where only f was varied. This observation suggests that the choice of f has a more significant influence on performance than the choice of p. However, the decision to simulate only a second-order system may have affected these results.

Another observation is that distinct optima could be identified for the choice of f in the case of the SPC and CL SPC simulations. It was observed that increasing f beyond a certain point resulted in performance degradation. This phenomenon could be attributed to a well-known issue in traditional Model Predictive Control (MPC): The estimated model, derived

from data, is inherently imperfect. As the prediction horizon extends, the mismatch between the estimated and actual models becomes more pronounced, which can lead to a decline in control performance [81, 82].

Furthermore, a clear optimum was found for p for the R-CL SPC and CR-CL SPC simulations. This observation coincided precisely with the system order and is quite intuitive, as the system order corresponds to the order of the Auto-Regressive with eXogenous input (ARX) model implicitly estimated within the control framework. By aligning p with the system order, the algorithm effectively captured the system dynamics while avoiding unnecessary complexity.

In all other simulations, an asymptotic decrease in performance was observed. While an optimum was identified across the investigated parameter ranges, further increases in these parameters did not yield significant improvements beyond a certain point. This trend suggests difficulty in quantifying the trade-off between computational effort and performance. Although selecting the parameter values that minimize cumulative stage cost may seem optimal, practical considerations regarding computational efficiency must also be considered. Beyond a certain threshold, increasing computational burden without substantial performance gains may not be justified.

4-5-2 FLOPs

In some simulations, the estimated amount of FLOPs was not in line with the results, especially if large matrix-matrix multiplications were used. For example, in the test where the Hankel matrix width was varied for the SPC algorithm in subsection 4-1-4. A $\bar{N}^2$ increase was expected, but only a $\bar{N}$ was shown. The reason is that the number of FLOPs does not always directly correspond to the computational time.

This can be explained by the notion that most FLOPs are executed using optimized Basic Linear Algebra Subprograms (BLAS) libraries since the introduction of LAPACK/BLAS in Matlab in 2000 [83]. These libraries make use of techniques such as Strassen's method [84] for fast matrix multiplication, which becomes particularly advantageous for larger matrices (matrix width > 100) [85]. Additionally, BLAS libraries are tailored to specific hardware architectures and are designed to optimize performance through techniques like cache and register blocking and the use of Single Instruction, Multiple Data (SIMD) instructions [86].

Due to these optimized BLAS libraries, the simulation time is more strongly influenced by memory references and cache utilization than by the number of FLOPs [83]. As a result, the number of FLOPs alone cannot fully represent the computational time on modern architectures, especially for larger matrix multiplications used in this research. For this reason, the tested computational times are leading.

4-5-3 Choice of cost function

The comparison between different cost functions in the simulations highlighted important trade-offs in control performance. Results indicate that the adapted cost function consistently outperformed the simple cost function regarding steady-state error. This difference arose because the simple cost function seeks an optimal balance between minimizing the output

error and input magnitude. Consequently, this optimization did not necessarily lead to the complete elimination of steady-state error, which could be preferable if small offsets in the output may be preferred over large control efforts.

One notable drawback of the adapted cost function is that it does not penalize the absolute value of the input directly. As a result, in scenarios where a significant offset is present, the control input can theoretically increase indefinitely. In practical applications, this behavior is undesirable, as it may lead to excessively large control actions, which could be infeasible or even damaging to the system.

Interestingly, the issue of steady-state error does not arise when tracking a reference of zero. In such cases, the steady-state condition naturally aligned with the point where both the input and output reached zero, corresponding to the cost function's minimum. This ensures that the simple cost function performs adequately in terms of steady-state error under these conditions.

The observations, as demonstrated in Figure 4-17, conclude that the adapted cost function is generally preferable when precise reference tracking is the primary objective. However, when tracking a zero reference, where steady-state errors are not an issue, and minimizing the magnitude in control effort is also desired, the simple cost function may be the better choice. It inherently avoids excessive control actions while still maintaining stable performance.

4-5-4 Choice of feedthrough matrix

Including a feedthrough matrix D in the algorithm significantly influenced its performance, particularly in cases where the system exhibited this direct input-output coupling. The results comparing simulations with and without a feedthrough matrix in Figure 4-19 highlight a clear advantage when $D \neq 0$. This improvement arose due to the presence of a significant feedthrough term ($D = 0.5$) in the simulated system. When the feedthrough term was omitted ($D = 0$), the algorithm incorrectly assumed that input at a given time step had no immediate effect on the system. Consequently, the control strategy failed to account for the direct influence of the input at that specific time step. Instead, it only considered its effect on subsequent time steps. This resulted in suboptimal control decisions and a degradation in performance.

Conversely, assuming a nonzero feedthrough term when the system has $D = 0$ can also lead to issues. If the algorithm incorrectly estimates D, it may introduce significant overcompensation or undercompensation in the resulting control actions. However, this potential problem is naturally mitigated by the algorithm's ability to infer $D \approx 0$ when the true system exhibits no direct feedthrough. This estimation capability minimizes significant errors in control performance due to an incorrectly assumed feedthrough term.

These findings suggest that unless it is definitively known that $D = 0$, it is generally preferable to use the version of the algorithm where $D \neq 0$. Including the feedthrough matrix introduces only a limited computational overhead while providing robustness against incorrect assumptions. This ensures that the control algorithm can effectively handle cases where direct input-output interactions exist, leading to improved performance and reliability.

4-5-5 Choice of λ_{exp}

The choice of the exponential forgetting factor, λ_{exp}, is crucial in determining the algorithm's performance under different noise conditions and system characteristics.

In the initial simulations, where the noise variance was set to $\sigma_v^2 = 0.01$, a clear trend was observed: lower values of λ_{exp} led to better performance. This indicates that when noise is minimal, the system requires only a limited amount of historical data to estimate the system dynamics accurately and remove the issues of the initial guess which is wrong. As a result, a lower forgetting factor allows the algorithm to react more quickly to new information, leading to improved control performance.

However, when the noise variance was increased to $\sigma_v^2 = 5$, a more intuitive asymptotic reduction in performance was observed as λ_{exp} increased. This suggests that more historical data is required to obtain a reliable system estimation in the presence of significant noise. The algorithm must rely on a longer history of past data to mitigate the noise, which explains partly why higher values of λ_{exp} are beneficial under noisy conditions. However, further research is necessary to assess robustness under higher noise conditions.

For the nonlinear case, it was demonstrated that when $\lambda_{\text{exp}} = 1$, the algorithm could still adapt if enough relevant new historical data was introduced. However, the adaptation process occurred with diminished speed compared to when $\lambda_{\text{exp}} = 0.99$. This observation highlights a fundamental trade-off: while a higher forgetting factor retains more past information and ensures stability, it reduces the rate at which the algorithm can respond to changes in the system.

Furthermore, in scenarios where $\lambda_{\text{exp}} = 1$, it is expected that the longer the system operates under one configuration (e.g., system A), the longer it will take to adapt when transitioning to a different configuration (e.g., system B). In contrast, with $\lambda_{\text{exp}} = 0.99$, the adaptation process will occur over a more consistent timespan, ensuring that the algorithm remained responsive to system changes.

These findings confirm that choosing a higher value for λ_{exp} or even setting $\lambda_{\text{exp}} = 1$ is generally preferable, especially when dealing with a system that is known to be Linear Time-Invariant (LTI) and does not undergo further changes. However, higher values enhance stability and long-term accuracy but also reduce adaptability. If the system is expected to evolve over time, lower values of λ_{exp} should be considered to enable more rapid adaptation. Further research is required to understand this balance better and potentially optimize it for certain system properties.

4-5-6 Notable Results

The simulation results reveal an intriguing observation concerning CL SPC and the influence of the Hankel matrix width ($\bar{N}$) on the cumulative stage cost. A distinct peak in cumulative cost was detected when $\bar{N}$ was set between 200 and 400. This suggests that the algorithm performs better for very low values of $\bar{N}$ than for slightly higher values. This phenomenon was consistently observed across multiple simulations where different noise seeds were applied. While the underlying cause of this behavior needs more research, it is important to note that larger values of $\bar{N}$ yielded even better results. Since higher values did not exhibit this anomaly

and generally led to improved performance, no further investigation was conducted into this effect.

Furthermore, the performance outcomes for the CR-CL SPC algorithm were equivalent to those of the R-CL SPC. The results can be attributed to the input constraints not being reached during the simulations, combined with the use of identical noise seeds were used for both methods. In practical applications where input constraints are rarely exceeded or not exceeded, the use of R-CL SPC in combination with a simple saturation mechanism may suffice.

However, if the control limits had been reached, the CR-CL SPC algorithm would have actively enforced these constraints, leading to a more complex optimization problem at each time step. This would result in an increased computational burden, as the QP solver (ForcesPro [78, 79]) would need to handle additional constraints, extending the computation time per iteration. Consequently, real-time feasibility could be affected even more, particularly in systems with stringent timing requirements.

This suggests that, unless strict enforcement of constraints is required, the additional computational complexity introduced by CR-CL SPC may not be necessary, making R-CL SPC a more efficient alternative in such cases.

Chapter 5

Piezo Actuated Beam Setup

The preceding chapters have presented various Subspace Predictive Control (SPC) algorithms, both theoretically and through simulations. These results now form the basis for a real-life experiment using a piezo-actuated beam as the test system. This chapter starts with a detailed description of the piezo-actuated beam setup, including both the hardware components and the software framework used to enable real-time controller deployment. The second part focuses on model estimation and potential internally stable controller, establishing a baseline model for controller performance analysis. This was achieved using the built-in System Identification (SysID) capabilities of the Recursive Closed Loop Subspace Predictive Control (R-CL SPC) algorithm. While similar functionality exists in other SPC variants, this algorithm was selected due to the best computational feasibility needed for real-time control as described in chapter 4. The final part of this chapter presents an analytical Closed Loop (CL) stability assessment to identify which controllers can operate on the experimental setup. This chapter is critical for addressing the research questions related to real-world implementation.

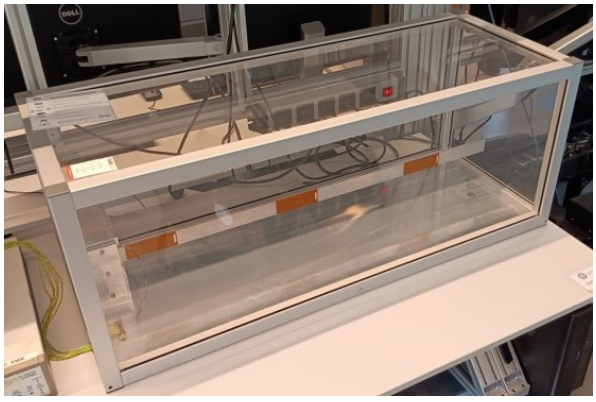


Figure 5-1: Photograph of the piezo actuated beam, which is encased in a glass box due to safety measures

5-1 What is a Piezo Actuated Beam (Hardware)

The piezo-actuated beam is a one-sided clamped metal beam approximately 1 meter long, encased in a glass box for safety reasons, with three pairs of piezos along its length, as shown in Figure 5-1. The piezos are connected to an amplifier that amplifies the signals of the dSPACE MicroLabBox to the needed voltages. This section describes the setup of the piezos, amplifier, piezo I/O box, and dSPACE MicroLabBox, which form the hardware of the beam. The final part of this section describes the beam's dynamic behavior.

Piezos

In this setup, the actuators and the sensors are flexible Macro Fiber Composite (MFC) devices, type M8528, from Smart Material Corp. The piezos have an operational range of -500 to 1500 Volts and can be used as actuators or sensors. The piezos respond to electrical inputs by mechanically deforming, introducing a bending moment. This enables control over mechanical movement. When used as a sensor, the piezo has an operation range of -10 to 10 Volts and has a measurement noise assumed to be white noise with variance $\sigma^2_{piezo} = 3 \times 10^{-6}$. A photograph of a piezo in shown in Figure 5-2.

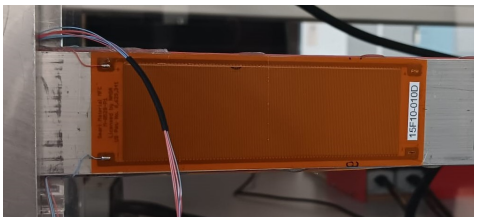


Figure 5-2: Photograph of a piezo used in this thesis, the beam has three pairs of these piezos

Amplifier

The system amplifier can output voltages from -700 to 700 volts. Therefore, it is essential to scale the low-voltage signals of the dSPACE MicroLabBox up to a level that effectively activates the piezo.

Figure 5-3: Photograph of the amplifier used in this thesis.

Because the piezo has an operational range of -500 to 1500 V and the amplifier can output -700 to 700 V, the inputs were constrained between -500 and 700 V. A photograph of the amplifier is shown in Figure 5-3

Piezo I/O box

To connect the amplified signals to the piezo actuators and the signals of the piezo sensors to the data acquisition system, a piezo Input-Output (I/O) box was used. It is an interface between the amplifier, piezos, and data acquisition system, ensuring signal transmission and simplifying the wiring. A photograph of the I/O box is shown in Figure 5-5.

Figure 5-4: Photograph of the piezo I/O box used in this thesis.

dSPACE MicroLabBox

The dSPACE MicroLabBox is a compact, real-time data acquisition system used primarily in laboratory settings for rapid control prototyping. It offers I/O capabilities and provides the computing power of this setup.

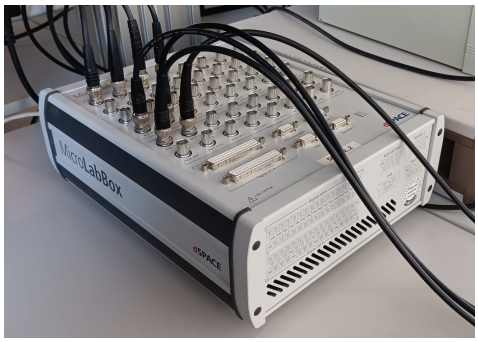


Figure 5-5: Photograph of the MicroLabBox of dSPACE used in this thesis.

5-1-1 Beam Dynamics

This subsection explores the dynamic behavior of the piezo-actuated beam, including how the piezo actuators influence the beam's movement and stability while also considering its resonant frequencies.

A flexible beam can exhibit nonlinear behavior under large applied forces [87]. However, it is assumed that the forces in the setup are sufficiently small to keep the material within the linearly elastic deformation range [88].

In theory, the beam exhibits an infinite number of vibration modes, with the lowest-frequency modes typically having the least intrinsic damping [50]. This intrinsic damping means that the internal transfer of kinetic energy to other forms of energy is higher for higher vibration modes, resulting in faster (autonomous) damping [89]. Therefore, the most interesting modes to look into are the lower vibration modes, which are shown in Figure 5-6.

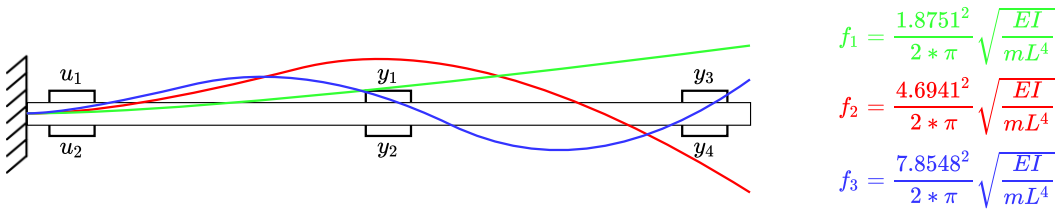


Figure 5-6: Schematic top view of the beam configuration with the associated natural vibration modes [90], where it is shown that the two input channels (u_1 and u_2) are located close to the clamped side. The first two output channels (y_1 and y_2) are located approximately at the middle and the last two outputs (y_3 and y_4) at the free side of the beam. Furthermore all the even numbers are on one side and the odd numbers on the other. Inspired by [50].

The beam that was used for this research had approximately the following properties:

- $L = 0.95$ m (Length of the beam)
- $B = 0.035$ m (Width of the beam)
- $D = 0.004$ m (Thickness of the beam)
- $E = 69$ GPa (Young's modulus of aluminum)
- $\rho = 2700$ kg/m^3 (Density of aluminum)
- $I = \frac{BD^3}{12} = 2.00 * 10^{-10}$ m^4 (Inertia of rectangle [90])
- $m = \rho LBD = 0.359$ kg (Mass of the beam)

Given these properties, the first three natural vibration modes are approximately $f_1 = 3.71$ Hz, $f_2 = 23.27$ Hz, and $f_3 = 65.16$ Hz.

5-2 How does the Piezo Actuated Beam Work (Software)

In addition to the hardware of the beam, software was used to control its operation, which is described in this section. The software consists of Simulink and dSPACE. In addition to those, the QP solver is described.

5-2-1 Simulink

Simulink was used to design and simulate the control systems for the piezo-actuated beam. It provided a MATLAB based graphical environment for modeling, simulating, and analyzing dynamic systems, enabling a clear understanding of the system's response.

5-2-2 dSPACE

The dSPACE software interfaces seamlessly with the MicroLabBox to efficiently deploy the compiled code of the Simulink models onto real-time hardware. This integration allows for direct manipulation and observation of the beam's behavior under controlled conditions, enabling iterative development and refinement of the control system.

5-2-3 Quadratic Programming (QP) solver

ForcesPro [79, 78] was employed as a custom optimization solver to address the QP problems that arise in the control applications for the piezo-actuated beam. It provides a fast and reliable means to optimize control actions based on predefined criteria and constraints. This solver is particularly important for achieving input and output constraints and is supported by an engineering license and a software testing license to work with the dSPACE environment.

5-3 Data-Driven Model Estimation

All the analyzed algorithms in this thesis are characterized by their ability to estimate models at each time step rather than relying on fixed models. This section investigates the model estimation of the R-CL SPC algorithm. In this section, several Auto-Regressive with eXogenous input (ARX) models are compared with one another and with the ground truth, which is represented by the frequency response function. The objective is to identify a baseline model that can subsequently be used for stability analysis.

5-3-1 Frequency Response function

To determine the frequency response function that would serve as the ground truth for evaluating the model estimations, a white noise signal with a variance of $\sigma^2 = 10^5$ and a duration of 100 seconds was applied to input u_1, using a sampling time of 0.005 seconds. The frequency response function was constructed using the measured data from u_1 and y_1 (see Figure 5-6) and is shown in Figure 5-7. The resulting frequency response aligns with the calculated natural frequencies based on the beam's properties as shown in subsection 5-1-1. The experimentally identified natural frequencies of the real system corresponded well, although slightly higher with values at $f_1 = 3.78\,\text{Hz}$, $f_2 = 23.97\,\text{Hz}$, and $f_3 = 66.86\,\text{Hz}$.

5-3-2 Estimated ARX model

To compare the ground truth with the model estimate, a model representation must first be defined. Therefore, an ARX model was constructed with the estimated the predictor-form Markov parameters. The ARX representation was chosen to facilitate analysis and comparison within a familiar polynomial framework, enabling the description of the system as a rational transfer function [91]. The resulting ARX model is presented in Equation 5-1, and the full derivation can be found in Appendix B-3.

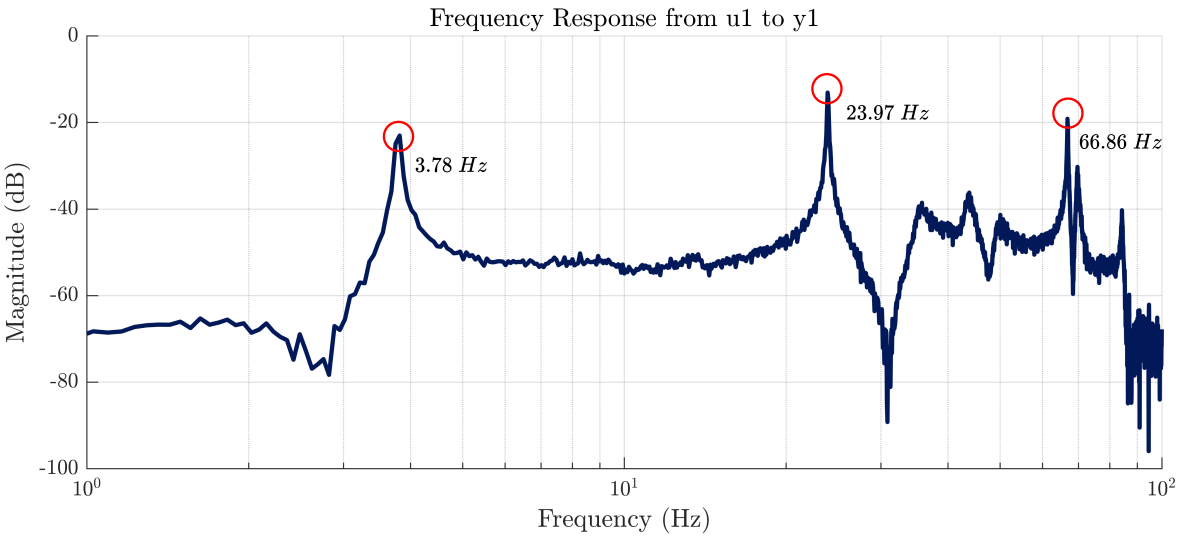


Figure 5-7: Frequency response from the first input to the first output of the real system, With peaks shown on $f_1 = 3.78\,\text{Hz}$, $f_2 = 23.97\,\text{Hz}$, and $f_3 = 66.86\,\text{Hz}$, corresponding to the natural frequencies of the piezo actuated beam setup.

$$y(z) = \frac{D + C\tilde{B}z^{-1} \ \cdots \ + C\tilde{A}^{p-1}\tilde{B}z^{-p-1}}{1 - CKz^{-1} \ \cdots \ - C\tilde{A}^{p-1}Kz^{-p-1}}u(z) + \frac{1}{1 - CKz^{-1} \ \cdots \ - C\tilde{A}^{p-1}Kz^{-p-1}}e(z) \quad (5\text{-}1)$$

5-3-3 ARX Model estimations

To adjust the performance of the ARX model estimation, two key parameters can be tuned: the exponential forgetting factor λ_{exp} and the model order, which is implicitly defined by the past window size p. This section analyzes these parameters' influence on the model estimation quality.

Exponential Forgetting Factor

A real-life experiment was conducted to analyze the influence of the exponential forgetting factor. A 100-second Pseudo-Random Binary Sequence (PRBS) signal with an amplitude of 200 was applied to the first input u_1 of the system in an open-loop configuration with the controller deactivated, and the output signal of the first output signal y_1 was used (see Figure 5-6). The impact of six different values of the exponential forgetting factor was investigated, while the model order parameter p was kept fixed at 20.

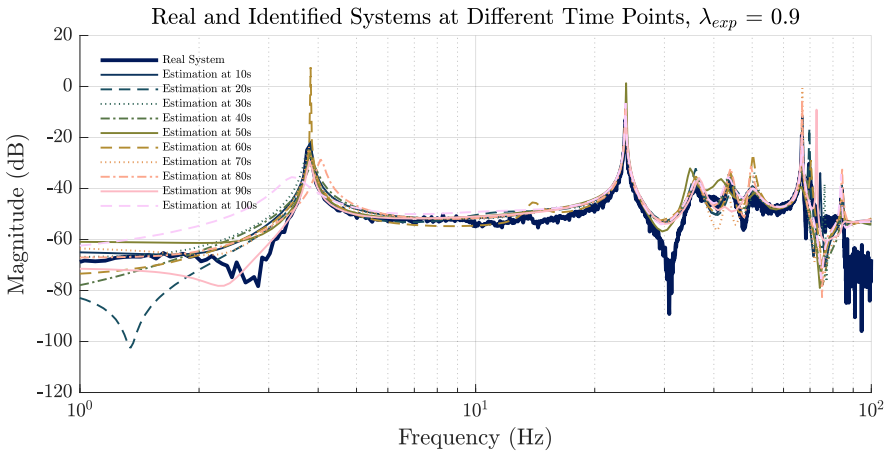


Figure 5-8: Comparison of ground truth Frequency Response Function and estimated systems when $\lambda_{\text{exp}} = 0.9$ at different points in time

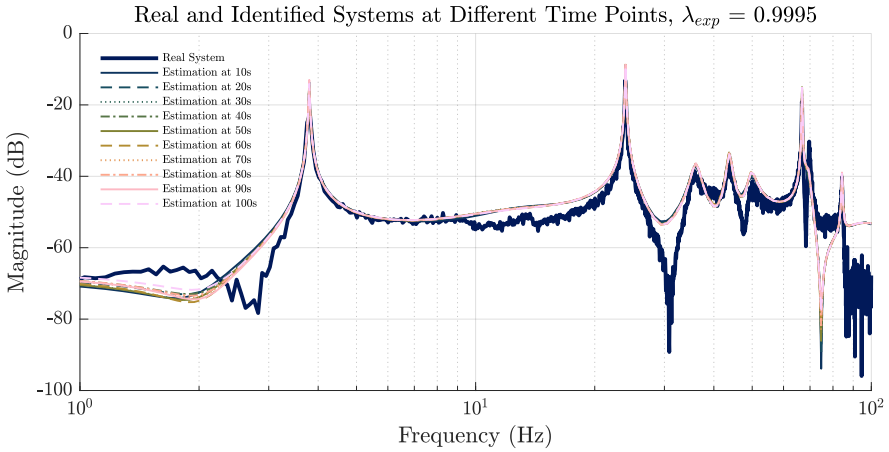


Figure 5-9: Comparison of ground truth Frequency Response Function and estimated systems when $\lambda_{\text{exp}} = 0.9995$ at different points in time

As anticipated, lower values of $\lambda_{\text{exp}} < 0.995$ resulted in significant variations in the estimated system models, as shown in Figure 5-8. In contrast, for values greater than $\lambda_{\text{exp}} > 0.9995$, the system estimates exhibited high consistency, as illustrated in Figure 5-9.

While a lower λ_{exp} enhances adaptability, which is an advantage in time-varying or nonlinear systems, this benefit is less relevant in the present case, as the system is approximately Linear Time-Invariant (LTI). Therefore, ensuring that enough historical data is retained to satisfy the Persistance of Excitation (PE) conditions became the primary objective. Based on this reasoning, a value of $\lambda_{\text{exp}} = 0.99995$ was selected.

ARX Model Order

Another parameter that was tuned is the ARX model order, implicitly defined by p. To tune this parameter, the same experiment as described previously was conducted. In this experiment, a 100-second PRBS signal with an amplitude of 200 was applied to the system, and the first input u_1 and output y_1, as shown in Figure 5-6, were considered. In this case, however, the exponential forgetting factor was fixed at the previously selected value of $\lambda_{\text{exp}} = 0.99995$, and different values of p were tested.

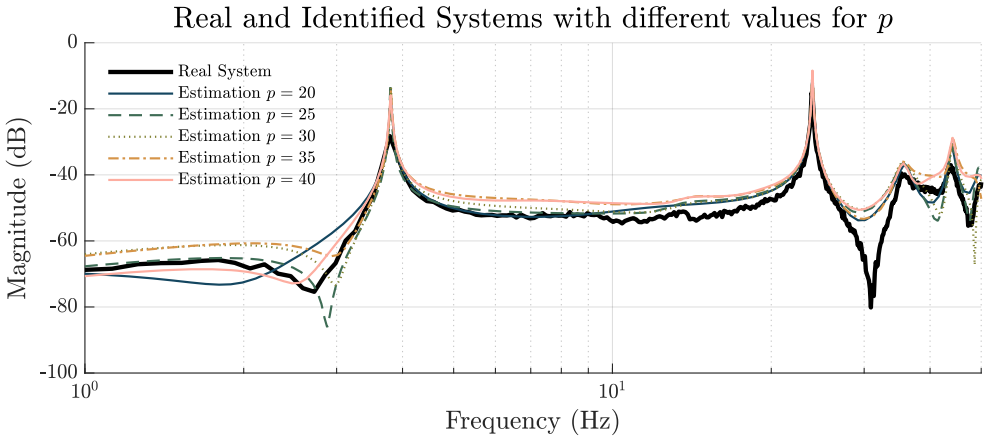


Figure 5-10: Comparison of real and estimated systems with different values for the past window size p, obtained after 100 seconds.

Figure 5-10 presents the estimation results for $p = 20, 25, 30, 35$, and 40 at the 100-second mark in a single plot. Additional figures illustrating system estimates at various time steps, as well as results for lower model orders $p = 5, 10$, and 15, are provided in Appendix C-2.

These results demonstrate the successful identification of the first two natural frequency peaks. However, variations can be observed in the lower and higher frequency regions, indicating the estimation's sensitivity to the chosen model order. From these results, it is difficult to quantify definitively which model order yielded the best overall performance. Therefore, all considered model orders were retained for further analysis.

The following chapter presents a detailed stability analysis of the analytical controller. To determine the appropriate baseline model in terms of the past window size p, the configuration yielding the most consistently stable internal controllers was identified. In addition, the influence of key parameters, including the future window size f, the penalty on output deviation Q, and the penalty on control effort R, was investigated to understand their impact on overall CL stability and lastly some frequency and time domain experiments are performed.

Chapter 6

Experimental Results: Subspace Predictive Vibration Control of Piezo Actuated Beam

Real-life experiments were conducted on the setup described in chapter 5, still considering $p = 20, 25, 30, 35$, and 40. To find the best value for p and, therefore, the best baseline model. The model order with the most internally stable controllers of the Recursive Closed Loop Subspace Predictive Control (R-CL SPC) algorithm is chosen in the first part of this chapter. In the second part, Closed Loop (CL) stability of the internally stable controllers together with the found baseline model is analyzed. Finally, this chapter presents both time-domain and frequency-domain results, beginning with a description of the experimental methodology followed by the results.

The experiments serve as a theoretical and practical validation of the proposed controllers, confirming their stability and assessing their effectiveness in damping the first two vibration modes. These results directly contribute to identifying the best-performing controller and support the research goals related to real-world implementation and performance comparison.

6-1 Controller Tuning and Analysis

For the analytical solution in the R-CL SPC algorithm, the controller can be characterized with an Auto-Regressive with eXogenous input (ARX) model, which forms the focus of this section. To gain a comprehensive understanding of the controller's behavior, the resulting ARX model of the controller was analyzed. The simple cost function was chosen, as the objective is to track a zero reference. This choice resulted in no steady state error, as discussed in subsection 4-5-3, and therefore, the analysis focused on varying the tuning parameters Q, R, and the prediction horizon f. The control penalty was set to $R = 1$ everywhere, as the relative weighting between Q and R was most influential in the controller's behavior.

Controller ARX model

The formula of the controller for the future input sequence was inferred from the analytical solution. Starting from Equation 2-21, the expression used to calculate the input sequence is given in Equation 6-1:

$$U_{\hat{i}_p,f,1} = -\left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q} \Gamma \mathcal{K} w_p \tag{6-1}$$

From this expression, a ARX model representation of the controller can be derived, as shown in Equation 6-2. Here, the matrix α is defined as:

$$\alpha = -\left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q} \Gamma \mathcal{K}$$

The MATLAB indexing convention was used to reference the elements of α. The full derivation is provided in Appendix B-4.

$$u(z) = \frac{\alpha(1,p)z^{-1} + \cdots + \alpha(1,1)z^{-p}}{1 - \alpha(1,2p)z^{-1} - \cdots - \alpha(1,p+1)z^{-p}} y(z) + \frac{1}{1 - \alpha(1,2p)z^{-1} - \cdots - \alpha(1,p+1)z^{-p}} e(z) \tag{6-2}$$

Controller Structure

Figure 6-1 illustrates that the controller consists of two distinct components: a part that depends on the last p values of the output y, denoted by G_y, and a part that depends on the previous p values of the input u, denoted by G_u. These two components are mathematically defined in Equation 6-3 and Equation 6-4, respectively.

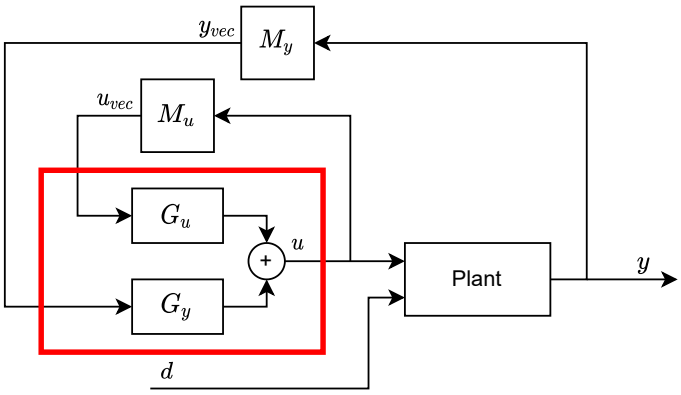


Figure 6-1: Block diagram of the SPC controller. The red box represents the controller, which includes a component dependent on the last p values of y (G_y) and a component dependent on the last p values of u (G_u).

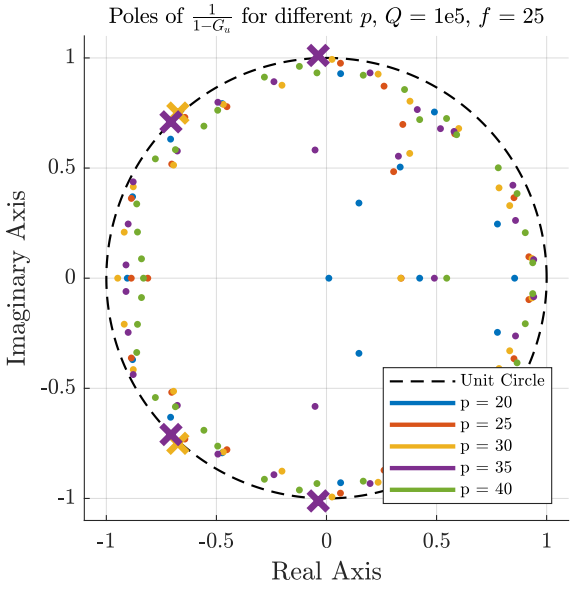
$$G_y = \alpha(1,p)z^{-1} + \cdots + \alpha(1,1)z^{-p} \tag{6-3}$$

$$G_u = \alpha(1, 2p)z^{-1} + \cdots + \alpha(1, p+1)z^{-p} \tag{6-4}$$

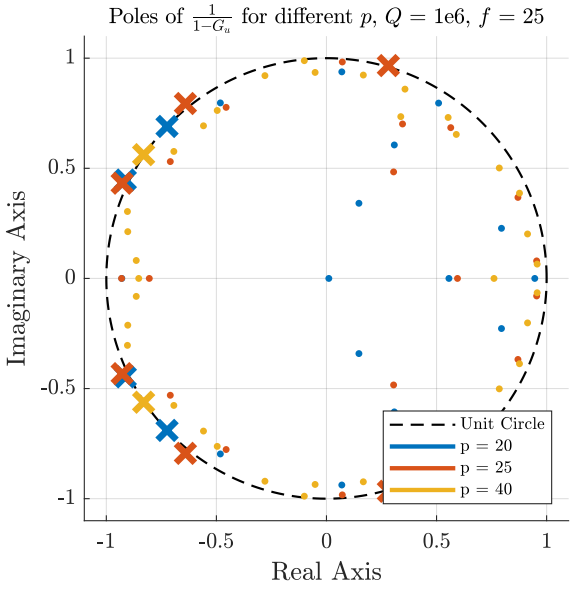
This structure reveals a feedback loop over the past values of u, which may have introduced internal instability in the controller. In the following part of this section, the poles of $\frac{1}{1-G_u}$ will be analyzed to ensure the internal stability of the controller.

6-1-1 Internal Stability of Controllers

As discussed before, the controller's ARX model can be used to analyze its internal stability. This part of the section evaluates the stability characteristics of various controller configurations, which are defined by the parameters p, f, Q, and R, by explicitly examining the poles of $\frac{1}{1-G_u}$. The models after 100 seconds of Pseudo-Random Binary Sequence (PRBS) signal with amplitude 200, and $\lambda_{\text{exp}} = 0.99995$ were considered.



(a) Map of the poles of $\frac{1}{1-G_u}$ for different past window sizes p, with fixed $f = 25$, $Q = 1 \times 10^5$ ($R = 1$). The crosses mark unstable poles. (Configurations with $p = 30$ and 35 have unstable poles)



(b) Map of the poles of $\frac{1}{1-G_u}$ for different past window sizes p, with fixed $f = 25$, $Q = 1 \times 10^6$ ($R = 1$). The crosses mark unstable poles. (All onfigurations have unstable poles)

Figure 6-2: Poles of $\frac{1}{1-G_u}$ for varying p, with fixed $f = 25$ and $R = 1$. (a) $Q = 10^5$, some stable controllers. (b) $Q = 10^6$, all unstable.

Figure 6-2a shows the stability analysis of the controller for various past window sizes p, using a fixed future horizon $f = 25$ and output penalty $Q = 1 \times 10^5$. The results reveal that the controller with $p = 30$ exhibited one unstable pole pair, while the controller with $p = 35$ showed two unstable pole pairs located outside the unit circle. All other configurations remained stable. This suggests that setting $Q = 1 \times 10^5$ (and $R = 1$) is a reasonable starting point for the output penalty in the R-CL SPC framework.

Further examination of the stable controllers as shown in Figure 6-2b for $p = 20, 25$, and 40 shows that increasing Q to 1×10^6 caused all three controllers to become unstable. This highlights that excessive penalization of output deviations, although potentially improving performance, can lead to degraded controller behavior due to internal instability.

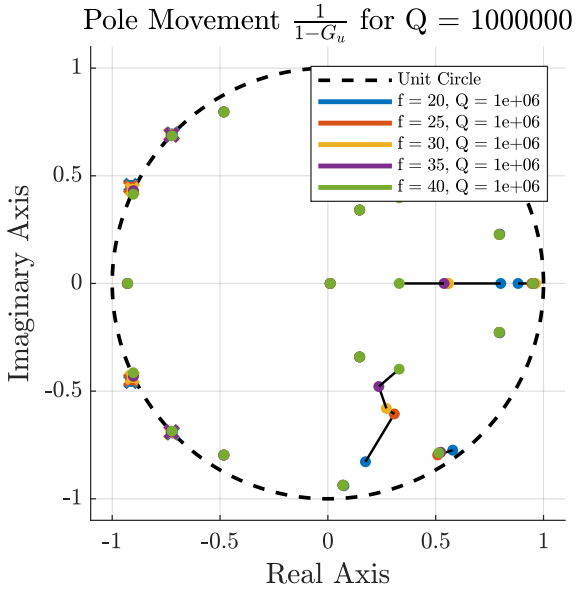
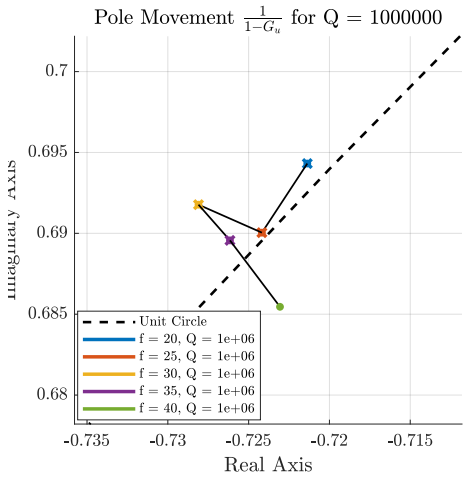
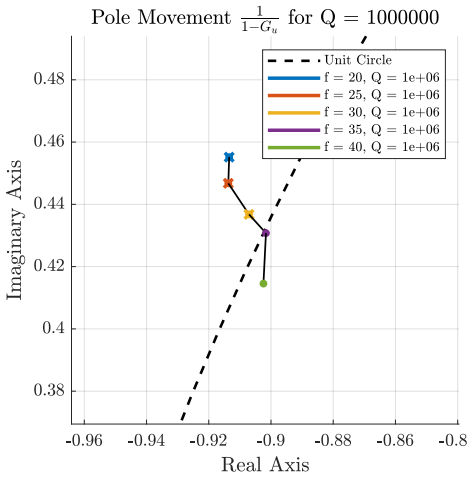


Figure 6-3: Pole movement of controllers for varying future window f, with $p = 20$ and $Q = 1 \times 10^6$. The crosses mark unstable poles. Stability is regained at $f = 40$.



(a) Zoom on the first unstable pole, which becomes stable at $f = 40$.



(b) Zoom on the second unstable pole, which becomes stable at $f = 35$.

Figure 6-4: Zoomed-in view of the unstable poles of $\frac{1}{1-G_u}$ for varying future window f, with $p = 20$ and $Q = 1 \times 10^6$. Where the unstable pole in (a) becomes stable when $f = 40$ and the unstable pole in (b) becomes stable at $f = 35$.

Figure 6-3 explores the effect of increasing the prediction horizon f, particularly for the unstable configuration with $p = 20$ and $Q = 1 \times 10^6$. The results indicate that increasing f to 40 restored stability. The pole trajectories are further examined in Figure 6-4, where it is shown that one of the unstable poles became stable at $f = 40$ (Figure 6-4a), while the other already stabilized at $f = 35$ (Figure 6-4b).

For $p = 40$ and $Q = 1 \times 10^6$, stability was also restored at $f = 30$. However, for $p = 25$, the controller remained unstable for all investigated choices of the future window size f. These results suggest that increasing the future horizon f can sometimes compensate for instabilities introduced by more aggressive controllers. Nonetheless, considering real-time feasibility, significantly increasing f is not always practical. Figures containing the results for $p = 25$ and $p = 40$ are included in Appendix C-3.

Based on these findings, internally stable controllers can be identified. In the next section, the focus shifts from the internal stability of the controller to analyzing the stability of the complete CL system. This analysis is based on configurations with $p = 20$, which most consistently yielded stable controller performance while maintaining real-time feasibility. This configuration, with an exponential forgetting factor of $\lambda_{\text{exp}} = 0.99995$ after 100 seconds of PRBS excitation, will be referred to as the baseline model. Although configurations with $p = 40$ also produced stable controllers, they were excluded due to their higher computational burden and potential problems with real-time feasibility.

6-1-2 Closed-Loop Stability

To analyze the CL stability of the system, the evaluation focused on the Gain Margin (GM), Phase Margin (PM), the maximum of the sensitivity function (M_S) (or modulus margin $\frac{1}{M_S}$), and the Nyquist criterion applied to the loop transfer function $L(z)$, as defined in Equation 6-5. This loop transfer function incorporated one of the internally stable controllers derived previously, denoted as $C(z)$, and formulated in Equation 6-2. The controller's negative was considered to ensure a negative feedback system.

The baseline model was used as the plant model, $G(z)$, which corresponded to the identification result for $p = 20$ and $\lambda_{\text{exp}} = 0.99995$ after 100 seconds of PRBS excitation, as discussed at the end of the previous section. The structure of $G(z)$ followed the same formulation as presented in Equation 6-2 and was found to have no unstable poles.

$$L(z) = C(z)G(z) \tag{6-5}$$

The GM and PM of the system were assessed to ensure compliance with the rule of thumb presented in Equation 6-6 [92]. The GM indicates how much the system's open-loop gain can increase before the CL system becomes unstable. While the PM represents how much additional phase lag can be introduced before instability occurs. Together, they provide insight into the robustness to stability of the CL system [93].

$$PM > 30^\circ, GM > 2 \tag{6-6}$$

Because gain and phase perturbations can occur simultaneously [94], the infinity norm of the sensitivity function M_S was also examined, as shown in Equation 6-7. This norm provides a

robust measure of the worst-case amplification of disturbances or model uncertainties within the CL system. The modulus margin ($\frac{1}{M_S}$) quantifies the shortest distance from the Nyquist curve of $L(z)$ to the critical -1 point [95]. Specifically, the sensitivity function $S(z)$ is defined as:

$$S(z) = \frac{1}{1 + L(z)}$$

The sensitivity function characterizes how output disturbances and reference tracking errors propagate through the system. A smaller magnitude of $S(z)$ generally indicates better disturbance rejection and tracking performance [96].

The infinity norm of S, denoted M_S, is the maximum magnitude of the sensitivity function over all frequencies:

$$M_S = \max_{\omega} |S(e^{j\omega})| = \|S\|_\infty \tag{6-7}$$

Using the value of M_S, conservative lower bounds for the GM and PM can be computed, as shown in Equation 6-8 [92]. This offers a more complete overview of the CL robustness:

$$GM \geq \frac{M_S}{M_S - 1}, \quad PM \geq 2 \arcsin\left(\frac{1}{2M_S}\right) \tag{6-8}$$

Given the relationships in Equation 6-8 and the Rules of thumb in Equation 6-6 it follows that the modulus margin $\frac{1}{M_S}$ must approximately be greater than 0.5 to achieve the wanted margins. This gives a practical benchmark for evaluating whether a controller design satisfies typical robustness criteria.

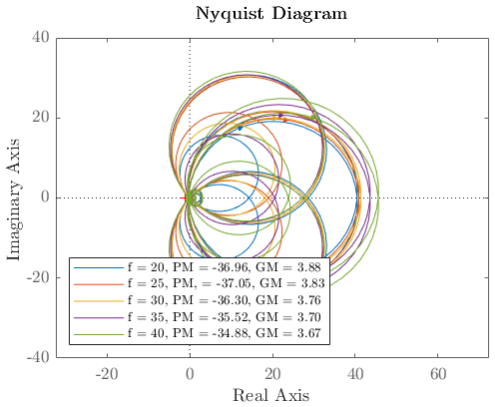
A more formal and comprehensive method for verifying CL stability is to apply the Nyquist Criterion, as shown in Equation 6-9 [97]. Unlike gain, phase, and modulus margins, which offer conservative robustness indicators, the Nyquist criterion directly evaluates the CL pole locations by analyzing the loop transfer function $L(z)$.

$$Z = N + P \tag{6-9}$$

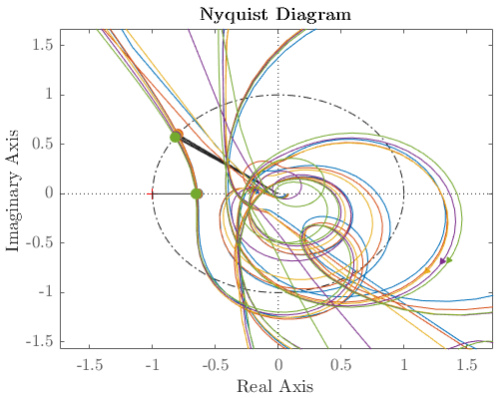
Here, Z denotes the number of unstable CL poles, N is the number of clockwise encirclements of the critical point -1 in the complex plane, and P is the number of unstable poles in the open-loop system. For a system to be stable, the Nyquist plot must satisfy this relation such that $Z = 0$ [97].

Varying the future window size f

In Figure 6-5, the influence of varying the future window size f is illustrated. A slight decrease in the GM was observed as f increases. However, the GM remained above the threshold defined by the rule of thumb in Equation 6-6. Despite a negative PM, the CL system remained stable for all three values of f.



(a) Nyquist plot for different values of f, with $p = 20$ and $Q = 1 \times 10^5$. The legend displays the Gain and Phase margins.

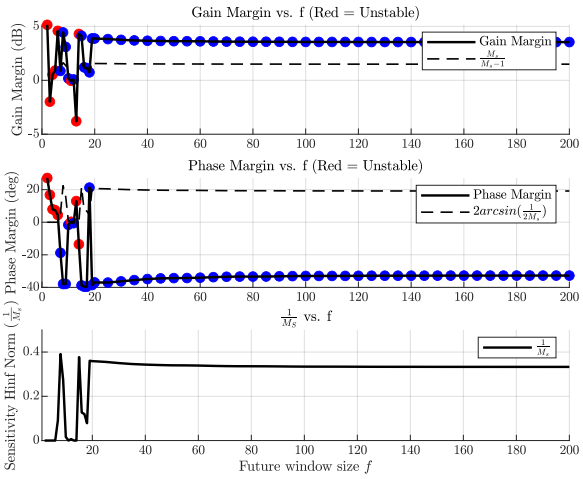


(b) Zoomed-in Nyquist plot of positive frequencies for the same data as in (a), highlighting the region near the critical point.

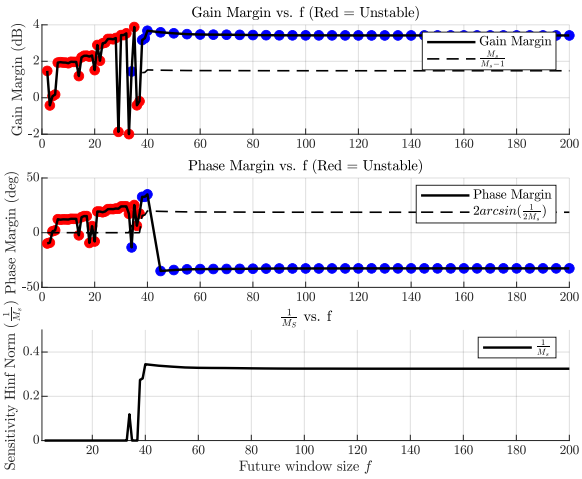
Figure 6-5: Nyquist plots showing the stability characteristics for different future window values f, with $p = 20$ and $Q = 1 \times 10^5$. Subfigure (b) provides a zoomed-in view of the same results shown in (a), focusing on the region near the -1 point while showing only the positive frequencies

This can be explained using the Nyquist criterion described in Equation 6-9. Since the Nyquist plot of $L(z)$ did not encircle the critical point -1 ($N = 0$), and the open-loop system $L(z)$ had no unstable poles ($P = 0$), the criterion yielded $Z = N + P = 0$. Thus, the CL system had no unstable poles and was, therefore, stable.

The reason a negative PM is reported, despite the absence of encirclements and unstable poles of $L(z)$, is further clarified in Appendix B-5, which provides an illustrative example of how this can occur due to how the PM is conventionally computed.



(a) Stability margins versus f for $Q = 1 \times 10^5$.



(b) Stability margins versus f for $Q = 1 \times 10^6$.

Figure 6-6: Gain and Phase margins, as well as $\frac{1}{M_S}$, plotted as functions of the future window size f, for two different values of Q. Red markers indicate unstable closed-loop systems; stable systems are marked in blue. It can be seen that the value $\frac{1}{M_S}$ often reflects the PM more accurately than the phase margin itself, especially in cases where the PM is incorrectly reported as negative. Additionally, it can be observed that the system is unstable when $\frac{1}{M_S} = 0$.

To further analyze the PM, GM, and $\frac{1}{M_S}$ when varying the prediction horizon f, Figure 6-6a and Figure 6-6b illustrate the evolution of the stability margins for $f = 2$ to $f = 200$, with $Q = 1 \times 10^5$ and $Q = 1 \times 10^6$, respectively. In these plots, unstable CL systems are indicated with red dots, while stable systems are marked with blue dots.

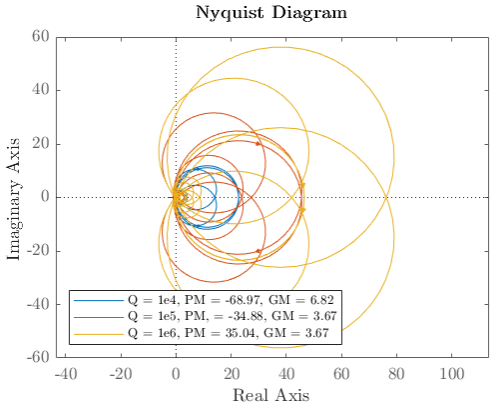
The results suggest that, despite negative phase margins, a stability threshold existed. Beyond this threshold, all controllers remained stable—approximately at $f = 20$ for $Q = 1 \times 10^5$ and at $f = 40$ for $Q = 1 \times 10^6$. Beyond this point, the value of $\frac{1}{M_S}$ stabilized around 0.3, which, although indicative of stability, did not satisfy the robustness guidelines defined by the rules of thumb in Equation 6-6.

Varying the output penalty Q

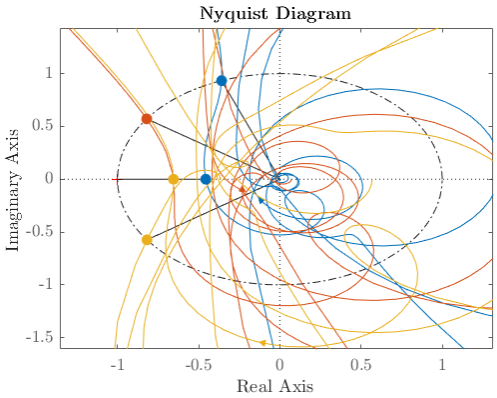
Figure 6-7 presents a Nyquist diagram for three different values of Q, revealing some fluctuations in the stability margins. The GM remained above the threshold of 2 for all tested values of Q. However, negative PM values were observed for $Q = 1 \times 10^5$ and $Q = 1 \times 10^6$.

Despite the negative phase margins, the Nyquist criterion confirms that the CL system remained stable. This is attributed to the absence of any encirclement around the critical point at -1, which, when combined with the fact that all the considered loop transfer functions have no unstable poles, ensures that the number of unstable CL poles remains zero.

To determine an appropriate value of Q that ensures CL stability, Figure 6-8a and Figure 6-8b depict the stability margins as functions of Q for future window sizes $f = 25$ and $f = 40$, respectively. In Figure 6-8a, when Q exceeds 2.8×10^5, the GM significantly declines and $\frac{1}{M_S}$ begins to decrease rapidly, leading to instability only when the $\frac{1}{M_S}$ reaches approximately zero at $Q = 3.7 \times 10^5$. A similar trend is observed in Figure 6-8b, where for $f = 40$, the GM decreases and $\frac{1}{M_S}$ decreases rapidly at a much higher threshold, $Q = 1.2 \times 10^6$.



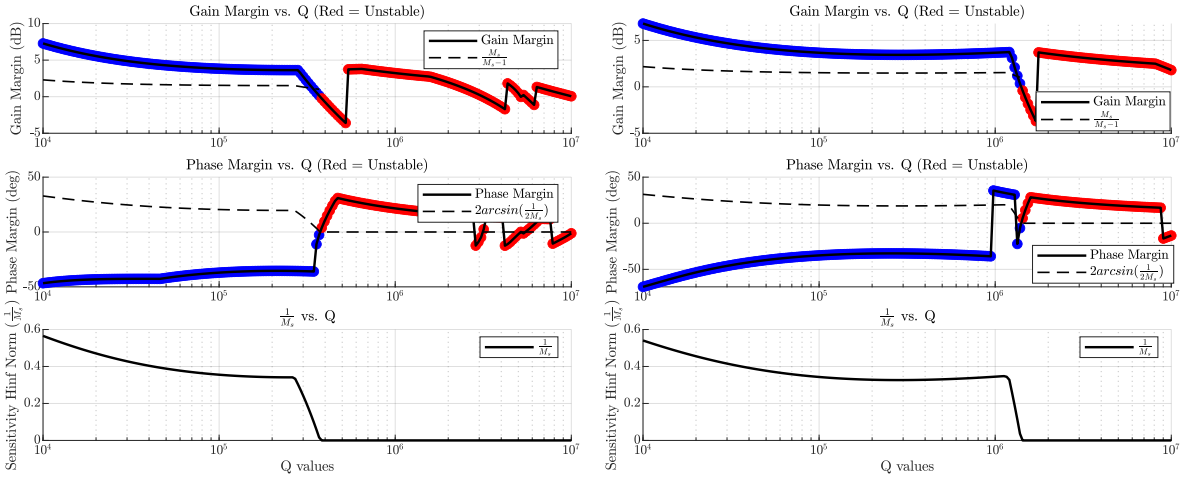
(a) Nyquist plot for different values of Q, with $p = 20$ and $f = 40$. The legend includes Gain and Phase margins.



(b) Zoomed-in Nyquist plot of positive frequencies for the same data as in (a), focusing on the region near the critical point.

Figure 6-7: Nyquist plots illustrating the stability characteristics for different values of Q, with $p = 20$ and $f = 40$. Subfigure (b) provides a zoomed-in view of the same results shown in (a), emphasizing the behavior near the -1 point while showing only the positive frequencies.

Interestingly, a slight increase in the GM was observed in both cases just before the sharp decline. Furthermore, it was observed that to satisfy the robustness guidelines defined by the rules of thumb—specifically that $\frac{1}{M_S} > 0.5$—a value of Q around 10^4 was required.



(a) Stability margins versus Q for $f = 25$.

(b) Stability margins versus Q for $f = 40$.

Figure 6-8: Gain and Phase margins, as well as $\frac{1}{M_S}$, plotted as functions of Q for $f = 25$ and $f = 40$. Red markers indicate unstable closed-loop systems; stable systems are marked in blue. The value $\frac{1}{M_S}$ often reflects the PM more accurately than the phase margin itself, especially in cases where the PM is incorrectly reported as negative. Additionally, it can be observed that the system becomes unstable when $\frac{1}{M_S} = 0$.

The stability analysis provides a comprehensive overview of the system's gain and phase margins across various controller configurations, offering valuable insights into ensuring CL robustness. However, while these theoretical results are essential, practical validation is critical to account for discrepancies between the identified model and the actual system behavior. Therefore, the next sections present real-life time-domain and frequency-domain experiments conducted on the piezo-actuated beam setup.

6-2 Time Domain Experiments

This section outlines the methodology and experiments in the time domain. It includes a description of the algorithms and controllers applied, the experimental setup, the characteristics of the input signals used during testing, and the performance measures used. Together with the results of the described tests.

6-2-1 Time Domain Experiment Methodology

First, the system's performance is analyzed in the time domain. A simplified block diagram of the experimental setup is shown in Figure 6-9. The experiment began with a 40-second PRBS signal applied to the control input channel u (u_1 in Figure 5-6) to initialize the controller. This was followed by a 10-second settling period, during which the controller was activated and

the system was allowed to stabilize. At the 50-second mark, a block wave with an amplitude of 400 was applied to the second input channel, denoted as d (u_2 in Figure 5-6). At the same time, the controller continued to operate on the input u. The system response was measured at the first output, denoted as y (y_1 in Figure 5-6). The objective was to suppress the vibrations induced by the block disturbance as quickly as possible.

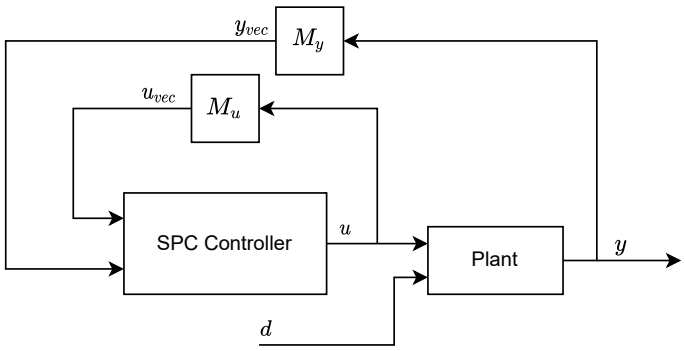


Figure 6-9: Block scheme of the experimental setup, where u is u_1, d is u_2, and y is y_1 from Figure 5-6.

An initial test was conducted in which a block wave was applied to the second input channel without any active control, serving as a baseline for performance evaluation.

Subsequently, a series of controlled experiments were performed to evaluate the effect of the prediction horizon f. In these tests, f was varied in increments of 5 while keeping $p = 20$, $\lambda_{\text{exp}} = 0.99995$, and $Q = 10^5$ constant. The lower bound of $f = 15$ was selected due to poor performance observed at $f = 10$, confirming the instability found in subsection 6-1-2. In contrast, the upper limit was set to $f = 40$ due to real-time feasibility constraints.

In addition, another experiment was carried out with $p = 20$, $\lambda_{\text{exp}} = 0.99995$, and $f = 40$, motivated by the stability observed in subsection 6-1-2 for higher values of Q. In this experiment, Q was varied across 10^4, 10^5, and 10^6 to gain insights into the effect of output penalty weighting on control performance.

The experiments were compared based on two metrics: the cumulative stage cost, which reflects how effectively the cost function is minimized, and the cumulative absolute error, which quantifies the output performance and provides insight into the controller's vibration suppression capabilities.

The experiments were initially conducted using the Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC) algorithm. However, the input and output constraints were rarely activated during testing. This is primarily because, when the output exceeds a value of 10, it is clipped, resulting in distorted output signals under aggressive inputs. Such distortion negatively affects the accuracy of the System Identification (SysID) process, as the measured outputs no longer represent the actual system behavior. In practice, this imposed an implicit input limitation that is more restrictive than system constraints. Additionally, explicitly enforcing output constraints at 10 did not improve performance, as the control objective already minimized the output and implicitly tried to avoid reaching this limit. When the output reached 10, the clipping again compromised identification, resulting in unwanted behavior.

Due to these factors, the R-CL SPC algorithm demonstrated similar performance. Furthermore, because of real-time feasibility issues observed for $f > 30$ in the CR-CL SPC algorithm, it was decided to present only the results of the R-CL SPC, while still reporting the computational times of both algorithms for completeness.

6-2-2 Time Domain Experiment Results

The first time-domain result is the baseline test, presented in Figure 6-10, where the block disturbance was applied without active control. This served as a reference for evaluating the performance of the controllers and shows that when no control was applied, the system was unstable.

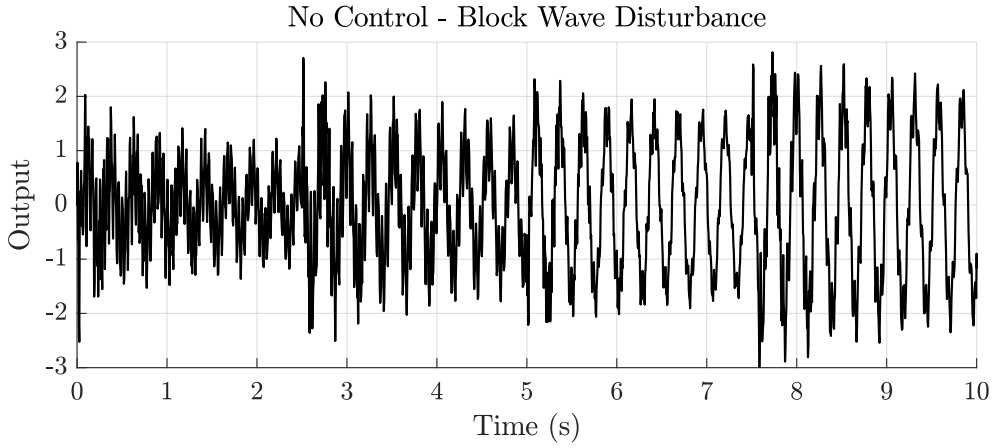


Figure 6-10: Time-domain response of the system without control, serving as a benchmark and showing that the system is unstable with no control.

Subsequently, the effect of varying the prediction horizon f was tested while keeping $p = 20$, $\lambda_{\text{exp}} = 0.99995$, and $Q = 10^5$ constant. The time-domain responses for several values of f are shown in Figure 6-11.

As shown in Figure 6-11, all controllers were able to stabilize the system in response to the block wave disturbance, with overall damping behavior remaining consistent across different values of f. To evaluate performance more quantitatively, Figure 6-12 presents the cumulative stage cost, which reflects how well the cost function was minimized, showing overall controller performance. Additionally, Figure 6-13 shows the cumulative output error, capturing the quality of output tracking and implicitly how well the introduced vibrations were damped.

The cumulative stage cost analysis in Figure 6-12 shows relatively small differences across the values of f since the input costs are more significant than the output costs. However, when focusing only on the output performance, as shown in Figure 6-13, it is clear that $f = 25$ yielded the lowest cumulative error, indicating the best performance in terms of vibration suppression.

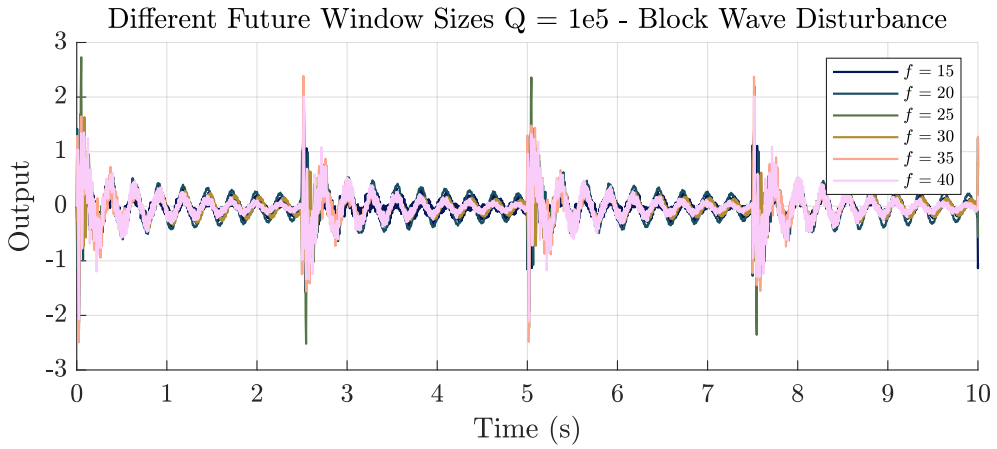


Figure 6-11: Time-domain response for different values of prediction horizon f, with $Q = 10^5$, $R = 1$, and $p = 20$.

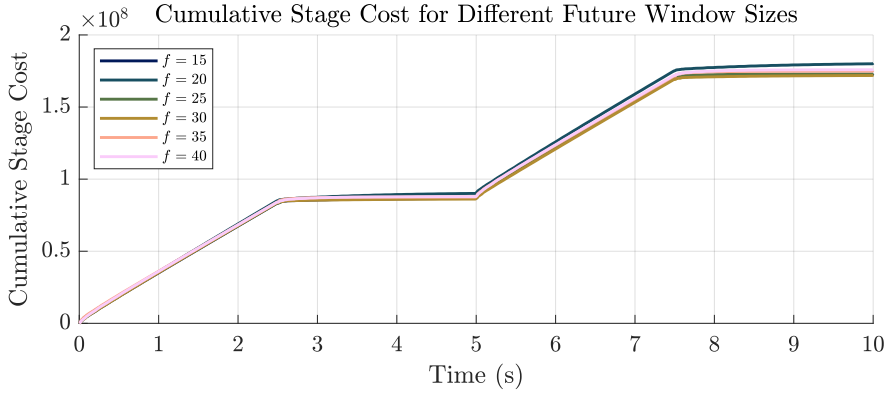


Figure 6-12: Cumulative stage cost for different values of f, with $Q = 10^5$, $R = 1$, and $p = 20$.

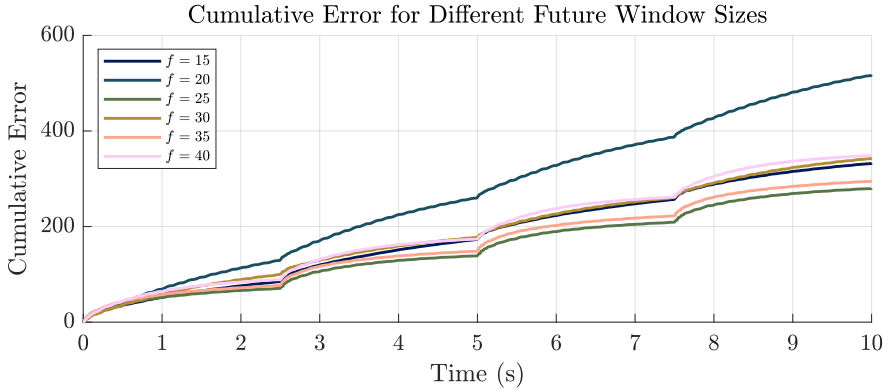


Figure 6-13: Cumulative output error for different values of f, with $Q = 10^5$, $R = 1$, and $p = 20$.

To complement the performance evaluation, the average and maximum computational times for both the R-CL SPC and CR-CL SPC algorithms were recorded for each value of f. These results are summarized in Table 6-1. While the mean computational times for the CR-CL SPC algorithm were only slightly higher than those of the R-CL SPC, results show that the maximum computational times represent the bottleneck for real-time implementation.

Table 6-1: Mean and maximum computational times (in seconds) for different values of f. Looking at the R-CL SPC algorithm and the CR-CL SPC algorithms. When $p = 20$ the CR-CL SPC algorithm becomes unfeasible for $f > 30$

f	R-CL SPC		CR-CL SPC	
	Mean	Max	Mean	Max
15	0.0006	0.0006	0.0006	0.0007
20	0.0011	0.0011	0.0012	0.0013
25	0.0016	0.0016	0.0017	0.0020
30	0.0025	0.0025	0.0025	0.0034
35	0.0036	0.0036	Not Feasible	
40	0.0045	0.0045	Not Feasible	

To evaluate the influence of the output penalty weighting Q, a second series of tests was conducted with a fixed prediction horizon of $f = 40$, while varying Q across 10^4, 10^5, and 10^6. The results are presented in Figure 6-14. Computational times are not explicitly reported here, as the value of Q does not affect execution time, resulting in consistent computational times of approximately 0.45 ms across all tests.

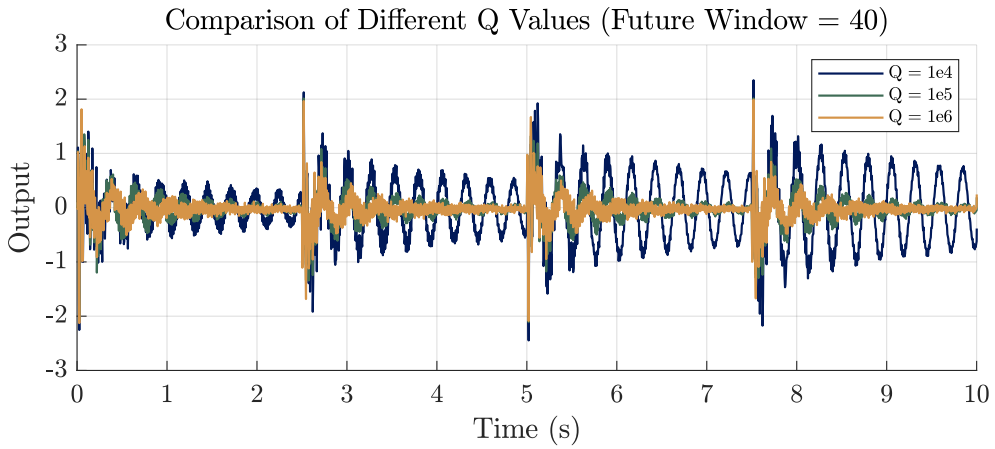


Figure 6-14: Time-domain response for three values of Q, with $f = 40$, $R = 1$, and $p = 20$.

While a visual inspection of the responses suggests improved performance when increasing Q, further analysis using the cumulative stage cost and output error offers more insight, as shown in Figure 6-15 and Figure 6-16, respectively.

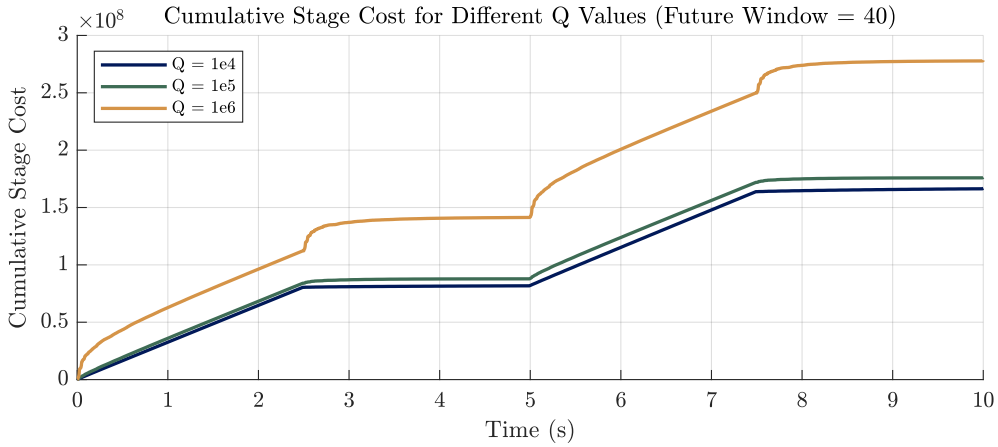


Figure 6-15: Cumulative stage cost for three values of Q, with $f = 40$, $R = 1$, and $p = 20$.

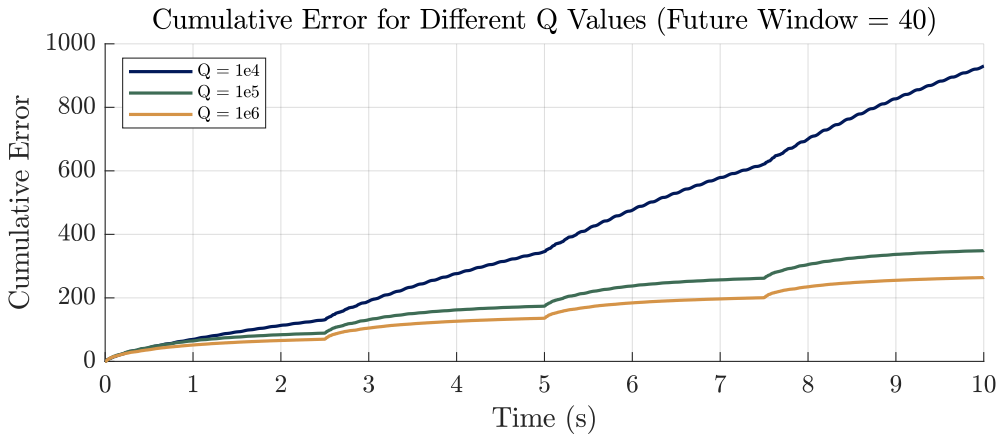


Figure 6-16: Cumulative output error for three values of Q, with $f = 40$, $R = 1$, and $p = 20$.

As shown in Figure 6-15, lower values of Q (e.g., $Q = 10^4$) resulted in a smaller cumulative stage cost, but this did not directly translate into better output performance. In contrast, the output error resulted in Figure 6-16 show that $Q = 10^6$ yielded the best output performance and, therefore, was more effectively damping the disturbance.

While the time-domain tests provided valuable insights into control effectiveness and computational feasibility, they did not fully capture the system's behavior in the frequency domain, especially with respect to the suppression of the first two vibration modes. Therefore, the following section continues the evaluation using frequency-domain analysis.

6-3 Frequency Domain Experiment

This section presents and analyzes the methodology and results of the frequency-domain experiments. The methodology describes the applied controllers, the experimental setup,

and the characteristics of the input signals during testing, and the results analyze how much the natural vibration peaks are attenuated. Due to the similarity of the results in the time domain, the computational times are not presented in this section.

6-3-1 Frequency Domain Experiment Methodology

This part of the section analyzes the system's performance in the frequency domain, beginning with the used methodology. The experiment began with a 40-second white noise signal with variance $\sigma^2 = 2 \times 10^3$ applied to the control input channel u to initialize the controller. This was followed by a 10-second settling period, during which the controller was activated, and the system was allowed to stabilize.

At the 50-second mark, a white noise signal with the same variance was applied to the disturbance input channel d for 100 seconds, while the controller continued to operate on the input u. The system response was measured at the first output channel y. The objective was to suppress the first two vibration modes excited by the disturbance.

A baseline test without active control was first conducted to establish a reference for comparison. In this case, the white noise disturbance was applied to d while keeping the control input fixed at $u = 0$, allowing the open-loop frequency response from d to y to be obtained.

Subsequently, a series of controlled experiments were carried out to evaluate the effect of the prediction horizon f. In these tests, f was varied in increments of 5, while keeping $p = 20$, $\lambda_{\text{exp}} = 0.99995$, $R = 1$, and $Q = 10^5$ constant. The lower and upper bounds for f were chosen to match those used in the time-domain experiments.

In addition, a separate set of tests was performed using the best-performing controller from the f-variation tests and the only stable configuration with a higher weighting $Q = 10^6$. During the frequency-domain analysis, the controller with $f = 40$, $R = 1$ and $Q = 10^6$ was unstable under the given conditions. To restore stability, the forgetting factor was increased to $\lambda_{\text{exp}} = 1$. This adjustment was also applied to the best-performing controller from the f-variation tests to ensure a consistent basis for comparison.

The controllers' performance was quantified by evaluating the attenuation of the first two vibration modes, measured by the peak magnitude reduction near their respective frequencies, and by calculating the cumulative absolute error, which reflects the overall output performance of each controller configuration.

6-3-2 Frequency Domain Experiment Results

After the baseline experiment was conducted, the first frequency-domain test experiment, which is the test for varying values of f, is performed. The results of this experiment are presented in Figure 6-17. The figure shows that all controllers effectively attenuated the second natural frequency, achieving approximately 30 dB of damping. The first natural frequency was also significantly suppressed, with attenuation levels of around 8 dB for $f = 20$, 11 dB for $f = 30$, 13 dB for $f = 40$, and 15 dB for both $f = 25$ and $f = 35$. The corresponding computational times are not explicitly reported here due to their strong similarity to the results shown in Table 6-1.

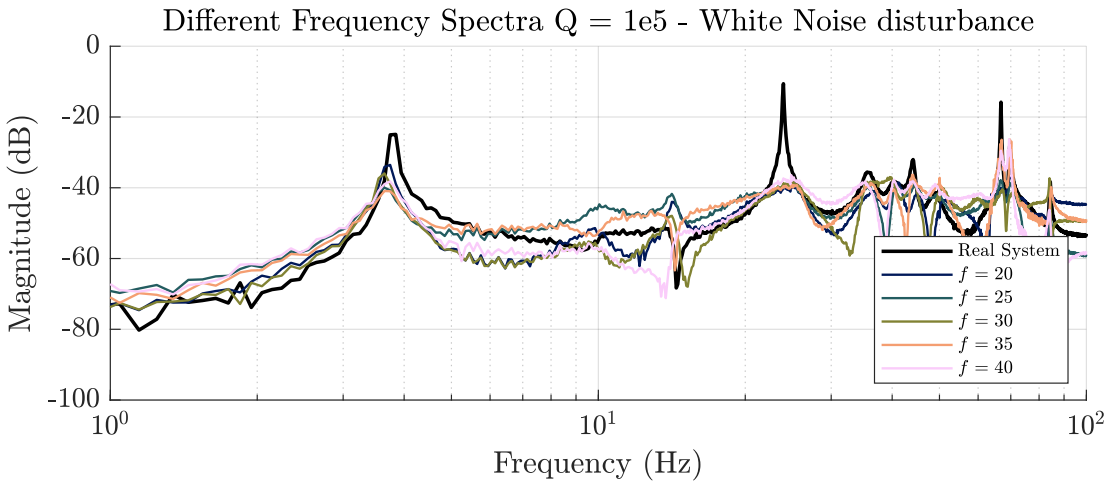


Figure 6-17: Frequency domain response from d to y for different values of prediction horizon f. $Q = 10^5$, $R = 1$, and $p = 20$.

The cumulative output error analysis in Figure 6-18 shows that the controller with $f = 35$ initially performed comparably to the others but deteriorated more rapidly after the first 5 seconds, while the controller with $f = 40$ exhibited a steeper overall increase in cumulative error. This behavior may point to potential SysID issues caused by insufficient Persistance of Excitation (PE) and limited robustness in stability. These issues were likely caused by the unknown white noise signal introduced through the second input d, which is not explicitly accounted for by the algorithm. As a result, more historical data may be required to compensate for the disturbance. Due to the limited robustness in CL stability, as shown in subsection 6-1-2, it is likely that a change of the model estimate can destabilize the CL system.

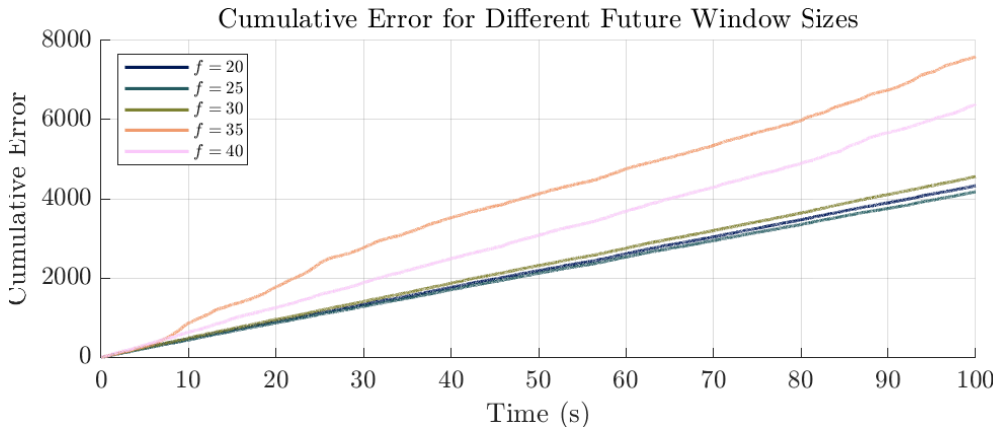


Figure 6-18: Cumulative output error in the frequency domain for varying f. $Q = 10^5$, $R = 1$, and $p = 20$.

The controllers with $f = 25$ and $f = 30$ demonstrated the best overall performance, with a slight advantage for $f = 30$ in terms of cumulative output error. These results are consistent

with the time-domain analysis, where the configuration with $f = 25$ was previously identified as the best-performing setup.

To further investigate performance under more aggressive tracking, the only stable configuration with a higher weighting on output error, $Q = 10^6$, was revisited using $p = 20$, $f = 40$. However, this configuration exhibited similar instability to what was observed in the earlier $f = 35$ and 40 tests.

To address the observed stability issues, which are potentially caused by degradation in system identification, a higher exponential forgetting factor of $\lambda_{\text{exp}} = 1$ was applied. The same adjustment was also made to the best-performing controller from the varying f tests ($f = 25$) to ensure a fair comparison. The resulting frequency responses of both modified controllers are presented in Figure 6-19.

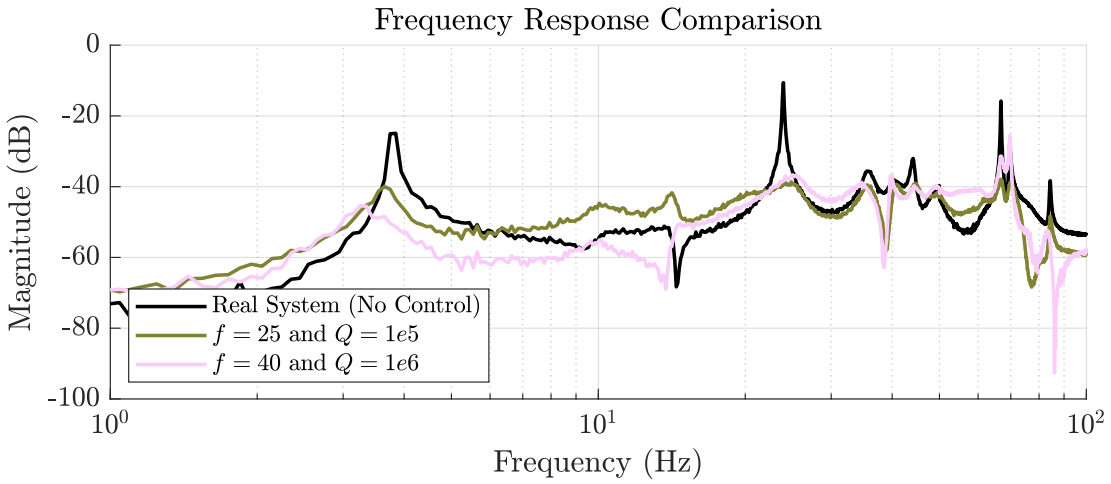


Figure 6-19: Frequency response of the two best-performing controllers compared to the ground truth. $R = 1$, and $p = 20$.

The controller with $Q = 10^6$ and $f = 40$ demonstrated improved attenuation at the first natural frequency, achieving a reduction of 19 dB, an improvement of 4 dB compared to the controller with $Q = 10^5$ and $f = 25$. However, performance slightly decreased at the second natural frequency, with attenuation reduced to 31 dB, representing a 1 dB drop.

As shown in Figure 6-20, the cumulative output error indicates that the higher Q configuration performs worse in terms of overall output suppression despite its localized frequency-domain advantage.

While both controllers show strengths, the configuration with $Q = 10^5$, $f = 25$ offered the best overall trade-off between performance and robustness. Although the controller with $Q = 10^6$, $f = 40$ provided slightly better attenuation at the first resonance peak, it performed worse in terms of cumulative output error, and it exhibited reduced damping at the second resonance. Additionally, it suffered from stability issues unless a high value of λ_{exp} was used.

In contrast, the controller with $f = 25$ delivered consistent performance across both vibration modes, maintained stability more reliably, and is therefore considered feasible for implementation with the CR-CL SPC algorithm. This makes it the most practical and effective choice overall.

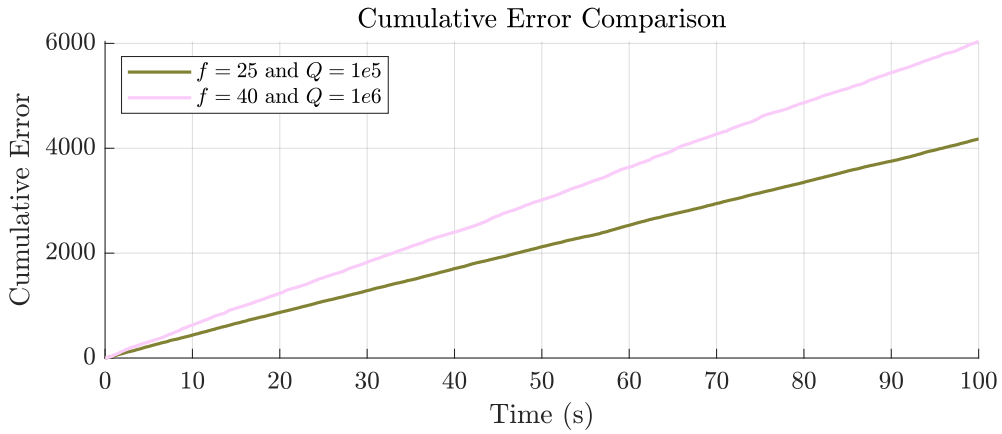


Figure 6-20: Cumulative error of frequency response tests for the two best-performing controllers. $R = 1$, and $p = 20$.

Chapter 7

Conclusion

This final chapter reflects on the research presented throughout the thesis. It synthesizes the insights gained from theoretical development, simulation results, and real-world implementation. The goal of this work was to develop a subspace predictive vibration control strategy that not only meets real-time performance requirements but is also capable of handling constraints and adapting to changes in system behavior. By grounding the work in practical challenges encountered in the vibration control of a piezo-actuated beam, this thesis aimed to bridge the gap between data-driven control theory and real-world applicability. The main research question that drove the research:

> *How can an adaptive direct data-driven predictive controller be designed to enable constrained, real-time vibration control of a real-world piezo-actuated beam setup?*

To answer this overarching question, the research was structured around four guiding sub-questions: how to incorporate adaptivity into the algorithm, how to reduce computational time for real-time feasibility, how different controllers perform in comparison, and how vibration control can be practically implemented on a real-world system.

This chapter summarizes the findings in relation to these sub-questions and reflects on the contributions, limitations, and directions for future research.

7-1 Summary of Findings

The thesis findings are structured around the four sub-questions that framed the research. Each subsection below directly addresses one of these sub-questions, outlining the key insights and results obtained.

7-1-1 Sub-question 1: How can the adaptivity of the algorithm be incorporated?

This sub-question was addressed by implementing adaptive capabilities within the Subspace Predictive Control (SPC) framework through adaptive Hankel matrices or a recursive System Identification (SysID) update. These enhancements allowed the controller to adjust to changing system dynamics in simulation environments. Although hardware limitations prevented full validation on nonlinear or time-varying real-world systems, preliminary simulation results indicated the potential of these adaptive techniques.

7-1-2 Sub-question 2: How can computational time be reduced to achieve real-time feasibility?

To answer this sub-question, the thesis focused on overcoming the computational burden of repeated pseudo-inverse calculations in standard SPC algorithm (3.33 ms). The solution involved reducing the optimization problem for Closed Loop Subspace Predictive Control (CL SPC) (1.40 ms) and employing recursive SysID updates in the proposed Recursive Closed Loop Subspace Predictive Control (R-CL SPC) and Constrained Recursive Closed Loop Subspace Predictive Control (CR-CL SPC) algorithms. As a result, execution times were drastically reduced to 0.11 ms and 0.51 ms per sample, respectively, maintaining better computational performance even when incorporating input constraints via a Quadratic Programming (QP) solver.

7-1-3 Sub-question 3: How do the proposed controllers perform compared to each other?

To address this question, the thesis compared SPC, CL SPC, R-CL SPC, and CR-CL SPC controllers using the cumulative stage cost and computational times. The analysis revealed a trade-off: while SPC and CL SPC offered better control performance, the proposed R-CL SPC and CR-CL SPC significantly improved execution speed, a crucial requirement for real-time applications. Furthermore, CR-CL SPC demonstrated the capability to handle constraints, although in the tested cases, constraints remained inactive, and performance resembled that of R-CL SPC.

7-1-4 Sub-question 4: How can vibration control be effectively implemented?

This final sub-question was addressed through experimental implementation on a real-world piezo-actuated beam setup. The R-CL SPC and CR-CL SPC controllers were deployed to dampen the first two resonance modes while operating at a sampling frequency of 200 Hz. Despite some feasibility issues at higher future window sizes ($p = 20$ and $f > 30$) with CR-CL SPC, both controllers effectively suppressed vibrations. Time-domain tests showed suppression of a step input within 2.5 seconds, while frequency-domain analysis revealed reductions of 15 dB and 30 dB at the first and second natural frequencies, respectively.

7-2 Key Contributions

The key contributions of this work include:

- Introduction of adaptive capabilities into the SPC algorithms using adaptive Hankel matrices or recursive SysID updates.
- The implementation of computationally efficient R-CL SPC and CR-CL SPC algorithms suitable for real-time control.
- A comprehensive performance comparison of different SPC algorithms, highlighting trade-offs between accuracy and speed.
- Successful real-time implementation of adaptive controllers on a physical piezo-actuated beam setup, validating their practical feasibility for vibration suppression.

7-3 Limitations and Future Work

While the results presented in this thesis are promising in terms of computational efficiency and real-time feasibility, several areas warrant further investigation—either due to limitations of this work or because they represent valuable directions for future research:

- **SISO-Only Evaluation:** All algorithms were developed and tested for Single-Input Single-Output (SISO) systems. Extending these methods to Multi-input Multi-output (MIMO) settings could be essential for some practical applications and presents additional challenges regarding feasibility and performance.
- **Simplified Simulation Model:** All simulations were conducted on a second-order linear system to isolate and evaluate the algorithmic behavior. While this approach offered clear insights into computational efficiency and control performance, the results may not fully generalize to more complex systems, such as those with higher-order dynamics, time delays, or coupled interactions.
- **Impact of Model Estimation:** Since the algorithms depend on real-time SysID, their sensitivity to estimation errors or model mismatch deserves further investigation. Studying how uncertainty in estimated Markov parameters affects stability and control performance would be an important step toward robust deployment.
- **Robustness to Noise and Disturbances:** Although one simulation was performed with a higher noise variance to evaluate the influence of noise on controller performance, a more systematic robustness analysis was not conducted. Future work could investigate how different levels of measurement noise or external disturbances affect performance and stability.
- **Input Constraints Rarely Active in Practice:** Although the CR-CL SPC algorithm was designed to handle constraints, they were almost unactivated during the simulations and real-time experiments. As a result, the added value of constraint enforcement could not be fully assessed. Further simulations or experiments with tighter

constraint margins would be beneficial in investigating the added value of constraint enforcement.

- **Handling Time-Varying and Nonlinear Systems:** While initial simulations have provided insight into time-varying system behavior, further research, extensive simulation studies, and experimental validation are necessary. Establishing the algorithm's effectiveness on nonlinear or time-varying systems is essential for confirming its practical relevance and broader applicability.
- **Validation on Different Real-World Systems:** Applying the proposed algorithms to other real-world systems would help validate their performance and robustness. A slow system may favor the use of standard SPC or CL SPC, while systems with strict constraints may highlight the strengths of the CR-CL SPC, furthermore a nonlinear or time varying system could show the added value of the adaptive nature of all algorithms.
- **Improved QP Solvers:** The QP solver in CR-CL SPC proved to be computationally demanding, particularly as the prediction horizon increases. Exploring alternative solvers to ForcesPro or implementing warm-start strategies could enhance real-time feasibility in constrained applications.
- **Comparison with Other Control Techniques:** While this thesis focused on comparisons between multiple SPC variants, future work could benchmark the proposed algorithms against other control strategies, such as Data-Enabled Predictive Control (DeePC) or Model Predictive Control (MPC), to position their advantages and limitations better.
- **Evaluation of Directional Forgetting:** The directional forgetting scheme described by van der Veen [50] may enhance Persistance of Excitation (PE) conditions under adaptive conditions. However, this thesis did not explore its integration and therefore further investigation is required to assess stability, convergence, and practical benefits of the directional forgetting scheme.

7-4 Final Remarks

Although this thesis has limitations and highlights several directions for future work, it demonstrates that SPC can be effectively adapted and implemented for constrained, real-time vibration control of a piezo-actuated beam setup. Adaptivity was achieved using recursive system identification and adaptive Hankel matrices, enabling the controller to respond to time-varying or nonlinear dynamics. Constraints on input and output were handled within the CR-CL SPC formulation, ensuring safe and feasible operation. Incorporating recursive estimation techniques significantly reduced computational demands, allowing both R-CL SPC and CR-CL SPC to operate at real-time sampling rates on practical hardware. Experimental results confirmed the effective suppression of the beam's first two resonance modes, thereby validating the approach. This work contributes to developing adaptive and direct Data-Driven Predictive Control (DDPC) strategies suitable for real-world applications.

Appendix A

FLOPs

This appendix provides a detailed analysis of the computational complexity of key components within the proposed control algorithms. In particular, it quantifies the number of Floating Point Operations (FLOPs) required to perform common matrix operations, solve least squares problems, and recursively update estimates and system representations.

A lookup table summarizing the FLOPs costs of standard matrix and vector operations is first presented to aid this analysis. This table serves as a reference for subsequent calculations.

The remainder of the appendix is dedicated to the explicit derivation of FLOPs counts for:

- Solving least-squares problems in both constrained and unconstrained settings, with and without feedthrough terms.
- Constructing matrices such as Ψ and Λ, which are central to the predictive control formulations.
- Recursively updating system parameters using techniques such as Givens rotations.

These complexity assessments are important for evaluating the proposed algorithms' real-time feasibility. By explicitly quantifying the cost of each operation, this appendix supports the claims made in the main body of the thesis regarding algorithmic efficiency.

A-1 Lookup Table

$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{C} \in \mathbb{R}^{n\times l}$ are arbitrary matrices. $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix, $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is lower triangular, $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{R}^{n}$, $\boldsymbol{c} \in \mathbb{R}^{m}$, and $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is positive definite.

Expression	Description	Products	Summations	FLOPs
Basic Vector and Matrix Operations				
$\alpha\mathbf{a}$	Vector Scaling	n	–	n
$\alpha\mathbf{A}$	Matrix Scaling	mn	–	mn
$\mathbf{a}^H\mathbf{b}$	Inner Product	n	$n-1$	$2n-1$
$\mathbf{a}\mathbf{c}^H$	Outer Product	mn	–	mn
Matrix Multiplications				
$\mathbf{A}\mathbf{b}$	Matrix-Vector Prod.	mn	$m(n-1)$	$2mn-m$
$\mathbf{A}\mathbf{C}$	Matrix-Matrix Prod.	mnl	$ml(n-1)$	$2mnl-ml$
$\mathbf{A}\mathbf{D}$	Diagonal Matrix Prod.	mn	–	mn
$\mathbf{L}\mathbf{D}$	Matrix-Matrix Prod.	$\frac{1}{2}n^2+\frac{1}{2}n$	0	$\frac{1}{2}n^2+\frac{1}{2}n$
$\mathbf{L}\mathbf{C}$	Matrix Product	$\frac{n^2l}{2}$	$\frac{n^2l-nl}{2}$	n^2l
Factorizations and Decompositions				
$\mathbf{L}$	Cholesky $\mathbf{R}=\mathbf{L}\mathbf{L}^H$ (Gaxpy version)	$\frac{n^3+n^2}{2}$	$\frac{n^3-n^2}{3}$	$\frac{n^3}{3}+\frac{1}{2}n^2+\frac{1}{6}n$
$\mathbf{L},\mathbf{D}$	Cholesky $\mathbf{R}=\mathbf{L}\mathbf{D}\mathbf{L}^H$	$\frac{n^3+n^2}{2}-\frac{13n}{6}+1$	–	$\frac{n^3}{3}+\frac{1}{2}n^2-\frac{7n}{3}+1$
$\mathbf{U}\mathbf{D}\mathbf{V}^T$	SVD $\mathbf{A}=\mathbf{U}\mathbf{D}\mathbf{V}^T$			mn^2+n^3 [98]
$\mathbf{B}\mathbf{L}^T$	QR decomposition $\mathbf{A}_1=\mathbf{A}_2\mathbf{L}^T$			$3mn^2-n^3$ [60]
Inverses				
$\mathbf{L}^{-1}$	Inverse of Triangular	$\frac{n^3+n^2}{2}$	–	$\frac{n^3}{3}+\frac{n^2}{2}+\frac{n}{3}$
$\mathbf{R}^{-1}$	Inverse of Pos. Definite	$n^3+\frac{3n^2}{2}$	$\frac{n^3-n^2}{2}$	n^3+2n
$\mathbf{L}^{-1}\mathbf{C}$	$\mathbf{L}^{-1}$ unknown	$\frac{n^2l}{2}$	$\frac{n^2l-nl}{2}$	n^2l

Table A-1: Table of Expressions, Products, Summations, and FLOPs, inspired by [70]

A-2 CL SPC FLOPs

A-2-1 Least Squares With $D = 0$

FLOPs calculation of:

$$\hat{\Xi}_0 = Y_{i_p,1,\bar{N}} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}^{\dagger} \tag{A-1}$$

- **Step 1: Pseudo Inverse** $\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}^{\dagger}$: Since the matrix is not square, we use the Singular Value Decomposition (SVD) to compute the pseudo-inverse. The matrix is decomposed as $A = U\Sigma V^T$ where U and V are orthogonal matrices of sizes $\bar{N} \times \bar{N}$ and $(2p) \times (2p)$, respectively, and Σ is a diagonal rectangular matrix of $\bar{N} \times (2p)$ containing $2p$ singular values. The pseudo-inverse is obtained as $A^{\dagger} = V\Sigma^{\dagger}U^T$, where $\Sigma^{\dagger}$ is computed by inverting its nonzero singular values. The computational cost of SVD is approximately $(\bar{N}^2(2p) + (2p)^3)$, while inverting the singular values adds an additional $(2p)$ FLOPs. Furthermore, a sparse matrix product of $V\Sigma^{\dagger}$ requires $(2(2p))$, and a matrix-matrix multiplication with U^T costs $(\bar{N}(2p)^2)$. This sums to $((2p)(\bar{N}^2 + \bar{N}(2p) + (2p)^2 + 3))$ FLOPs.

- **Step 2: Vector-Matrix Multiplication** $Y_{i_p,1,\bar{N}} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix}^\dagger$: The matrix multiplication of a $\bar{N}$ vector with a $\bar{N} \times 2p$ matrix typically costs $(2\bar{N}p)$ FLOPs.

This totals up to $((2p)(\bar{N}^2 + \bar{N}(2p) + (2p)^2 + 2\bar{N}p + 3))$ FLOPs.

A-2-2 Least Squares with $D \neq 0$

FLOPs calculation of:

$$\hat{\bar{\Xi}}_0 = Y_{i_p,1,\bar{N}} \left(\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \right)^\dagger \tag{A-2}$$

- **Step 1: Pseudo Inverse** $\left(\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \right)^\dagger$: Since the matrix is not square, we use SVD for the pseudo-inverse. The computational cost of SVD is $(\bar{N}^2(2p+1)+(2p+1)^3)$, with additional costs for inverting singular values $(2p+1)$, sparse matrix multiplications of $V\Sigma^\dagger$ at $(2(2p+1))$, and a matrix-matrix multiplication with U^T at $(\bar{N}(2p+1)^2)$. This results in a total cost of $((2p+1)(\bar{N}^2 + \bar{N}(2p+1) + (2p+1)^2 + 3))$ FLOPs.
- **Step 2: Vector-Matrix Multiplication** $Y_{i_p,1,\bar{N}} \left(\begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \end{bmatrix} \right)^\dagger$: The matrix multiplication of a $\bar{N}$ vector with a $\bar{N} \times (2p+1)$ matrix typically costs $(2\bar{N}(p+1))$ FLOPs.

This totals up to $((2p+1)(\bar{N}^2 + \bar{N}(2p+1) + (2p+1)^2 + 2\bar{N}(p+1) + 3))$ FLOPs.

A-2-3 Construction of Ψ

Each term in the summation:

$$\overline{C\tilde{A}^{i-\tau-1}K} \cdot \Psi_\tau \tag{A-3}$$

involves multiplying a scalar by a $2p \times 1$ vector. This requires:

$$(2p) \text{ FLOPs.} \tag{A-4}$$

Summation Over τ

Since the summation runs from $\tau = 0$ to $i-1$ with $i = f$, we have f such terms. The total cost of summation is:

$$(f \cdot 2p) = (2pf). \tag{A-5}$$

Addition with $\hat{\Xi}_i$

Since $\hat{\Xi}_i$ is also a $2p$-dimensional vector, adding two vectors of size $2p$ requires:

$$(2p) \text{ FLOPs.} \tag{A-6}$$

Total FLOP Count

For computing a single Ψ_i, the total number of FLOPs is:

$$(2pf + 2p) \tag{A-7}$$

Since all Ψ_i vectors are stacked to form a $2p \times f$ matrix, the total FLOPs for computing the entire matrix is:

$$(f \cdot (2pf + 2p)) = (2pf^2 + 2pf). \tag{A-8}$$

Conclusion

Thus, the overall computational complexity for constructing the full Ψ matrix is:

$$(2pf^2 + 2pf) \tag{A-9}$$

This indicates that the FLOP count scales quadratically with f and linearly with p.

A-2-4 Construction of Λ without feedthrough

Each term in the summation:

$$\overline{C\tilde{A}^{j-\tau-1}K} \cdot \Lambda_\tau \tag{A-10}$$

involves multiplying two scalars, which requires:

$$(1) \text{ FLOP.} \tag{A-11}$$

Summation Over τ

Since the summation runs from $\tau = 1$ to $j - 1$, we have $j - 1$ such terms. The total FLOP count is:

$$(j - 1). \tag{A-12}$$

Addition with $\overline{C\tilde{A}^{j-1}B}$

Adding two scalars requires:

$$(1) \text{ FLOP.} \tag{A-13}$$

Total FLOP Count for a Single Λ_j

$$((j-1)+1) = (j). \tag{A-14}$$

Since the entire vector Λ consists of f such values, summing over all $j = 1, \ldots, f$ gives:

$$\sum_{j=1}^{f}(j) = \left(\sum_{j=1}^{f} j\right). \tag{A-15}$$

Using the formula for the sum of the first f natural numbers:

$$\sum_{j=1}^{f} j = \frac{f(f+1)}{2}, \tag{A-16}$$

we obtain:

$$\left(\frac{f(f+1)}{2}\right). \tag{A-17}$$

Conclusion

Thus, the overall computational complexity for constructing the full Λ vector is:

$$\left(\frac{f(f+1)}{2}\right). \tag{A-18}$$

This confirms that the FLOP count scales quadratically with f.

A-2-5 Construction of Λ with feedthrough

Each term in the summation:

$$\overline{C\tilde{A}^{j-\tau-1}K} \cdot \Lambda_\tau \tag{A-19}$$

involves multiplying two scalars, which requires:

$$(1) \text{ FLOP}. \tag{A-20}$$

Additionally, the presence of the feedthrough term introduces an extra scalar multiplication:

$$\overline{C\tilde{A}^{j-1}K}D, \tag{A-21}$$

which also requires:

$$(1) \text{ FLOP}. \tag{A-22}$$

Summation Over τ

Since the summation runs from $\tau = 1$ to $j-1$, we have $j-1$ such terms. The total FLOP count is:

$$(j-1). \tag{A-23}$$

Additional Scalar Multiplications

The additional term:

$$\overline{C\tilde{A}^{j-1}\tilde{B}} \tag{A-24}$$

is another scalar that needs to be added, contributing:

$$(1) \text{ FLOP.} \tag{A-25}$$

Addition with $\overline{C\tilde{A}^{j-1}B} + \overline{C\tilde{A}^{j-1}K}D$

Since we are adding three scalars together, we have:

$$(2) \text{ FLOPs.} \tag{A-26}$$

Total FLOP Count for a Single Λ_j

Summing all operations for one Λ_j:

$$(j - 1 + 1 + 1 + 2) = (j + 2). \tag{A-27}$$

Since the entire vector Λ consists of f such values, summing over all $j = 1, \ldots, f$ gives:

$$\sum_{j=1}^{f}(j+2) = \left(\sum_{j=1}^{f} j + \sum_{j=1}^{f} 2\right). \tag{A-28}$$

Using the formula for the sum of the first f natural numbers:

$$\sum_{j=1}^{f} j = \frac{f(f+1)}{2}, \quad \sum_{j=1}^{f} 2 = 2f, \tag{A-29}$$

we obtain:

$$\left(\frac{f(f+1)}{2} + 2f\right). \tag{A-30}$$

Conclusion

Thus, the overall computational complexity for constructing the full Λ vector with feedthrough is:

$$\left(\frac{f(f+1)}{2} + 2f\right). \tag{A-31}$$

A-2-6 Updating Covariance

Updating the covariance contains two steps, the givens rotation or LQ factorization of a $(2p+2) \times (2p+2)$ takes $(2(2p+2)^3)$ and $\hat{\Xi}_{0_k} = \hat{\Xi}_{0_{k-1}} + \sqrt{\lambda_{exp}} G_k a^{-1}(y_k - \varphi_k^T \hat{\Xi}_{0_{k-1}})$.

- **Step 1: Square root of scalar** $\sqrt{\lambda_{exp}}$: This step takes (1) FLOP.
- **Step 2: Scalar-vector Multiplication** $\sqrt{\lambda_{exp}} G_k$: Here the calculated scalar is multiplied to $2p+1$ vector G_k and requires a $(2p+1)$ FLOPs.
- **Step 3: Inverse of scalar** a^{-1}: This step takes (1) FLOP.
- **Step 4: Vector-Scalar Multiplication** $\sqrt{\lambda_{exp}} G_k a^{-1}$: The multiplication of a $2p+1$ vector is multiplied with a scalar and requires a $(2p+1)$ FLOPs.
- **Step 5: Inner Product** $\varphi_k^T \hat{\Xi}_{0_{k-1}}$: Of two $2p+1$ vectors is $(2(2p+1)-1)$.
- **Step 6: Substition** $(y_k - \varphi_k^T \hat{\Xi}_{0_{k-1}})$: The substitution of 2 scalars (1) FLOP.
- **Step 7: Vector-Scalar Multiplication** $\sqrt{\lambda_{exp}} G_k a^{-1}(y_k - \varphi_k^T \hat{\Xi}_{0_{k-1}})$: The multiplication of a $2p+1$ vector with a scalar is $(2p+1)$ FLOPs.
- **Step 8: Addition of 2 vectors** $\hat{\Xi}_{0_{k-1}} + \sqrt{\lambda_{exp}} G_k a^{-1}(y_k - \varphi_k^T \hat{\Xi}_{0_{k-1}})$: The addition of two $2p+1$ vectors is $(2p+1)$ FLOPs.

Which totals up to $(12p+7)$ FLOPs for $\hat{\Xi}_{0_{k-1}} + \sqrt{\lambda_{exp}} G_k a^{-1}(y_k - \varphi_k^T \hat{\Xi}_{0_{k-1}})$ and in total for the whole covariance update we get: $(16p^3 + 48p^2 + 60p + 23)$

Appendix B

Additional Theoretical Details and Calculations

This appendix provides supplementary theoretical insights and derivations that support developing and understanding the proposed control framework. While the main chapters focus on implementation, performance, and experimental validation, this section delves deeper into several core components' mathematical underpinnings.

First, the concept of *directional forgetting* is explored, including its mathematical formulation and its role in recursive parameter estimation, as well as an efficient implementation using Givens rotations. Next, additional details are provided on the mechanics of *Givens rotations*, a numerically stable technique widely used for matrix factorization and covariance updates.

Furthermore, this appendix presents derivations of the Auto-Regressive with eXogenous input (ARX) models that underlie the system and controller structure.

Finally, an illustrative example explains how a *negative phase margin* may still be consistent with closed-loop stability, addressing a common point of confusion when using frequency-domain stability metrics such as those computed in MATLAB.

B-1 Directional Forgetting

In the work of Cao and Schwartz [99], a directional forgetting algorithm is introduced that operates on the decomposition of the information matrix $\mathcal{I}_k$, which is the inverse of the covariance matrix P_k. At each time step, this information matrix is partitioned as follows:

$$\mathcal{I}_k = \mathcal{I}_k^{(1)} + \mathcal{I}_k^{(2)}.$$

Van der Veen presents an alternative and insightful derivation in [50]. Given a regressor vector φ_k, we define the following orthogonal projection matrices:

$$\Pi_{\varphi_k} = \frac{\varphi_k \varphi_k^T}{\varphi_k^T \varphi_k}, \quad \Pi_{\varphi_k}^{\perp} = I - \Pi_{\varphi_k} = I - \frac{\varphi_k \varphi_k^T}{\varphi_k^T \varphi_k}.$$

These projections satisfy:

$$\Pi_{\varphi_k} \xi \parallel \varphi_k, \quad \Pi_{\varphi_k}^{\perp} \xi \perp \varphi_k,$$

for any arbitrary vector ξ. Hence, Π_{φ_k} projects vectors (or matrices) onto the direction of φ_k, while $\Pi_{\varphi_k}^{\perp}$ projects onto the orthogonal complement.

The core idea behind directional forgetting is to apply the forgetting factor λ_{dir} exclusively to the component of the information matrix aligned with φ_k. This strategy ensures that historical information is only discounted in directions where new information is introduced. As a result, the need for strong persistence of excitation is relaxed, making the method efficient and robust for recursive parameter estimation.

The resulting decomposition of the information matrix becomes:

$$\mathcal{I}_k = \mathcal{I}_k \underbrace{\Pi_{\varphi_k}^{\perp}}_{\perp \varphi_k} + \lambda_{dir} \mathcal{I}_k \underbrace{\Pi_{\varphi_k}}_{\parallel \varphi_k}. \tag{B-1}$$

To avoid directly computing the inverse of $\mathcal{I}_k$ at every iteration, Van der Veen proposes a numerically efficient method using two successive Givens rotations [50]. The second Givens rotation mirrors the covariance update depicted in Figure 2-2, but with $\lambda_{exp} = 1$.

Let the covariance matrix be decomposed as $P_k = R_k R_k^T$, where R_k is the lower-triangular Cholesky factor. A scaling factor α_k is computed as:

$$\alpha_k = \frac{1 - \lambda_{dir}}{\lambda_{dir}} \cdot \frac{1}{\left\| R_k^{-1} \varphi_k \right\|_2^2}. \tag{B-2}$$

Using these Givens rotations (illustrated in Figure B-1), the updated Cholesky factor R_k can be efficiently obtained, allowing the recursive update of the estimate $\hat{\Xi}_0$ as previously defined in Equation 2-45.

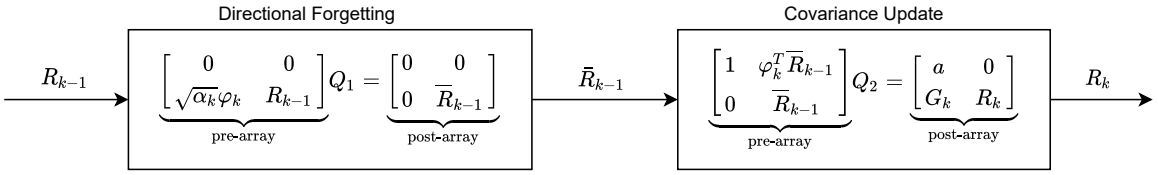


Figure B-1: Sequence of Givens rotations used in the directional forgetting algorithm, adapted from [50]

B-2 Givens Rotations

A Givens rotation introduces a zero in a specified matrix position while preserving its structure. This method is widely used in numerical linear algebra, particularly in QR decomposition and recursive least squares problems. The implementation follows the approach described in [60].

B-2-1 Overview

The operation is performed on a selected two-column submatrix:

- The first column serves as the reference column that remains after rotation.
- The second column contains the element that must be eliminated.
- A specific row index is selected where the elimination occurs.

B-2-2 Algorithm Description

The Givens rotation is computed as follows:

1. **Extract values:** Two elements from the matrix are selected, one from each column.
2. **Compute rotation parameters:** The rotation coefficients are determined numerically stable, ensuring that division by small numbers is avoided.
3. **Construct the rotation matrix:** A small 2×2 orthogonal matrix is formed based on the computed parameters.
4. **Apply the transformation:** The selected columns are modified using the rotation matrix, ensuring that the targeted element is set to zero while preserving numerical stability.

B-3 Guess of System (CL SPC)

$$Y_{i_p,1,\bar{N}} = C\tilde{A}^p X_i + \overline{\Xi}_0 \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,1\bar{N}} \end{bmatrix} \tag{B-3}$$

Assuming $C\tilde{A}^p X_i = 0$ and $\overline{\Xi}_0 = \begin{bmatrix} C\tilde{A}^{p-1}\tilde{B} & \cdots & C\tilde{B} & C\tilde{A}^{p-1}K & \cdots & CK & D \end{bmatrix}$

$$Y_{i_p,1,\bar{N}} = \begin{bmatrix} C\tilde{A}^{p-1}\tilde{B} & \cdots & C\tilde{B} & C\tilde{A}^{p-1}K & \cdots & CK & D \end{bmatrix} \begin{bmatrix} U_{i,p,\bar{N}} \\ Y_{i,p,\bar{N}} \\ U_{i_p,1\bar{N}} \end{bmatrix} \tag{B-4}$$

Taking the first only the first column of the data matrices

$$Y_{i_p,1,\bar{N}} = \begin{bmatrix} C\tilde{A}^{p-1}\tilde{B} & \cdots & C\tilde{B} \end{bmatrix} U_{i,p,\bar{N}}(:,1) + \begin{bmatrix} C\tilde{A}^{p-1}K & \cdots & CK \end{bmatrix} Y_{i,p,\bar{N}}(:,1) + DU_{i_p,1\bar{N}}(1) \tag{B-5}$$

Taking the z-transform

$$y(z) = C\tilde{A}^{p-1}\tilde{B}u(z)z^{-p-1} \ \cdots \ + C\tilde{B}u(z)z^{-1} + C\tilde{A}^{p-1}Ky(z)z^{-p-1} \ \cdots \ + CKy(z)z^{-1} + Du(z) \tag{B-6}$$

All terms with $y(z)$ to the left

$$y(z) - C\tilde{A}^{p-1}Ky(z)z^{-p-1} \ \cdots \ - CKy(z)z^{-1} = C\tilde{A}^{p-1}\tilde{B}u(z)z^{-p-1} \ \cdots \ + C\tilde{B}u(z)z^{-1} + Du(z) \tag{B-7}$$

Factor out $y(z)$ and $u(z)$.

$$(1 - C\tilde{A}^{p-1}Kz^{-p-1} \ \cdots \ - CKz^{-1})y(z) = (C\tilde{A}^{p-1}\tilde{B}z^{-p-1} \ \cdots \ + C\tilde{B}z^{-1} + D)u(z) \tag{B-8}$$

$$y(z) = \frac{C\tilde{A}^{p-1}\tilde{B}z^{-p-1} \ \cdots \ + C\tilde{B}z^{-1} + D}{1 - C\tilde{A}^{p-1}Kz^{-p-1} \ \cdots \ - CKz^{-1}} u(z) \tag{B-9}$$

$$y(z) = \frac{D + C\tilde{B}z^{-1} \ \cdots \ + C\tilde{A}^{p-1}\tilde{B}z^{-p-1}}{1 - CKz^{-1} \ \cdots \ - C\tilde{A}^{p-1}Kz^{-p-1}} u(z) \tag{B-10}$$

B-4 Guess of Controller

$$U_{\hat{i}_p,f,1} = \left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q}(r_f - \Gamma\mathcal{K}w_p) \tag{B-11}$$

r_f is a zero vector.

$$U_{\hat{i}_p,f,1} = -\left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q}\Gamma\mathcal{K}w_p \tag{B-12}$$

Simplify with:

$$\alpha = -\left(\tilde{R} + H_{(B,D)}^T \tilde{Q} H_{(B,D)}\right)^{-1} H_{(B,D)}^T \tilde{Q} \Gamma \mathcal{K} \tag{B-13}$$

$$U_{\hat{i}_p,f,1} = \alpha w_p \tag{B-14}$$

We focus mainly on the first row because this is the only input we give to the system:

$$u_{k+1} = \alpha(1,:) w_p \tag{B-15}$$

$$u_{k+1} = \alpha(1,1:p) Y_{\hat{i},p,1} + \alpha(1,p+1:2p) U_{\hat{i},p,1} \tag{B-16}$$

If we take the z transform:

$$u = \alpha(1,1) y z^{-p-1} + \cdots + \alpha(1,p) y z^{-1} + \alpha(1,p+1) u z^{-p-1} + \cdots + \alpha(1,2p) u z^{-1} \tag{B-17}$$

All terms with u to the left:

$$u - \alpha(1,p+1) u z^{-p-1} - \cdots - \alpha(1,2p) u z^{-1} = \alpha(1,1) y z^{-p-1} + \cdots + \alpha(1,p) y z^{-1} \tag{B-18}$$

Factor out u:

$$(1 - \alpha(1,p+1) z^{-p-1} - \cdots - \alpha(1,2p) z^{-1}) u = (\alpha(1,1) z^{-p-1} + \cdots + \alpha(1,p) z^{-1}) y \tag{B-19}$$

$$u(z) = \frac{\alpha(1,1) z^{-p-1} + \cdots + \alpha(1,p) z^{-1}}{1 - \alpha(1,p+1) z^{-p-1} - \cdots - \alpha(1,2p) z^{-1}} y(z) \tag{B-20}$$

$$u(z) = \frac{\alpha(1,p) z^{-1} + \cdots + \alpha(1,1) z^{-p-1}}{1 - \alpha(1,2p) z^{-1} - \cdots - \alpha(1,p+1) z^{-p-1}} y(z) \tag{B-21}$$

B-5 Negative Phase Margin

Due to the way MATLAB calculates the Phase Margin (PM), it is possible to obtain a negative phase margin even when the system remains stable. This does not necessarily imply instability, as stability is ultimately determined by the Nyquist criterion rather than the phase margin alone.

Figure B-2 presents a simple illustrative example demonstrating how a system can have a negative PM while still being stable—i.e., the Nyquist plot does not encircle the critical point -1. In this example, both the green and red curves exhibit similar overall stability margins, even though the red curve is associated with a negative phase margin according to MATLAB's convention.

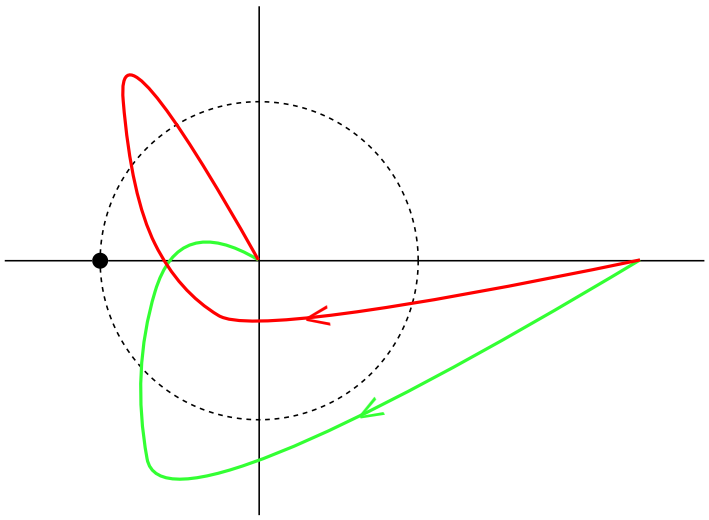


Figure B-2: Illustrative example showing a Nyquist plot with a positive phase margin (green) and a negative phase margin (red), both having approximately the same overall stability margins.

Appendix C

Additional Results

This appendix presents supplementary results that support and extend the findings discussed in the main chapters. These results provide deeper insight into specific theoretical and empirical aspects of the work. The appendix is structured into three sections:

- **Closed-Loop SPC with or without Feedthrough:** A comparison of control performance with and without the feedthrough term D in the system model, highlighting its effect under different noise realizations.
- **Model Estimation Results:** Visualizations of time-varying system identification results using different past window sizes p illustrate the model order's influence on estimation quality and variability.
- **Controller Estimation Results:** Stability analysis of estimated controllers, showing pole locations under various tuning parameters and demonstrating the sensitivity of Closed Loop (CL) stability to design choices.

C-1 Closed-Loop SPC with or without Feedthrough

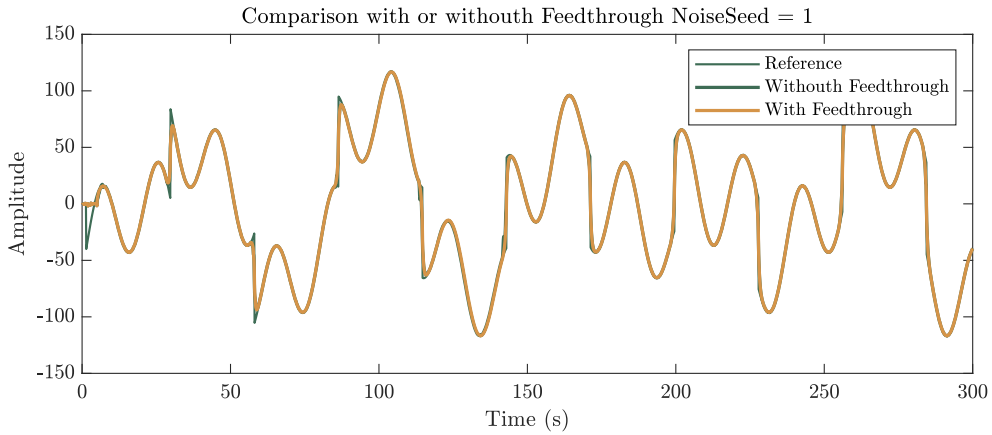


Figure C-1: Comparison of closed-loop control simulations for systems with and without a feedthrough matrix ($D = 0$ vs. $D \neq 0$). Results shown for the first noise seed.

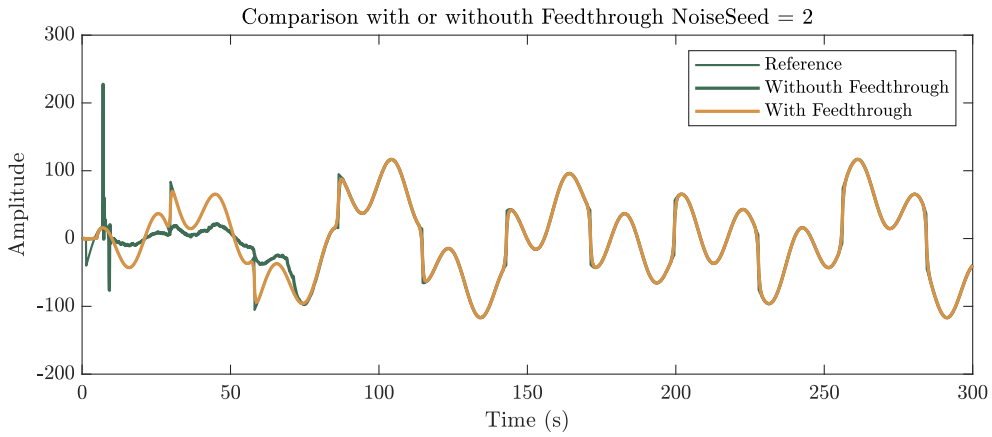


Figure C-2: Comparison of closed-loop control simulations for systems with and without a feedthrough matrix ($D = 0$ vs. $D \neq 0$). Results shown for the second noise seed.

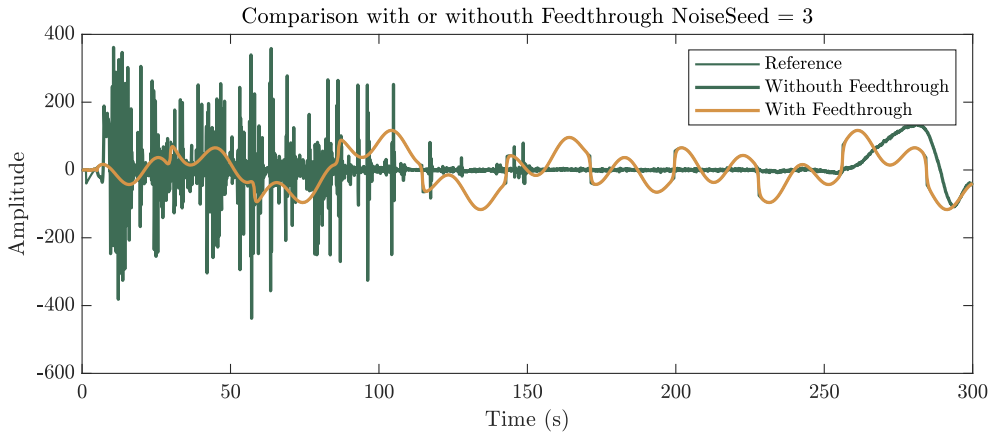


Figure C-3: Comparison of closed-loop control simulations for systems with and without a feedthrough matrix ($D = 0$ vs. $D \neq 0$). Results shown for the third noise seed.

C-2 Model Estimation Results

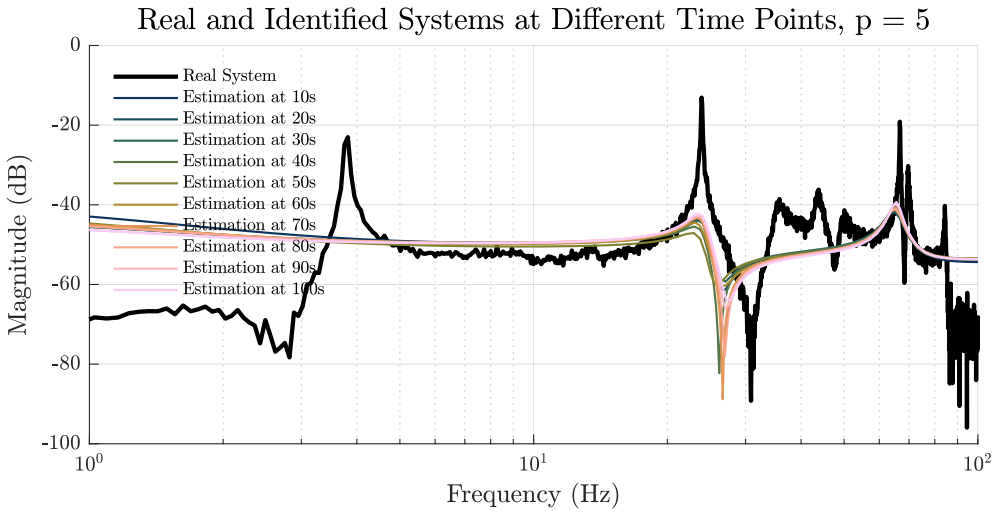


Figure C-4: Time-varying estimated system response for window size $p = 5$, evaluated at 10 distinct time instances.

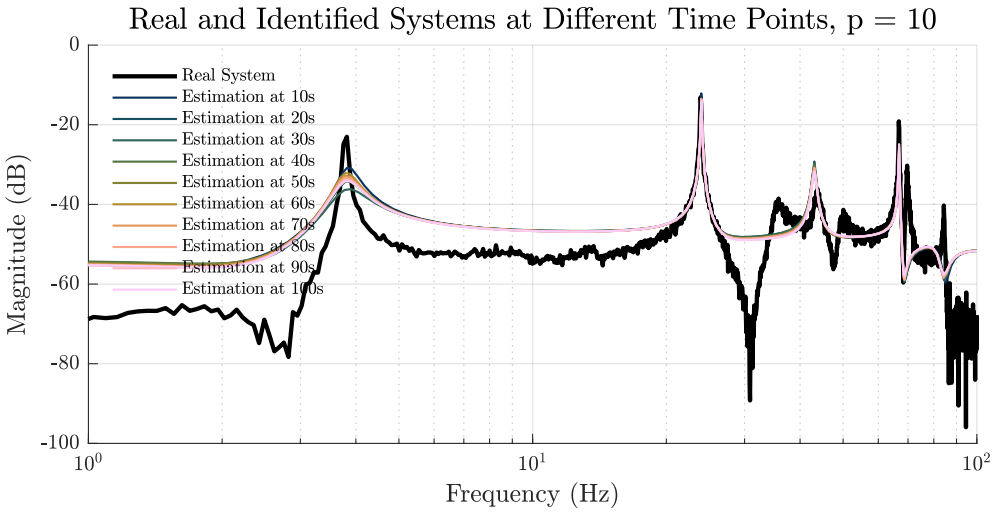


Figure C-5: Time-varying estimated system response for window size $p = 10$, evaluated at 10 distinct time instances.

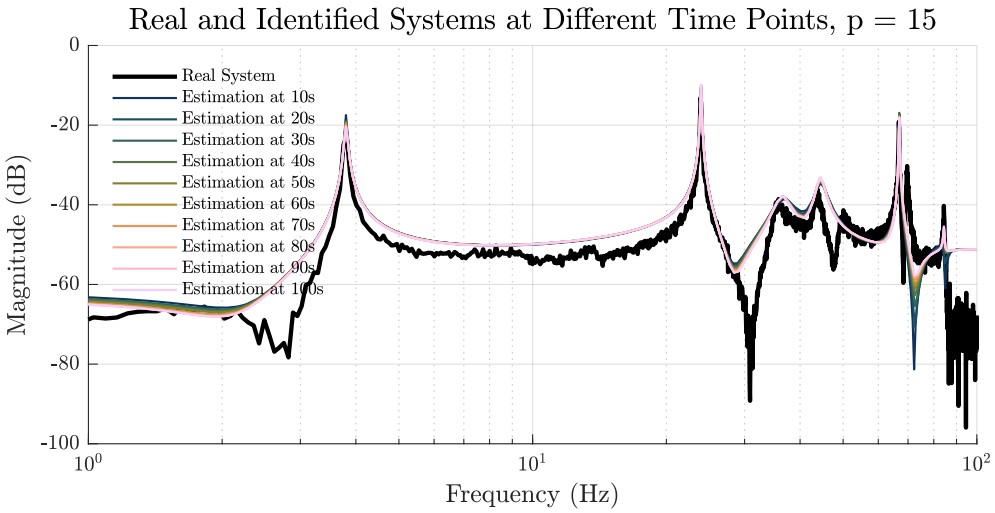


Figure C-6: Time-varying estimated system response for window size $p = 15$, evaluated at 10 distinct time instances.

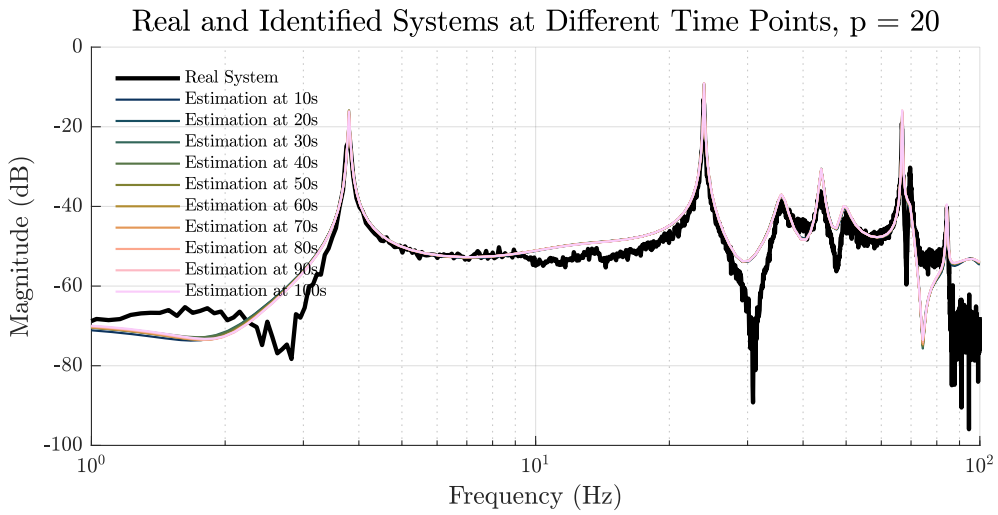


Figure C-7: Time-varying estimated system response for window size $p = 20$, evaluated at 10 distinct time instances.

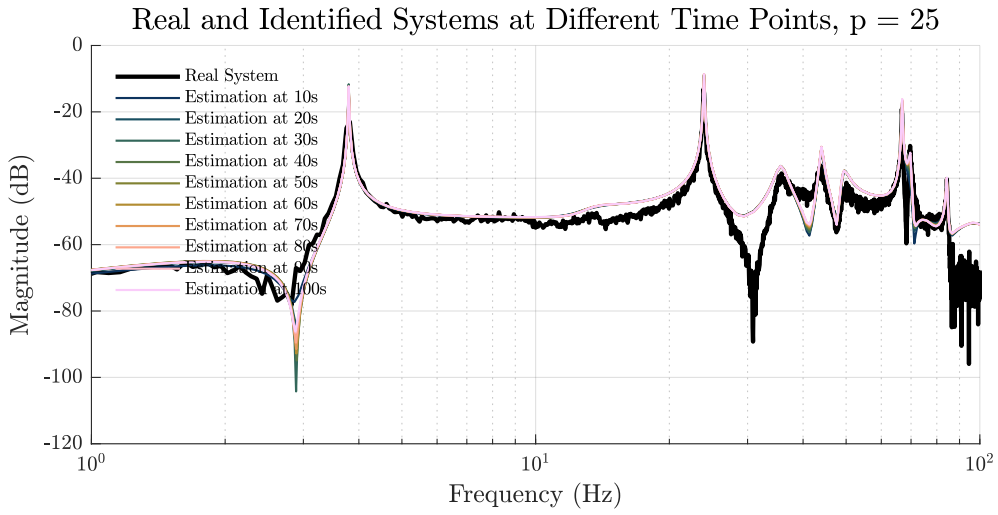


Figure C-8: Time-varying estimated system response for window size $p = 25$, evaluated at 10 distinct time instances.

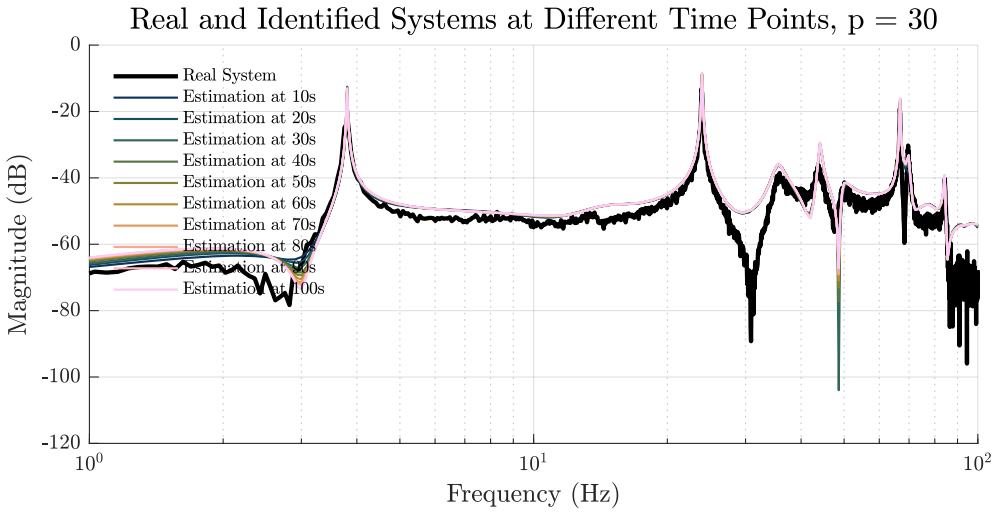


Figure C-9: Time-varying estimated system response for window size $p = 30$, evaluated at 10 distinct time instances.

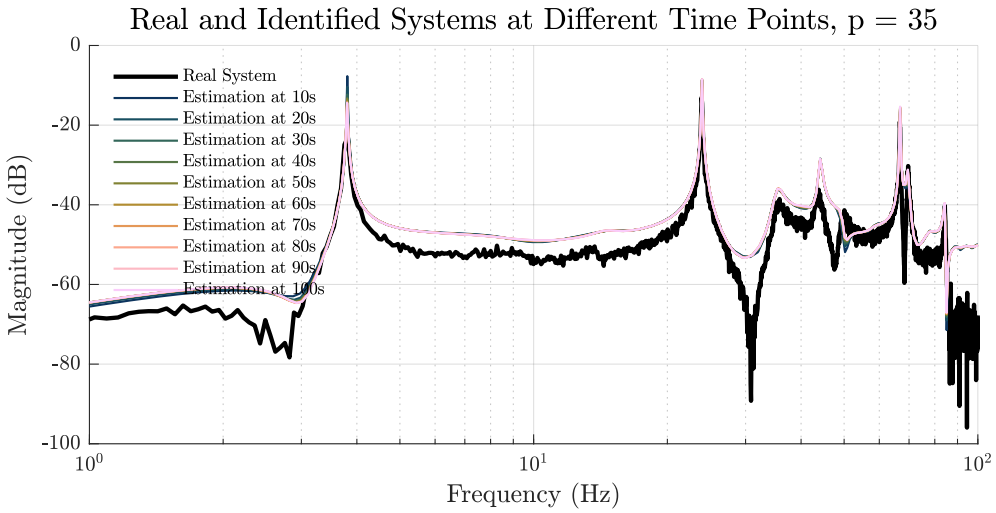


Figure C-10: Time-varying estimated system response for window size $p = 35$, evaluated at 10 distinct time instances.

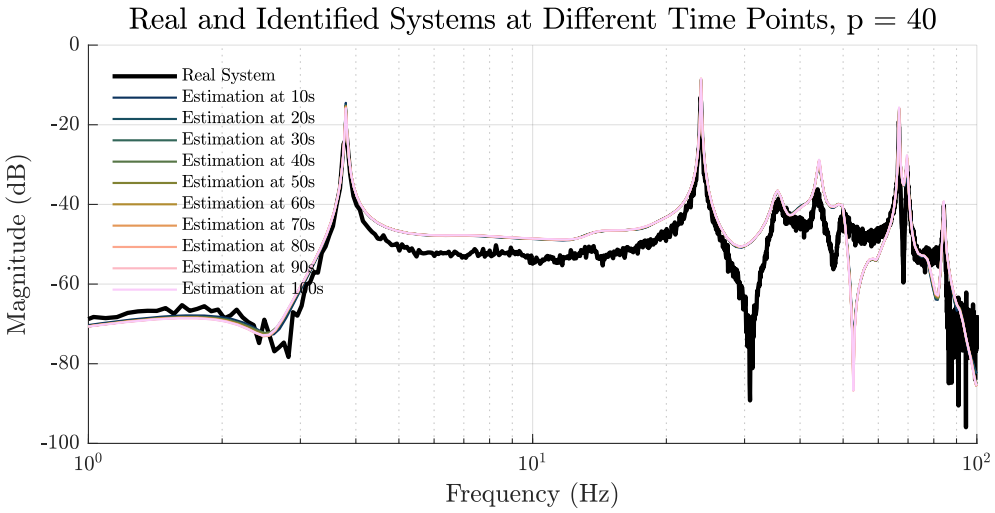


Figure C-11: Time-varying estimated system response for window size $p = 40$, evaluated at 10 distinct time instances.

C-3 Controller Estimation Results

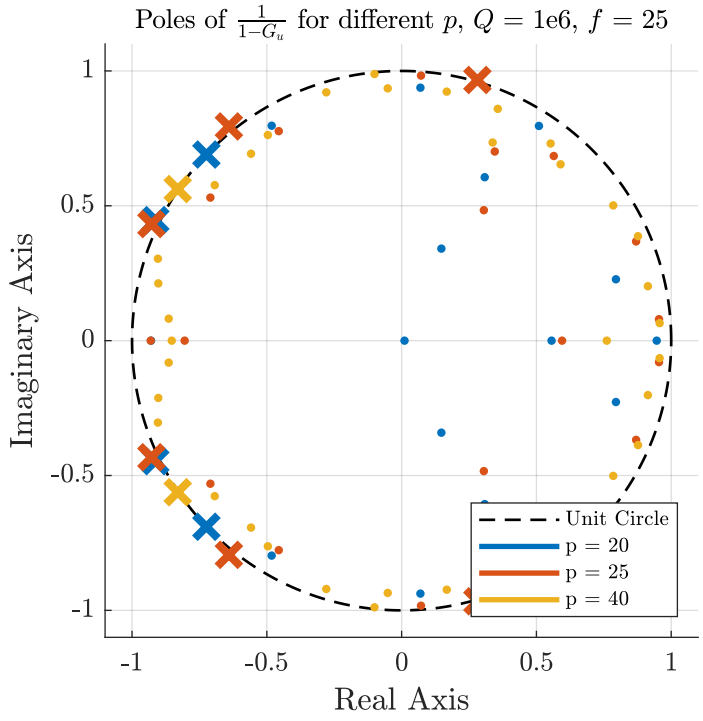


Figure C-12: Pole map of estimated controllers for varying window size p, with fixed $Q = 1 \times 10^6$, $f = 25$, and $R = 1$. Crosses denote unstable poles. All controllers in this set are unstable.

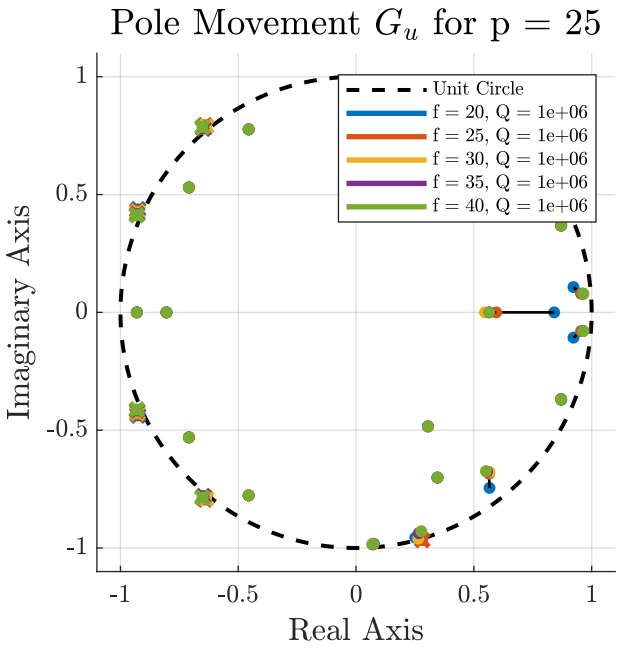


Figure C-13: Pole map of estimated controllers for varying control horizon f, with fixed $Q = 1 \times 10^6$, $p = 25$, and $R = 1$. Crosses denote unstable poles. All controllers in this set are unstable.

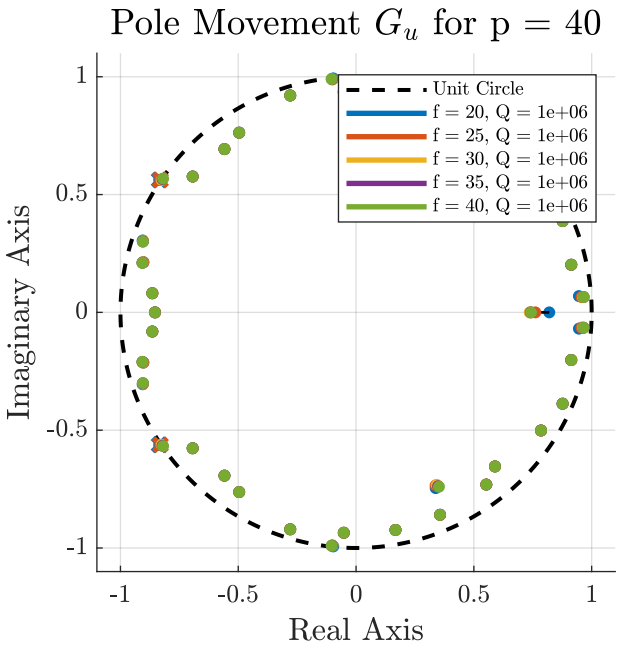


Figure C-14: Pole evolution of controller G_u for increasing values of control horizon f, with fixed $Q = 1 \times 10^6$, $p = 40$, and $R = 1$. A previously unstable pole becomes stable at $f = 30$.

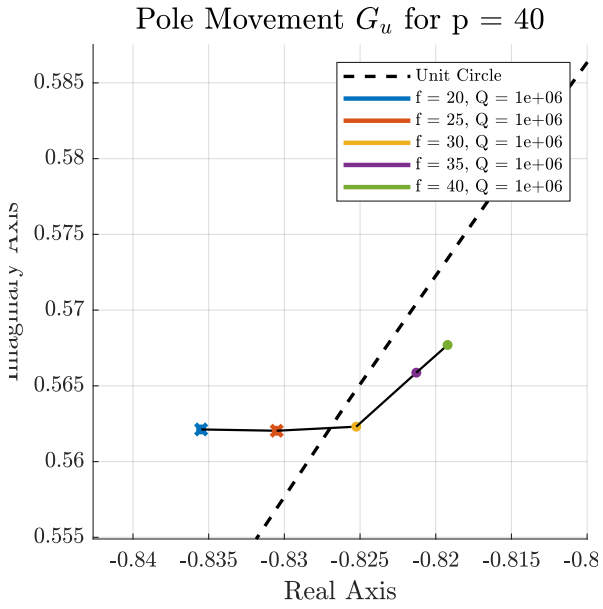


Figure C-15: Zoomed view of pole evolution in controller G_u, showing stabilization of a previously unstable pole at $f = 30$, for $Q = 1 \times 10^6$, $p = 40$, and $R = 1$.

Bibliography

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960.

[2] R. E. Kalman, "Contributions to the theory of optimal control," 1960.

[3] I. Markovsky, L. Huang, and F. Dorfler, "Data-Driven Control Based on the Behavioral Approach: From Theory to Applications in Power Systems," *IEEE Control Systems*, vol. 43, no. 5, pp. 28–68, 2023.

[4] A. Moallemi, R. Gaspari, F. Zonzini, L. De Marchi, D. Brunelli, and L. Benini, "Speeding up System Identification Algorithms on a Parallel RISC-V MCU for Fast Near-Sensor Vibration Diagnostic," *IEEE Sensors Letters*, vol. 7, no. 9, pp. 1–4, 2023.

[5] A. Bemporad, "Linear and nonlinear system identification under ℓ_1- and group-Lasso regularization via L-BFGS-B," pp. 1–23, 2024.

[6] N. K. Sinha, "Reduced-order models for linear systems," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 1992-Janua, pp. 537–542, 1992.

[7] D.-W. Gu, P. H. Petkov, and M. M. Konstantinov, "Lower-Order Controllers," in *Advanced Textbooks in Control and Signal Processing*, no. 9781447146810, pp. 73–91, 2013.

[8] Z. S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[9] G. Baggio, D. S. Bassett, and F. Pasqualetti, "Data-driven control of complex networks," *Nature Communications*, vol. 12, no. 1, 2021.

[10] W. Favoreel, B. D. Moor, and M. Gevers, "SPC: Subspace Predictive Control," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, 1999.

[11] J. Coulson, J. Lygeros, and F. Dorfler, "Data-enabled predictive control: In the shallows of the deepc," in *2019 18th European Control Conference, ECC 2019*, pp. 307–312, 2019.

[12] E. Elokda, J. Coulson, P. N. Beuchat, J. Lygeros, and F. Dörfler, "Data-enabled predictive control for quadcopters," *International Journal of Robust and Nonlinear Control*, vol. 31, pp. 8916–8936, 12 2021.

[13] P. Van Overschee and B. De Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, pp. 75–93, 1 1994.

[14] P. Kabaila, "On output-error methods for system identification," *IEEE Transactions on Automatic Control*, vol. 28, pp. 12–23, 1 1983.

[15] K. Pelckmans, "Prediction Error Methods," no. Ml, pp. 101–111, 2012.

[16] J. Sun, Y. Huang, W. Yu, and A. Garcia-Ortiz, "Nonlinear System Identification: Prediction Error Method vs Neural Network," *2021 10th International Conference on Modern Circuits and Systems Technologies, MOCAST 2021*, pp. 8–11, 2021.

[17] N. Kriegeskorte and T. Golan, "Neural network models and deep learning," *Current Biology*, vol. 29, no. 7, pp. R231–R236, 2019.

[18] L. Shi and X. C. Wang, "The application of neural network in nonlinear system," *Advanced Materials Research*, vol. 179-180, pp. 128–134, 2011.

[19] B. Lü, J. Murata, and K. Hirasawa, "A new learning method using prior information of neural networks," *Science in China, Series F: Information Sciences*, vol. 47, no. 6, pp. 793–814, 2004.

[20] H. K. Jabbar and R. Z. Khan, "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)," pp. 163–172, 2015.

[21] M. N. Gurcan, B. Sahiner, H. P. Chan, L. Hadjiiski, and N. Petrick, "Selection of an optimal neural network architecture for computer-aided detection of microcalcifications - Comparison of automated optimization techniques," *Medical Physics*, vol. 28, no. 9, pp. 1937–1948, 2001.

[22] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep networks for system identification: a Survey," no. January, 2023.

[23] M. Verhaegen and P. Dewilde, "Subspace model identification Part 1. The output-error state-space model identification class of algorithms," *International Journal of Control*, vol. 56, no. 5, pp. 1187–1210, 1992.

[24] M. Döhler and L. Mevel, "Fast multi-order computation of system matrices in subspace-based system identification," *Control Engineering Practice*, vol. 20, no. 9, pp. 882–894, 2012.

[25] W. Favoreel, B. De Moor, and P. Van Overschee, "Subspace state space system identification for industrial processes," *Journal of Process Control*, vol. 10, no. 2, pp. 149–155, 2000.

[26] S. J. Qin, "An overview of subspace identification," *Computers and Chemical Engineering*, vol. 30, no. 10-12, pp. 1502–1513, 2006.

[27] D. Kirk, "Optimal control theory: An introduction," 2004.

[28] F. Dorfler, P. Tesi, and C. De Persis, "On the Certainty-Equivalence Approach to Direct Data-Driven LQR Design," *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 7989–7996, 2023.

[29] N. Lehtomaki, N. Sandell, and M. Athans, "Robustness results in linear-quadratic Gaussian based multivariable control designs," *IEEE Transactions on Automatic Control*, vol. 26, pp. 75–93, 2 1981.

[30] B. Anderson and J. Moore, *Optimal control: Linear quadratic methods*. Prentice-Hall, 1990.

[31] R. R. Negenborn and J. M. Maestre, "Distributed model predictive control: An overview and roadmap of future research opportunities," *IEEE Control Systems*, vol. 34, no. 4, pp. 87–97, 2014.

[32] M. Alamir and F. Allgöwer, *Model predictive control*, vol. 18. 2008.

[33] J. Hu and B. Ding, "One-step ahead robust MPC for LPV model with bounded disturbance," *European Journal of Control*, vol. 52, pp. 59–66, 2020.

[34] P. J. Goulart, E. C. Kerrigan, and D. Ralph, "Efficient robust optimization for robust control with constraints," *Mathematical Programming*, vol. 114, no. 1, pp. 115–147, 2008.

[35] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," *2013 European Control Conference, ECC 2013*, pp. 502–510, 2013.

[36] E. Skjong, T. A. Johansen, and M. Molinas, "Distributed control architecture for real-time model predictive control for system-level harmonic mitigation in power systems," *ISA Transactions*, vol. 93, pp. 231–243, 2019.

[37] J. Maciejowski, P. Goulart, and E. Kerrigan, "Constrained Control Using Model Predictive Control," in *Advanced Strategies in Control Systems with Input and Output Constraints*, vol. 360, pp. 273–291, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[38] F. Dorfler, J. Coulson, and I. Markovsky, "Bridging Direct and Indirect Data-Driven Control Formulations via Regularizations and Relaxations," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 883–897, 2023.

[39] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Solà, N. Mansard, and S. Vijayakumar, "A feasibility-driven approach to control-limited DDP," *Autonomous Robots*, vol. 46, no. 8, pp. 985–1005, 2022.

[40] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems and Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[41] M. Alsalti, V. G. Lopez, and M. A. Muller, "On the Design of Persistently Exciting Inputs for Data-Driven Control of Linear and Nonlinear Systems," *IEEE Control Systems Letters*, vol. 7, pp. 2629–2634, 2023.

[42] J. Dong and M. Verhaegen, "Cautious H2 optimal control using uncertain Markov parameters identified in closed loop," *Systems and Control Letters*, vol. 58, no. 5, pp. 378–388, 2009.

[43] L. Ljung and T. McKelvey, "Subspace identification from closed loop data," *Signal Processing*, vol. 52, no. 2, pp. 209–215, 1996.

[44] M. Lazar and P. C. N. Verheijen, "Generalized Data-driven Predictive Control," 2023.

[45] F. Fiedler and S. Lucia, "On the relationship between data-enabled predictive control and subspace predictive control," 2021.

[46] J. W. van Wingerden, S. P. Mulders, R. Dinkla, T. Oomen, and M. Verhaegen, "Data-enabled predictive control with instrumental variables: the direct equivalence with subspace predictive control," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2022-Decem, pp. 2111–2116, 2022.

[47] R. Dinkla, S. Mulders, T. Oomen, and J. W. van Wingerden, "Closed-loop Data-Enabled Predictive Control and its equivalence with Closed-loop Subspace Predictive Control," no. February, 2024.

[48] R. Dinkla, S. P. Mulders, J. W. van Wingerden, and T. Oomen, "Closed-loop Aspects of Data-Enabled Predictive Control," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1388–1393, 2023.

[49] F. Wegner, J. Coulson, M. Hudoba de Badyn, J. Lygeros, and J. S. Trimpe, "Data-enabled Predictive Control of Robotic Systems," 2021.

[50] G. van der Veen, *Identification of wind energy systems*. PhD thesis, TU Delft, 2013.

[51] G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Tikhonov regularization and total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 1, pp. 185–194, 1999.

[52] J. Mairal and B. Yu, "Complexity analysis of the lasso regularization path," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, no. 1952, pp. 353–360, 2012.

[53] A. Sassella, V. Breschi, and S. Formentin, "A practitioner's guide to noise handling strategies in data-driven predictive control," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1382–1387, 2023.

[54] S. Haykin and J. Principe, "Making sense of a complex world [chaotic events modeling]," *IEEE Signal Processing Magazine*, vol. 15, pp. 66–81, 5 1998.

[55] A. Sassella, S. Formentin, A. Sassella, and P. It, "Noise handling in data-driven predictive control : a strategy based on dynamic mode decomposition," vol. 168, no. 2020, pp. 1–12, 2022.

[56] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic Mode Decomposition with Control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, pp. 142–161, 1 2016.

[57] Y. Arkun and G. Stephanopoulos, "Studies in the synthesis of control structures for chemical processes. Part V: Design of steady-state optimizing control structures for integrated chemical plants," *AIChE Journal*, vol. 27, pp. 779–793, 9 1981.

[58] V. Egunov and A. Andreev, "Implementation of QR and LQ decompositions on shared memory parallel computing systems," *2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2016 - Proceedings*, pp. 1–5, 2016.

[59] P. Strandmark, F. Kahl, and T. Schoenemann, "Parallel and distributed vision algorithms using dual decomposition," *Computer Vision and Image Understanding*, vol. 115, no. 12, pp. 1721–1732, 2011.

[60] D. F. Mayers, G. H. Golub, and C. F. van Loan, *Matrix Computations.*, vol. 47. 1986.

[61] V. Breschi, A. Chiuso, and S. Formentin, "Data-driven predictive control in a stochastic setting: a unified framework," *Automatica*, vol. 152, p. 110961, 6 2023.

[62] Y.-G. XI, D.-W. LI, and S. LIN, "Model Predictive Control — Status and Challenges," *Acta Automatica Sinica*, vol. 39, no. 3, pp. 222–236, 2013.

[63] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 16–31, 2014.

[64] Y. Yao and D. K. Shekhar, "State of the art review on model predictive control (MPC) in Heating Ventilation and Air-conditioning (HVAC) field," *Building and Environment*, vol. 200, no. February, p. 107952, 2021.

[65] Y. Ding, L. Wang, Y. Li, and D. Li, "Model predictive control and its application in agriculture: A review," *Computers and Electronics in Agriculture*, vol. 151, no. May, pp. 104–117, 2018.

[66] V. Vajpayee, S. Mukhopadhyay, and A. P. Tiwari, "Data-Driven Subspace Predictive Control of a Nuclear Reactor," *IEEE Transactions on Nuclear Science*, vol. 65, no. 2, pp. 666–679, 2018.

[67] N. A. Mardi and L. Wang, "Subspace-based model predictive control of time-varying systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 4005–4010, 2009.

[68] J. K. Hromatko, M. Svec, and S. Iles, "Autonomous Path Following using Data-Driven Predictive Control," *2023 27th International Conference on System Theory, Control and Computing, ICSTCC 2023 - Proceedings*, pp. 368–373, 2023.

[69] B. R. Woodley, *Model free subspace based H_inf control*. PhD thesis, Stanford University, 2001.

[70] R. Hunger, "Floating Point Operations in Matrix-Vector Calculus," *Technische Universität München*, vol. Version 1., p. 16, 2007.

[71] B. Zargar, F. Ponci, and A. Monti, "Evaluation of Computational Complexity for Distribution Systems State Estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, no. 1557, 2023.

[72] R. Kadali, B. Huang, and A. Rossiter, "A data driven subspace approach to predictive controller design," *Control Engineering Practice*, vol. 11, no. 3, pp. 261–278, 2003.

[73] J. Dong, M. Verhaegen, and E. Holweg, "Closed-loop Subspace Predictive Control for Fault Tolerant MPC Design," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3216–3221, 2008.

[74] G. van der Veen, J. W. van Wingerden, M. Bergamasco, M. Lovera, and M. Verhaegen, "Closed-loop subspace identification methods: An overview," *IET Control Theory and Applications*, vol. 7, no. 10, pp. 1339–1358, 2013.

[75] J. Dong, *Data Driven Fault Tolerant Control: A Subspace Approach*. 2009.

[76] I. Houtzager, J. W. van Wingerden, and M. Verhaegen, "Recursive predictor-based subspace identification with application to the real-time closed-loop tracking of flutter," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 4, pp. 934–949, 2012.

[77] K. J. Astrom and B. Wittenmark, "Adaptive Control (2nd ed.).," 2008.

[78] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

[79] AG Embotech, "FORCESPRO." https://forces.embotech.com/.

[80] A. Vince, "A framework for the greedy algorithm," *Discrete Applied Mathematics*, vol. 121, no. 1-3, pp. 247–260, 2002.

[81] A. Tregubov, P. Karamanakos, and L. Ortombina, "A Computationally Efficient Robust Direct Model Predictive Control for Medium Voltage Induction Motor Drives," *2021 IEEE Energy Conversion Congress and Exposition, ECCE 2021 - Proceedings*, pp. 4690–4697, 2021.

[82] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model Predictive Control of Power Electronic Systems: Methods, Results, and Challenges," *IEEE Open Journal of Industry Applications*, vol. 1, no. August, pp. 95–114, 2020.

[83] C. Moler, "MATLAB Incorporates LAPACK," 2000.

[84] V. Strassen, "Gaussian elimination is not optimal," *Matematika*, vol. 14, no. 3, pp. 127–128, 1970.

[85] N. J. Higham, "Exploiting Fast Matrix Multiplication Within the Level 3 BLAS," *ACM Transactions on Mathematical Software (TOMS)*, vol. 16, no. 4, pp. 352–368, 1990.

[86] X. Zhang, Q. Wang, and Y. Zhang, "Model-driven level 3 BLAS performance optimization on Loongson 3A processor," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pp. 684–691, 2012.

[87] J. M. Gere and B. J. Goodno, *Mechanics of Materials, Brief Edition*. No. 112, Stamford: Global Engineering.

[88] Z. Chen, H. J. Bong, D. Li, and R. H. Wagoner, "The elastic-plastic transition of metals," *International Journal of Plasticity*, vol. 83, pp. 178–201, 2016.

[89] J. Åberg and B. Widell, "Uniaxial Material Damping Measurements Using a Fiber Optic Lattice: A Discussion of its Performance Envelope," *Experimental Mechanics*, vol. 44, no. 1, pp. 33–36, 2004.

[90] D. J. Inman, *Engineering Vibrations Fourth edition*. 2001.

[91] E. E. Christopher, M. G. C, and O. Chinedu, "Analysis and Relevance of Z-Transform in Discrete Time Systems," no. July, 2023.

[92] J. W. van Wingerden, "SC42145 : Preliminaries Robust Control," 2022.

[93] S. S. Das, Y. Shtessel, and F. Plestan, "Phase and Gain Stability Margins for a class of Nonlinear Systems," *IFAC-PapersOnLine*, vol. 51, no. 25, pp. 263–268, 2018.

[94] Z. Y. Nie, Q. G. Wang, M. Wu, and Y. He, "Combined gain and phase margins," *ISA Transactions*, vol. 48, no. 4, pp. 428–433, 2009.

[95] A. Falcoz, C. Pittet, S. Bennani, A. Guignard, C. Bayart, and B. Frapard, "Systematic design methods of robust and structured controllers for satellites: Application to the refinement of Rosetta's orbit controller," *CEAS Space Journal*, vol. 7, no. 3, pp. 319–334, 2015.

[96] E. Kreindler, "On the definition and application of the sensitivity function," *Journal of the Franklin Institute*, vol. 285, pp. 26–36, 1 1968.

[97] M. C. Smith, "On the generalized nyquist stability criterion," *International Journal of Control*, vol. 34, no. 5, pp. 885–920, 1981.

[98] Y. Cho and T. Kailath, "Application of Fast Subspace Decomposition to Signal Processing and Communication Problems," *IEEE Transactions on Signal Processing*, vol. 42, no. 6, pp. 1453–1461, 1994.

[99] L. Cao and H. Schwartz, "Directional forgetting algorithm based on the decomposition of the information matrix," *Automatica*, vol. 36, no. 11, pp. 1725–1731, 2000.

Glossary

List of Acronyms

ARX	Auto-Regressive with eXogenous input
BLAS	Basic Linear Algebra Subprograms
CL	Closed Loop
CL SPC	Closed Loop Subspace Predictive Control
CR-CL SPC	Constrained Recursive Closed Loop Subspace Predictive Control
DDPC	Data-Driven Predictive Control
DeePC	Data-Enabled Predictive Control
DMD	Dynamic Mode Decomposition
FLOPs	Floating Point Operations
GM	Gain Margin
I/O	Input-Output
IV	Instrumental Variables
LTI	Linear Time-Invariant
LQR	Linear Quadratic Regulator
MBC	Model Based Control
MFC	Macro Fiber Composite
MOESP	Multivariable Output Error State Space
MPC	Model Predictive Control
MIMO	Multi-input Multi-output
N4SID	Numerical Algorithms for Subspace State Space System Identification
NN	Neural Network
PE	Persistance of Excitation
PEM	Prediction Error Method

PM	Phase Margin
PRBS	Pseudo-Random Binary Sequence
R-CL SPC	Recursive Closed Loop Subspace Predictive Control
RLS	Recursive Least Squares
SIM	Subspace Identification Methods
SIMD	Single Instruction, Multiple Data
SISO	Single-Input Single-Output
SPC	Subspace Predictive Control
SVD	Singular Value Decomposition
SysID	System Identification
QP	Quadratic Programming

List of Symbols

Γ	Observability matrix
λ_{exp}	Exponential Forgetting Factor
σ^2	Variance
$\bar{N}$	Hankel Matrix Width
$\hat{\bar{N}}$	Effective Window Length
$\mathcal{K}$	Reversed Extended Controllability matrix
f	Future Window Size
$H_{(B,D)}$	Block-Toeplitz matrix
p	Past Window Size
P_k	Covariance matrix at time k
Q	Penalty on state deviation
R	Penalty on control effort
R_k	Lower Triangular Cholesky factor of the Covariance matrix at time k
R_Δ	Penalty on control deviation