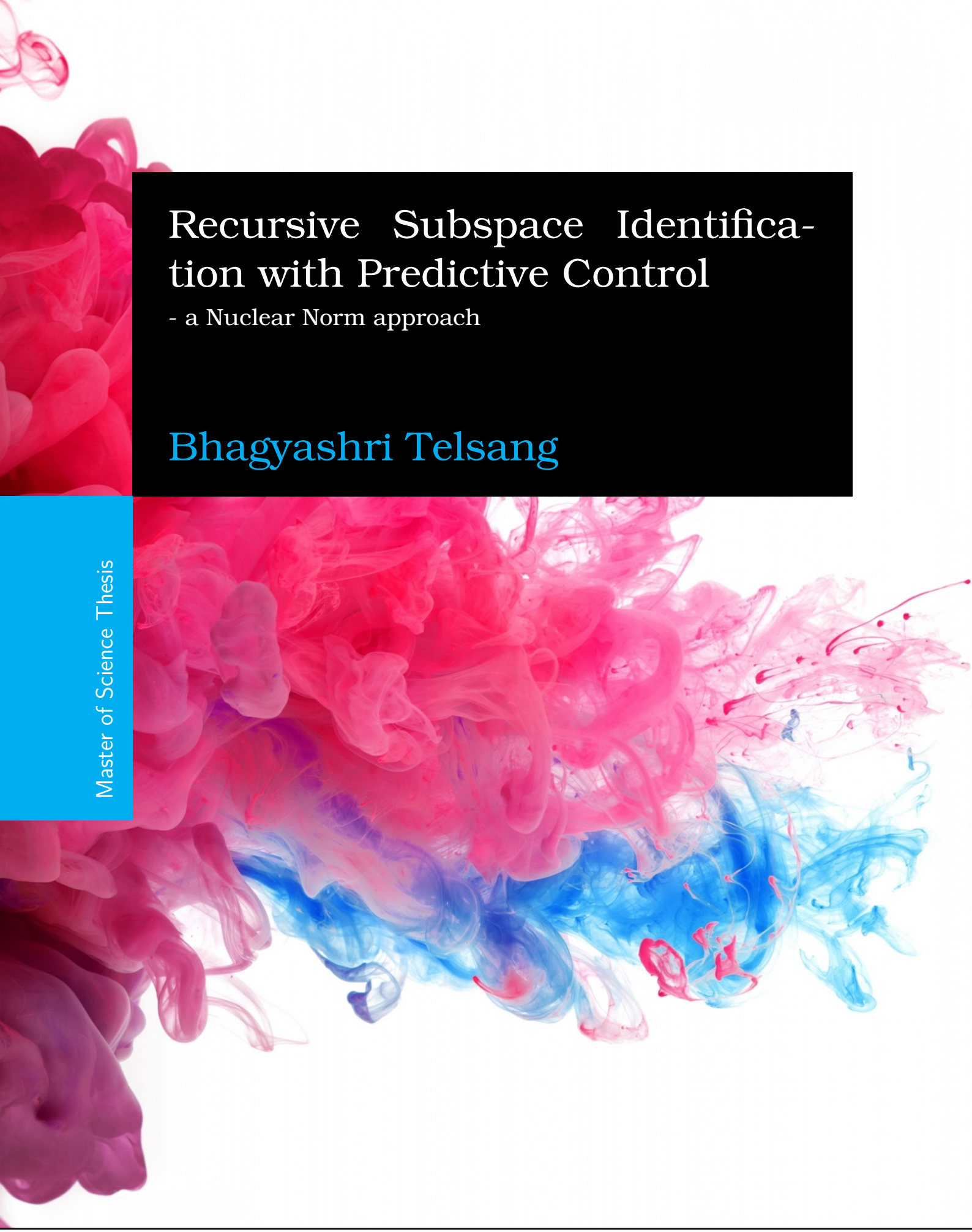
Recursive Subspace Identification with Predictive Control

- a Nuclear Norm approach

Bhagyashri Telsang

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Recursive Subspace Identification with Predictive Control

- a Nuclear Norm approach

Master of Science Thesis

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

Bhagyashri Telsang

August 22, 2016

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The main contribution of this thesis is the development of an inherently adaptive controller which recursively implements a system identification and control routine. The main idea here, is that if system identification techniques can be made computationally light, and if they can be combined efficiently with a controller formulation, such a framework can be implemented on real world systems to achieve the adaptive nature in control.

In the first part of the thesis, we focus on developing a computationally less expensive identification technique, by making various modifications on the N2SID algorithm, a nuclear norm subspace identification method. By allowing for early stopping of the ADMM algorithm (which plays a crucial role in N2SID) and by implementing an efficient alternative to Singular Value Thresholding, we are able to obtain significant reductions in computation time. Furthermore, we allow for a recursive implementation of N2SID by utilizing system information from the previous identification cycle. This results in a significant improvement of the convergence speed of the identification.

In order to create an efficient interface between the identification technique and the controller, we formulate an MPC control framework, which does not require the explicit computation of the system matrices at each identification cycle. This helps in reducing the computation time of the recursive algorithm. The overall methodology is such that an inherently adaptive controller is in play at every time instant. To test this algorithm, we implement it on different systems, including LTI and LPV systems.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank prof. dr. ir. Jan-Willem van Wingerden for giving me the opportunity to carry out the work and offering his constant guidance. I would also like to thank Sachin Navalkar for regularly supervising the work and providing assistance during the writing of this thesis.

Delft, University of Technology
August 22, 2016

Bhagyashri Telsang

Chapter 1

Introduction

"... if every instrument could accomplish its own work, obeying or anticipating the will of others ... if the shuttle weaved and the pick touched the lyre without a hand to guide them, chief workmen would not need servants, nor masters slaves."

This quote by Aristotle in his monograph "Politics" describes the guiding principle of control theory: the need to automatize processes [1]. Control theory, although existed earlier, mostly advanced in the way as we know today during the first few decades of 20^{th} century due to various developments – Wright brothers made four flights with gentle landings on *Flyer* on 17^{th} Dec 1903, guidance systems and electronics of World War II etc. Control theory has been a discipline where many mathematical ideas and methods have melted to produce a new body. Accordingly, it is nowadays a rich crossing point of Engineering and Mathematics [1].

Control Theory has come to play such an important role in modern life primarily because it induces the desired behavior in systems. Well developed control systems offer significant performance gains and improvements in efficiency which have prompted more and more industries to adopt control practices. As control becomes ubiquitous it is crucial that controllers are designed and deployed in an efficient manner. Currently designing an efficient control system in many applications requires an in-depth knowledge of the dynamics of the system being controlled. Understanding, modeling and validating the dynamic model of complex real world systems is a non trivial task, consuming vital human design time. Due to this, control design is a significant resource sink. Circumventing this design effort leads to substandard control design causing efficiency losses and may even pose safety issues.

In such a setting the idea of an adaptive controller is appealing. An adaptive controller is a controller that can modify its behavior in response to changes in the dynamics of the process and the character of the disturbances, by adjusting the controller parameters [2]. The field was bubbling with plenty of research – dynamic programming, dual control, learning and adaptation, certainty equivalence principle etc – and ample funding during $1950s$ and $1960s$. However, this drastically ended with the crash of a rocket powered using a self-oscillating adaptive controller [2]. The field resurged in $1970s$ with the developments in the field of system identification, among others.

The introduction of state-space representation of systems by Rudolf Kalman in 1960, not only made the analysis of Multi Input Multi Output (MIMO) systems easier but also laid the foundation for state-space model based controllers [3]. Formulation of such controllers pushed the control community to address issues in the then non-classical fields like process control, in which obtaining a model of the plant with physical modeling of the system was not easy. System identification developed in order to address this need. From the early days, System Identification branched into Prediction Error Methods (PEM) and Subspace IDentification (SID) methods. Detailed history of the development of the field is in [4].

PEM methods, as the name suggests, have the error between the estimated model and the "true" model as the identification criteria. One way to quantify this error is in terms of the output of the estimated model and the output measured from the plant. This branch was mainly developed by Ljung in [5], who changed the outlook towards the field by viewing identification as a design problem, in the sense that search for the "closest" model instead of the "true" model was performed. However, the main drawback of PEM is that they are not easily extendable to MIMO systems, to which SID methods provided a solution. Detailed developments of both the fields was reviewed in the literature survey [4]. Conclusions drawn from the survey are:

In the literature survey, after probing into the recent developments of the field, nuclear norm based subspace identification techniques were seen to be developing rapidly. Nuclear norm was proven to be a heuristic for rank minimization problems and the guarantee of obtaining low rank solutions was also shown. It was concluded in the literature survey that nuclear norm based subspace identification methods will be used for performing system identification. From further survey, the computational burden associated with this heuristic was revealed. The solution of this problem – reduction of the computational burden – is recognized as the direction for development.

1-1 Problem Statement

The combination of identification and control is attractive in the design and implementation of an adaptive controller, most of which run in an online fashion. Performing system identification as a part of design in adaptive controller is attractive primarily because, through identification, the system dynamics are accounted for in the computation of the control signal. This implies that the adaptive control algorithm must allow for real-time implementation and hence, must be efficient and fast. However, as seen in the literature survey, nuclear norm based identification techniques come with a heavy computational burden, which makes them unsuitable for implementation in a fast real-time environment. Thus, the thesis aims to:

Formulate a recursive version of the nuclear norm based subspace technique and combine it with a control law formulation using a MPC strategy.

To carry out the above defined task, it is broken down into the following smaller goals.

1. Reduce the computational time of the considered nuclear norm based subspace identification algorithm.
2. Devise a recursive subspace identification technique by tracking the underlying subspace that describes the system.
3. Validate the technique with the chosen identification database repository.
4. Design a control law with the subspace available from identification.
5. Combine identification and control in one step so that an inherently adaptive controller is in play at every time instant.
6. Test the entire methodology on suitable systems.

1-2 Structure of the thesis

Chapter 2 starts with an overview of N2SID, which is the system identification methodology employed in the thesis. Different modifications that can be made in order to reduce its computational time are suggested and their results are analyzed. A method to perform recursive identification is developed.

Chapter 3 introduces the framework of MPC and presents how the control law can be formulated using the information available from system identification.

The entire methodology is summarized into an algorithm and tested on various systems. The corresponding results are presented in Chapter 4. The document concludes in Chapter 5 by summarizing the current achievements, stating existing problems and accordingly setting the future goals.

Chapter 2

Recursive N2SID

As reviewed in Chapter 1, the main drawback of using Prediction-Error methods is that, unlike SISO systems, MIMO systems cannot be identified in a straightforward manner using these methods. Subspace identification methods mainly emerged as a solution to this drawback. Another drawback of PEM is that the minimization criterion in most of the prediction error methods is non-convex, because of which a global solution is not guaranteed. A detailed comparison between these two classes of system identification can be found in [4]. Most subspace identification methods consist of the following steps [5]:

1 Regression or projection: Estimate high-order model

2 Model reduction: Reduce the high order model to a lower dimensional subspace

3 Parameter estimation: Estimate the system matrices

MOESP [6] and N4SID [7] are examples of algorithms that employ the above defined steps. They require an estimate of the model order for parameter estimation. The model order can be estimated by using an estimation criterion, for example Akaike Information Criterion (AIC), or by a manual examination of the number of dominant singular values of the estimated subspace. The latter can be automated by combining the identification with rank minimization of the underlying subspace i.e., by combining the first two steps – Regression/projection and Model reduction. In the process, the order of the system gets computed and utilized accordingly. The obtained order is the result of a trade-off between model complexity (implying high order) and accuracy of identification. This can be mathematically included in the identification problem statement by the use of a rank minimization term. However, in general the rank minimization problem is known to be computationally intractable (NP - hard) [8].

This led to the development of heuristic techniques that solve the rank minimization problem approximately but efficiently. It was proved in [9] that the nuclear norm can be used as a heuristic for the rank of a matrix, hence translating a rank minimization problem (which is based on the l_0 norm), to a nuclear norm minimization problem (which uses the l_1 norm).

The nuclear norm of a matrix, also called as Ky Fan or trace norm, is defined as the sum of its singular values. Since this is the l_1 norm, translating the problem from rank minimization to nuclear norm minimization eliminates the use of the l_0 norm and relaxes it to the higher order l_1 norm. Another feature of the nuclear norm is that it forms a convex envelope on the rank function; hence the nuclear norm minimization problem is a convex optimization problem. These desirable properties of the nuclear norm led to the rise of nuclear norm based subspace identification techniques, for example [10] and [11].

One such method, [11], which is employed in this thesis is presented in Section 2-1. However, as we will see in Section 2-1-2, nuclear norm based minimization methods require high computational time. In order to reduce the computational time, some modifications are looked into in Section 2-2. For an identification method to be employed in adaptive control algorithms, it needs to perform system identification recursively. Therefore it is necessary to have a recursive nuclear norm based identification method which requires less computational time. Section 2-3 presents a method to use the N2SID algorithm in a recursive manner. All the modifications are evaluated in a combined manner and the corresponding results are presented in Section 2-4.

2-1 Overview of N2SID

As described in [9], the use of nuclear norm for rank minimization is a heuristic method for reducing the order of the model of the system which is to be identified. Reduction of the rank or model complexity is a desirable feature as it is easier to handle low order models. But if the reduced order is significantly smaller than the actual system order, then the estimated model will deviate too much from the actual plant. A trade-off between the extent of rank minimization and accuracy of the model can be obtained by adding a regularization term to the minimization criterion. In other words, regularization retains the parameters of the system that are significant (analogous to dominant singular values) while considering accuracy [12]. Employing nuclear norm in rank minimization is an l_1 regularization. A brief study on different types of regularization can be found in [4].

Generally, in most of the nuclear norm based subspace identification techniques, the rank minimization term is included in the optimization criterion as given in Eq. (2-1.0.1). $F(\hat{y})$ is a function whose rank has to be minimized while keeping $\hat{y}$ (estimated value of the output) close to y (measured value of the output) based on the regularization parameter λ.

$$\min_{\hat{y}} \quad ||F(\hat{y})||_* + \lambda||\hat{y} - y||_2^2 \tag{2-1.0.1}$$

Different choices of $F(\hat{y})$ describe different subspaces pertaining to the underlying system, whose dimensionality will be minimized. In this section, we discuss the N2SID algorithm [11] in detail, and in the process, the description of $F(\hat{y})$ as used in N2SID.

The mathematical model of the system to be identified is first expressed in innovation form as:

$$\begin{aligned} x(k+1) &= A_{sys}x(k) + B_{sys}u(k) + Ke(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned} \tag{2-1.0.2}$$

where $u(k) \in \mathbb{R}^{m_u}, x(k) \in \mathbb{R}^n, y(k) \in \mathbb{R}^{p_y}$ and $e(k)$ is zero mean white noise. Here the state $x(k)$, the system matrices $A_{sys}, B_{sys}, C_{sys}, D_{sys}$ and the Kalman gain K are unknown. System from Eq. (2-1.0.2) is then expressed in observer form as:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ky(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned} \tag{2-1.0.3}$$

where $A = (A_{sys} - KC_{sys})$, $B = (B_{sys} - KD_{sys})$, $C = C_{sys}$ and $D = D_{sys}$.

Measured input-output data from the system is stored in block Hankel matrices. The two parameters defining the size of the Hankel matrix are N (the number of measurements) and s (defining the number of block rows) where $s > n$, $N >> n$ given that n is the order of the system. The hankel matrix of the input $u(k)$ is defined as in Eq. (2-1.0.4). Since $u(k) \in \mathbb{R}^{m_u}$, $U_s \in \mathbb{R}^{m \times q}$, where $m = sm_u$ and $q = (N - s + 1)$. Similarly the Hankel matrix, Y_s, for the output is defined, with $Y_s \in \mathbb{R}^{p \times q}$ where $p = s * p_y$.

$$U_s = \begin{bmatrix} u(1) & u(2) & \dots & u(N-s+1) \\ u(2) & u(3) & & \vdots \\ \vdots & & \ddots & \\ u(s) & u(s+1) & \dots & u(N) \end{bmatrix} \tag{2-1.0.4}$$

The equation for $y(k)$ in the observer form, Eq. (2-1.0.3), can be extended to include N measurement values at once using the Hankel matrix definition of the input and output. Such an extended form, known as the data equation, can be expressed as given in Eq. (2-1.0.5). Matrices $T_{u,s} \in \mathbb{R}^{p \times m}$ and $T_{y,s} \in \mathbb{R}^{p \times p}$ contain the system matrices and are expressed in Toeplitz form. Matrix $O_s \in \mathbb{R}^{p \times n}$ is known as the extended observability matrix.

$$Y_s = O_s X + T_{u,s} U_s + T_{y,s} Y_s + E_s \tag{2-1.0.5}$$

where

$$T_{u,s} = \begin{bmatrix} D & 0 & \dots & 0 \\ CB & D & & \\ CAB & CB & \ddots & \vdots \\ \vdots & & & \\ CA^{s-2}B & \dots & CB & D \end{bmatrix} \quad T_{y,s} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CK & 0 & & \\ CAK & CK & \ddots & \vdots \\ \vdots & & & \\ CA^{s-2}K & \dots & CK & 0 \end{bmatrix}$$

$$O_s = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix} \qquad X = \begin{bmatrix} x(1) & x(2) & \dots & x(N-s+1) \end{bmatrix}$$

Introducing the estimated value of the output as $\hat{y}(k) = y(k) - e(k)$, and constructing the Hankel matrix $\hat{Y}_s$ from $\hat{y}$, Eq. (2-1.0.5) is expressed as,

$$\hat{Y}_s = O_s X + T_{u,s} U_s + T_{y,s} Y_s \tag{2-1.0.6}$$

Since the term $O_s X$ is a subset of the range space, it defines a subspace of the system to be identified, [13]. The system can therefore be identified by minimizing the rank of $O_s X$. Hence, the function $F(\hat{y})$, from Eq. (2-1.0.1) is $O_s X$ which is equal to $\hat{Y}_s - T_{u,s} U_s - T_{y,s} Y_s$. Including the rank minimization term and system output tracking, the optimization problem is formulated as:

$$\min_{\hat{Y}_s, T_{u,s}, T_{y,s}} \quad \frac{\lambda}{N} \sum_{k=1}^{N} ||y(k) - \hat{y}(k)||_2^2 + ||\hat{Y}_s - T_{u,s} U_s - T_{y,s} Y_s||_\star \tag{2-1.0.7}$$

Since both the terms in Eq. (2-1.0.7) are convex functions (the first being quadratic and the second term being the nuclear norm, which is a convex envelope [9]), the identification problem is a convex optimization problem.

There has been tremendous development in the field of convex optimization, giving rise to numerous algorithms for solving it. In N2SID [11], and in this thesis, the algorithm used to solve the convex optimization problem is ADMM; see [4] for details on other algorithms. Having formulated the N2SID problem, which is a convex optimization problem, we now look into the details of solving it using ADMM. The resulting optimization variables are denoted as $\hat{Y}_s^e, T_{u,s}^e, T_{y,s}^e$.

2-1-1 Solving the optimization problem using ADMM

The convex optimization problem given in Eq. (2-1.0.7), is formulated as a combination of nuclear norm and summation of errors. It can be solved using readily available SDP solvers, as was done in [14]. However, the computation time required by these solvers is high. On the other hand, ADMM was formulated to solve large scale problems. Although the convergence rate is poor in ADMM, [15], it results in modest accuracy within a few iterations. This fact is further looked into in Section 2-1-2 and is used to speed up the overall recursive identification. Another feature of ADMM is that it does not require the optimization function to be differentiable; sub-differentials of the function can be readily used. This is a major advantage in our case since the nuclear norm operator is not differentiable with respect to its arguments. The solution of the optimization problem using ADMM is considered in detail in this section since the theory to be developed in Section 2-3 is closely linked with the working of ADMM.

The standard form of the minimization problem solved using ADMM [10], is given in Eq. (2-1.0.8).

$$\begin{aligned} \min_{\mathbf{x}, \mathrm{X}} \quad & f(\mathbf{x}) + g(\mathrm{X}) \\ \text{subject to} \quad & \mathcal{A}(\mathbf{x}) - \mathrm{X} = \mathcal{B} \end{aligned} \tag{2-1.0.8}$$

The N2SID minimization problem can be expressed in the standard form by considering $f(\mathbf{x})$ to be the quadratic term in Eq. (2-1.0.7) and $g(\mathrm{X})$ to be the nuclear norm. The quadratic term, which is the summation of errors between measured output and estimated output of the system, can be expressed in the vector form as in Eq. (2-1.0.9). In the translation, $\mathbf{x} \in \mathbb{R}^n$ contains the estimated output $\hat{y}$, a is the measured output y and $\mathcal{C}$ contains the regularization parameter λ.

$$f(\mathbf{x}) = (1/2)(\mathbf{x} - a)^T \mathcal{C}(\mathbf{x} - a) \tag{2-1.0.9}$$

The second function $g(\mathrm{X})$ is the nuclear norm term $||\mathrm{X}||_*$ where

$$\mathrm{X} = \hat{Y}_s - T_{u,s}U_s - T_{y,s}Y_s$$

$\mathrm{X} \in \mathbb{R}^{p \times q}$ thus contains the optimization variables $\hat{Y}_s, T_{u,s}, T_{y,s}$. In the constraint, the linear mapping $\mathcal{A}$ is such that $\mathcal{A} : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}^{p \times q}$, where $n_\mathbf{x} = p_y(N + m + (s-1)p_y)$.

Since ADMM, which is a primal-dual algorithm [16], is a blend of dual ascent and method of multipliers, it inherits the decomposability of dual ascent [15]. This property allows for the separation of optimization variables into $\mathbf{x}$ and X along with the splitting of the objective function into two functions $f(\mathbf{x})$ and $g(\mathrm{X})$. The variables $\mathbf{x}$ and X are primal variables. Since ADMM is a primal-dual algorithm, it maximizes a dual function, which can be constructed using the Augmented Lagrangian L_{t_p} which is defined as:

$$L_{t_p} = f(\mathbf{x}) + g(\mathrm{X}) + \mathrm{trace}(Z^T(\mathcal{A}(\mathbf{x}) - \mathrm{X} - \mathcal{B})) + \frac{t_p}{2}||\mathcal{A}(\mathbf{x}) - \mathrm{X} - \mathcal{B}||_F^2 \tag{2-1.0.10}$$

where Z is the Lagrange dual variable and t_p is the penalty parameter. The dual function is given by:

$$g_{t_p}(Z) = \inf_{\mathbf{x},X} \; L_{t_p}(\mathbf{x}, X, Z) \tag{2-1.0.11}$$

The dual function is concave [17] and for any Z, we have,

$$g_{t_p}(Z) \quad \leq \quad p^*$$

where p^* , which is the optimal value of the problem in Eq. (2-1.0.8), is given by

$$p^* = \inf(f(\mathbf{x}) + g(X) \;\; | \;\; \mathcal{A}(\mathbf{x}) - X = \mathcal{B})$$

As the dual function yields lower bounds on the optimal value p^*, our aim is to find the greatest lower bound. Hence the problem to be solved now is:

$$\max_Z g_{t_p}(Z) \tag{2-1.0.12}$$

It can be noted that maximization of a concave function is equivalent to minimization of a convex function. The problem Eq. (2-1.0.12) is solved by the dual ascent method which is a gradient ascent technique. ADMM solves the problem by maximizing the dual function and finding the minimum of the primal variables for each dual update Z. ADMM can now be summarized as given in Algorithm 1, [10].

Algorithm 1 ADMM

`Initialize` $\mathbf{x} = 0, X = -\mathcal{B}, Z = 0, t_p = 1$
while stopping criteria is `false` **do**
`Update x as` $\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\texttt{argmin}} L_{t_p}(\mathbf{x}, X^k, Z^k)$
`Update` X `as` $X^{k+1} = \underset{X}{\texttt{argmin}} L_{t_p}(\mathbf{x}^{k+1}, X, Z^k)$
`Update` Z `as` $Z^{k+1} = Z^k + t_p(\mathcal{A}(\mathbf{x}^{k+1}) - X^{k+1} - \mathcal{B})$
`Update primal and dual residuals`
`Check for stopping criteria`

As the number of iterations $k \to \infty$ in the ADMM algorithm, we observe the following trends [15],

1 Residual convergence: Primal and Dual residuals tend to zero.

2 Objective convergence: $f(\mathbf{x}^k) + g(X^k)$ approaches the optimal value p^*

3 Dual variable convergence: Z^k approaches the dual optimal point Z^*

It is worth noting that the convergence of the primal variables to their respective optimal points is not guaranteed. This is because ADMM inherently uses dual ascent method and hence maximizes the concave dual function.

In this section, the theory and working of ADMM were studied since it is employed to solve the N2SID problem formulated in Eq. (2-1.0.7). The convergence behavior of ADMM in terms of the primal and dual residuals is analyzed in the next section, by performing system identification of a CD-player arm system using N2SID and solving the optimization problem using ADMM.

2-1-2 Convergence analysis of ADMM

In this section, the results of the N2SID algorithm, implemented as given in [11], are presented. System identification is carried out for a CD-player arm system whose dataset is obtained from [18]. Identification is performed using $N = 200$ measurements for a particular value of regularization parameter $\lambda = 4.28$. For a detailed analysis on varying N and λ, see [11]. These standard results, given in Table 2-1, will be used for direct comparison with the results obtained in this thesis.

It can be seen from the results that the system is estimated with sufficient VAF but the time taken is considerably high. From a comparison of N2SID with N4SID and PEM, it was seen in [11] that N2SID resulted in higher VAF than the rest for $N \leq 400$, hence outperforming the

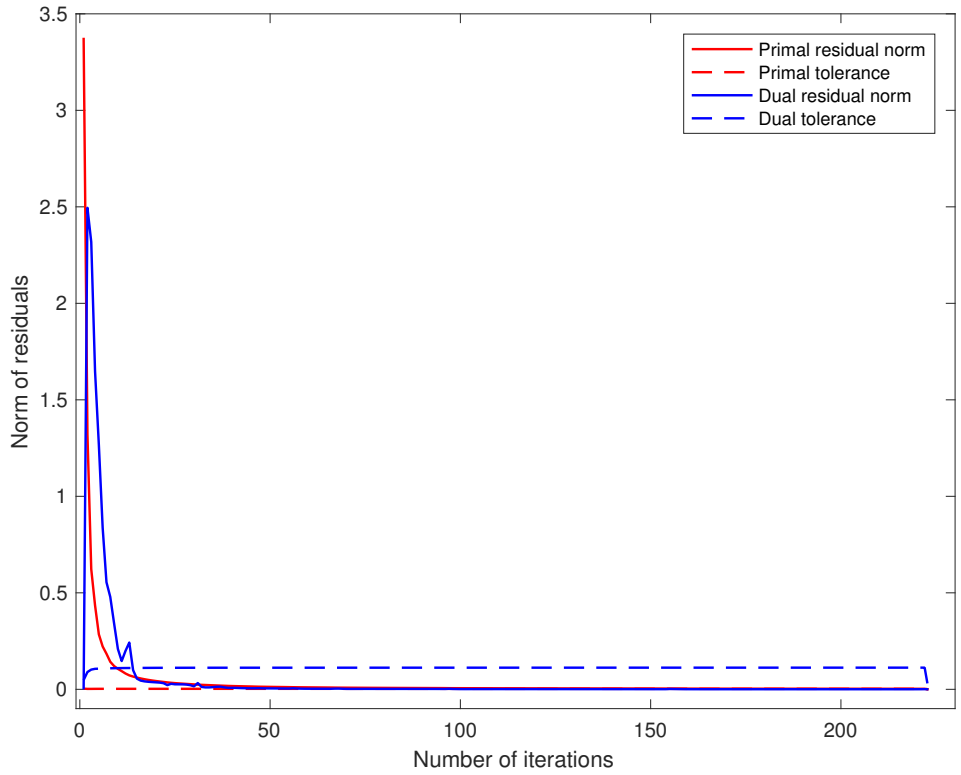


Figure 2-1: Primal and Dual residuals

other methods. However, to perform system identification on the CD-player arm system once takes almost 3 seconds on a standard desktop computer running Apple OS X and equipped with a 2.2GHz processor and 8GB RAM. Hence, the system, with sampling time lesser than 3 seconds, cannot be identified at every sampling time instant. This is the primary motivation to reduce the computational time taken to identify the system using N2SID. In order to achieve this reduction in computation time, we first analyze the convergence behavior of the ADMM algorithm which plays a vital role in the N2SID algorithm.

Table 2-1: Results of identification for a CD-player arm system

System Order	8
VAF	84.36
Total number of iterations	223
Total time taken	2.98 sec

The values of primal and dual residuals obtained in each iteration are shown in Figure 2-1 along with their respective tolerances. The iterations of ADMM stop once both the primal and dual residuals are lesser than their tolerances.

We can observe that the residuals tend to 0 with increasing number of iterations, in accordance

with the theoretical results. However, it can also be noticed that the rate of convergence is very slow after a few iterations. This is a typical characteristic of ADMM: slow convergence to obtain high accuracy [15].

In Section 2-2-1 we will see how this behavior can be exploited to achieve faster computation of N2SID. Along with this, other sections of Section 2-2 explore parts of ADMM which can be modified to obtain a further speedup of the algorithm.

2-2 Improving the speed of N2SID

During the review of the N2SID algorithm, it was seen in Section 2-1-2 that the computational time of N2SID is too high for it to be recursively implemented. This is a major drawback since it cannot be employed to perform system identification at each sampling time instant of the system, rendering it inapplicable when used as a part of an online adaptive control framework. In this section, we address this problem by improving the speed of the overall algorithm.

Since ADMM is used to solve the formulated N2SID problem, most of the computational burden of N2SID lies within ADMM. Therefore, we focus our attention mainly on the ADMM algorithm. In Section 2-2-1, we exploit the convergence behavior of ADMM to reduce the computational time. Furthermore, in Section 2-2-2, the step of performing SVT in ADMM is investigated and different alternatives that require lesser computation are looked into.

2-2-1 Early stopping of ADMM

In Section 2-1-2, it was suggested that the convergence behavior of ADMM can be exploited to potentially speed up the algorithm. It was seen that the ADMM algorithm is characterized by very slow convergence rates and very high accuracy. Therefore, if the application does not require extremely high accuracy, we can terminate the ADMM algorithm after a few iterations. For the same CD player system, the ADMM algorithm is terminated at 30 iterations. The corresponding results are compared with the results of N2SID when ADMM was not terminated early. It is noticed that while terminating ADMM early leads to a negligible reduction in VAF, the time taken is reduced by 36.57%.

Table 2-2: Early stopping of ADMM

	Time taken in sec	VAF
Baseline N2SID	2.98	84.36
Early stop ADMM	1.89	83.44

To summarize, the time taken to perform system identification of a CD-player arm system using N2SID, with ADMM terminated early, is 1.89 seconds. As new measurements arrive, the system identification process must be repeated at each sampling time. This implies that successive iterations of N2SID must be completed within the sampling period of the system. N2SID, as is now, can be used to recursively identify a slow system (system with slow sampling

times). In order to maximize the applicability of this recursive approach, we try to further reduce the computational time of the N2SID algorithm.

2-2-2 Alternative methods of Singular Value Thresholding

The minimization of the nuclear norm represents a major computational bottleneck in the N2SID algorithm. As seen in Section 2-1-1, each iteration of ADMM involves the minimization of the Augmented Lagrangian L_{t_p}, over the primal variables $\mathbf{x}$ and X and an update of the dual variable Z. Due to the decomposable nature of ADMM, the objective function was split into $f(\mathbf{x})$ and $g(\mathrm{X})$, hence allowing the primal variables to be contained in different functions. As a result, the minimization of the nuclear norm term $g(\mathrm{X})$ is reflected only in the minimizer X, i.e., the minimization of L_{t_p} over X, whose update formula for k^{th} iteration is given in Eq. (2-2.0.1). The minimizer X is obtained from the singular value thresholding operation D_τ, also known as soft thresholding, of the matrix $\mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p$.

$$\mathrm{X}^k = D_\tau(\mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p) \tag{2-2.0.1}$$

As the name suggests, soft thresholding is imposing a soft threshold, τ, on the singular values of the matrix whose nuclear norm has to be minimized. This is performed by first obtaining the SVD of the matrix as:

$$U\Sigma V^T = \mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p$$

The singular values, contained in Σ, are then shrunk by the predefined threshold τ as:

$$D_\tau(\Sigma) = diag(\{\sigma_i - \tau\}_+)$$

Then the soft thresholding of the matrix is obtained as:

$$D_\tau(\mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p) = UD_\tau(\Sigma)V^T$$

In other words, each singular value that is greater than τ is considered and the corresponding difference $(\sigma_i - \tau)$ is utilized. This effectively shrinks the singular values towards zero and hence reduces the nuclear norm. In our case, the soft threshold is $\tau = 1/t_p$. Since this operation involves SVD, it is computationally heavy and hence slows down the identification process.

To speed up the N2SID algorithm, either an alternative method to the computation of SVD or an alternative method to directly compute soft thresholding without SVD can be used. In this section we take up three different approaches to compute the soft thresholding along with their advantages and disadvantages with respect to their applicability to our problem statement. Accordingly, a suitable method is chosen as the alternative and employed to solve for the minimizer X in ADMM.

From the survey on literature of alternatives to SVD, [4], two of the methods that are considered in the thesis are from the papers [19] and [20]. As there are no particular names

specified for these two methods, we call the first method as "QR-updating with Jacobi-type SVD", and the second method as "Fast SVT without SVD", which is the title of the paper. The first method provides an alternative way of computing SVD whereas the second method directly computes the soft thresholding without SVD. Due to the problems associated with this method, which will be dealt soon, we later resort to the third method - Fast randomized SVT.

For quick demonstrations, we consider the soft thresholding of a test matrix Y with the soft threshold τ. At places where the results for soft thresholding of the matrix $\mathcal{A}(\mathbf{x}^k)-\mathcal{B}+Z^{k-1}/t_p$ are presented, it will be accordingly specified.

QR-updating with Jacobi-type SVD

In order to perform SVT in a computationally favorable manner while tracking the subspace, QR-updating with Jacobi-type SVD method, introduced in [19], is considered here. This method computes the SVD of a matrix, which can then be used to compute the soft thresholding of the matrix. Although this method does not directly result in the soft threshold of a matrix, it is helpful, as we will see, in tracking the subspace when the input matrix Y contains the system measurements. It focuses on updating the SVD of Y after appending a new row of measurements y to it, without explicitly computing the SVD again.

Consider an initial matrix $Y_i \in \mathbb{R}^{i\times n}$, where $n = 4$ and i is the sampling time instant. With each new measurement, i increases and a new set of measurements y_i are appended to Y_i, i.e.,

$$Y_i = \begin{bmatrix} Y_{i-1} \\ y_i \end{bmatrix}$$

The method updates the SVD of Y_i using the SVD of Y_{i-1} instead of explicitly computing the SVD with each new measurement. This method is compared with the conventional SVD, for random measurements for 500 updates of measurements. Note that the size of Y is growing with the increase in number of measurements. Figure 2-2 shows the singular values of Y_i for 500 updates, obtained from conventional SVD and the considered method QR-updating with Jacobi-type SVD. It can be seen that the results of the alternative method are fairly close to those obtained from SVD. An important and attractive result is that the time taken for computation of SVD for 500 updates is 0.3425 seconds, whereas time taken by the alternative method was 0.2641 seconds, which represents a 22.89% reduction in computation time.

While this method is computationally favorable and rectangular matrix friendly, it is useful when new measurements are appended with each update. In our case, in the i^{th} identification cycle, which uses the set of measurements from $(i - N + 1)$ to i, the ADMM iterates k times. In other words, for each identification cycle, the measurements are the same over the iterations of ADMM. The computation of the minimizer requires computing the SVD and furthermore, X^k does not involve new measurements with each iteration. Hence, this method is not applicable to our case. Therefore, we move on to evaluating the usefulness of the other alternative methods.

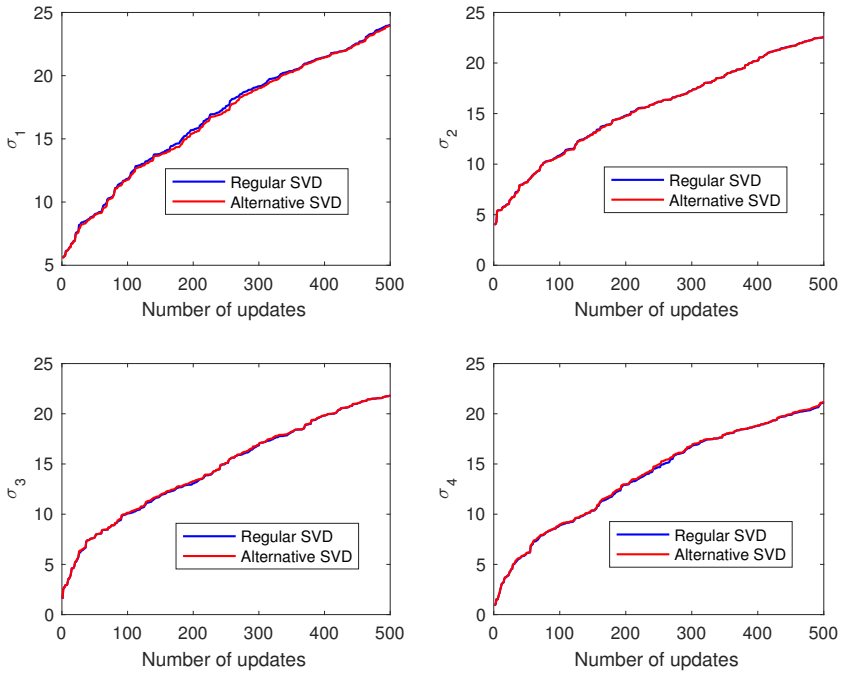


Figure 2-2: Comparison of singular values computed from SVD and QR-updating with Jacobi-type SVD

Fast SVT without SVD

This method, Fast Singular Value Thresholding without SVD, [20], directly computes the soft thresholding of a matrix without explicitly making use of SVD. It does so by first computing the Polar Decomposition of the input matrix $Y \in \mathbb{R}^{p\times q}$ as:

$$Y = W_{pd}Z_{pd}, \quad W_{pd} \in \mathbb{R}^{p\times q} \text{ and } Z_{pd} \in \mathbb{R}^{q\times q}$$

W_{pd}, an orthogonal matrix, is the "signum" of Y and Z_{pd}, a positive semi-definite matrix, is the "absolute value" of Y containing information about the singular values of Y. W_{pd} is initialized with Y and then iteratively computed. Z_{pd} is computed as $W_{pd}^T Y$ and then subsequently used to compute the soft threshold of Y. The detailed steps are given in Algorithm 2.

Algorithm 2 Fast SVT without SVD

```
Input matrix: Y
      Compute the polar decomposition Y = WZ
      Compute the projection P_τ(Z) = argmin_{||X||_2≤τ} ||Z − X||_F
      Set D_τ(Y) = Y − W P_τ(Z)
Output matrix:  SVT D_τ(Y)
```

This method is compared with the method of computing soft thresholding using SVD, by performing system identification of the CD-player arm system for different values of regularization parameter λ, as in Eq. (2-1.0.7). VAF obtained and corresponding model orders,

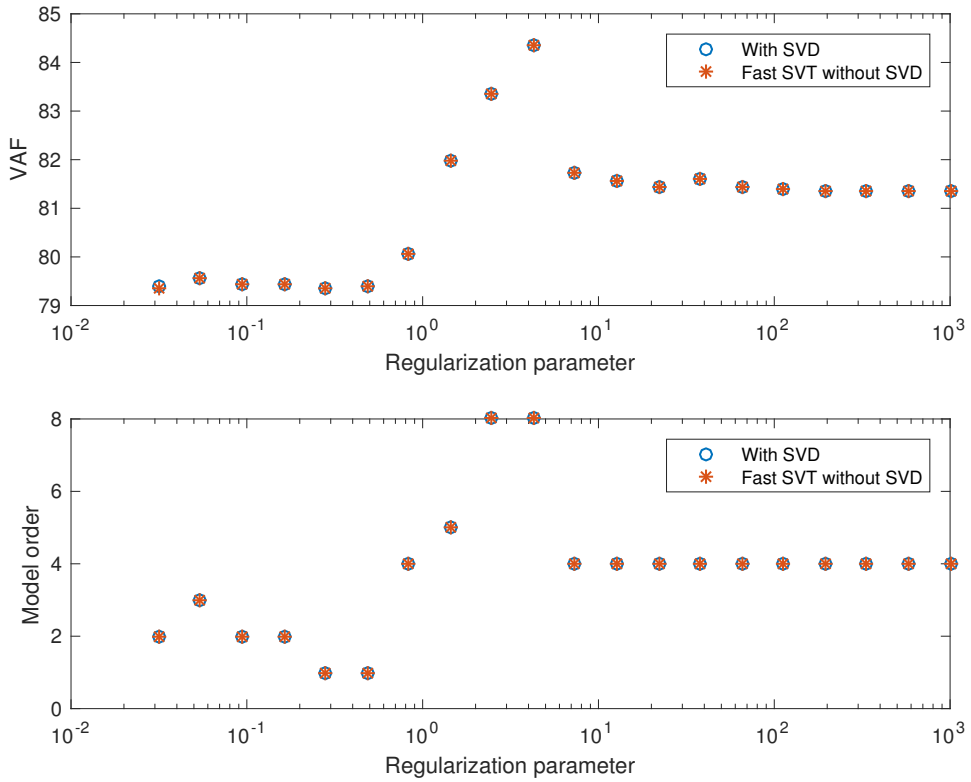


Figure 2-3: Comparison in terms of resulting VAF and model order

for different values of λ for both the methods are shown in Figure 2-3. The total time and iterations taken by the two methods are given in Table 2-3. It is noticed that the alternative method results in VAF and model orders that are comparable with the case when SVD is used for computation. However, the time taken by the alternative method is significantly more than the method using SVD.

The main reason for the increased computational time is that the matrix Y is not square, because of which the Polar Decomposition of Y cannot be computed directly. Complete Orthogonal Decomposition (COD), which can be found in [21], of Y is first computed as:

$$Y = O_{cod} \begin{bmatrix} R_{cod} & 0 \\ 0 & 0 \end{bmatrix} Q_{cod}^T$$

Algorithm 2 is then carried out for R_{cod} with the corresponding translations performed in the end to trace back to Y. Since in our case $Y = \mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p$ whose soft thresholding has to be computed is not square, computation of COD is significantly increasing the computational burden.

Table 2-3: Comparison of total time and number of iterations

	Total time taken	Total number of iterations
With SVD	15.37	1442
Without SVD	117.55	1443

Another disadvantage of this method is its inability to track a subspace. For computing the minimizer X^k in the k^{th} iteration of ADMM, information from X^{k-1} could be used since there exists a relationship between X^k and X^{k-1}. However, this method does not provide a framework in which we can utilize this information. To demonstrate this, we consider the algorithm in a little more detail.

In the Polar Decomposition step in Algorithm 2, it is Z_{pd}^k that contains the information about singular values of Y^k. Therefore, instead of initializing W_{pd}^k with Y^k and then computing Z_{pd}^k, the reverse could be tried, i.e, compute Z_{pd}^k iteratively and then compute W_{pd}^k as $Y^k(Z^k)^{-1}$. The idea behind this strategy is that Z_{pd}^k could be initialized as Z_{pd}^{k-1}, hence using the information from the soft thresholding computed in the previous iteration. However, due to the semi-definite nature of Z_{pd}^k it is not invertible and thus this method cannot be used to track the subspace.

As this method is neither favorably fast nor possesses the ability to track a subspace, the next alternative to SVD is looked into.

FRSVT

Fast Randomized Singular Value Thresholding (FRSVT) is introduced in [22] primarily for computing soft thresholding in nuclear norm minimization. As the name suggests, the method directly computes the singular value thresholding of a matrix, without using SVD. The special feature of this method is that it exploits the proximity of the range space over iterations (in our case, the relationship between X^k and X^{k-1} in an identification cycle) and propagates the estimated basis of the previous iteration to the next iteration thus accelerating the algorithm.

The method is summarized in Algorithm 3. Note that since the method does not compute Polar Decomposition directly on the matrix Y, there is no need to compute Complete Orthogonal Decomposition (COD) of Y if it is not square. This renders the method to be rectangular matrix friendly, which is a major advantage in our case.

The input matrix Y is randomized by multiplying it with a Gaussian random matrix Ω. The randomized matrix A is used to capture the range space of Y in Q and to reduce the dimensions of the SVT problem. Both are achieved by computing, on the randomized matrix A, the QR decomposition with column pivoting (QR-CP), which estimates the rank and dominant bases of Y through the randomized matrix A. Since this estimation of the orthonormal column matrix Q is the only approximation step in Algorithm 3, the accuracy of the method depends on this step. In order to better capture the range of Y in Q, the FRSVT method uses power iterations: repeating the last bit of the algorithm for η times.

It should be noted that the FRSVT algorithm uses only QR, Polar and Eigen decomposition and at no place uses SVD. FRSVT method is compared with the method of singular value

Algorithm 3 FRSVT

```
Input matrix: Y ∈ ℝ^{m×n} and the orthogonal column matrix Q̃ of the previous iteration
    if not Range propagation then
      Sample Gaussian random matrix Ω ∈ ℝ^{n×l}
       A = YΩ
       Q = QRCP (A)
    else
      Sample Gaussian random matrix Ω ∈ ℝ^{n×p}
       A = YΩ
       Q_Y = PartialOrthogonalization(Q̃, A)
       Q = [Q̃   Q_Y]
    end if
    repeat
      Q = QR (YY^T Q)
    until η times
      [H, C] = QR (Y^T Q)
      [W, P] = PolarDecomposition(C)
      [V, D] = EigenDecomposition(P)
      D_τ(Y) = (QV)D_τ(D)(HWV)^T
Output: SVT D_τ(Y), Q̃ = QV
```

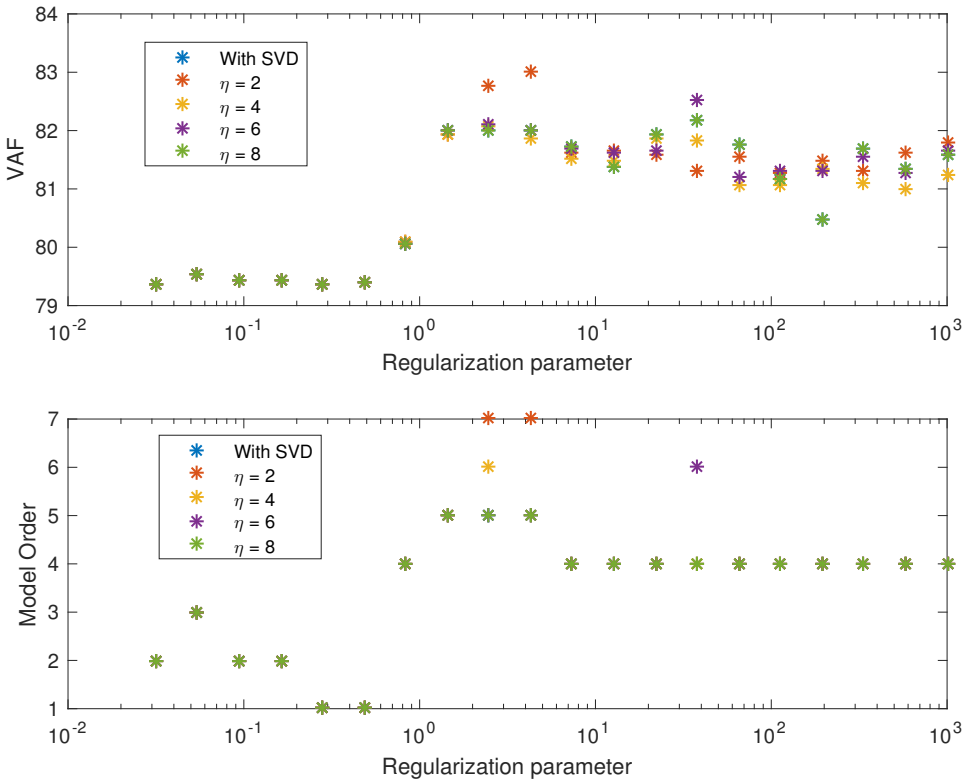


Figure 2-4: Evaluation of FRSVT terms of resulting VAF and model order

thresholding using SVD, by performing one identification cycle of a CD-player arm system for different values of regularization parameter λ, see Figure 2-4. The evaluation is done for a range of values of λ in order to check if FRSVT adversely affects the performance of the host algorithm (ADMM in our case). It can be recollected that similar evaluation was carried out for analyzing the previously considered method - Fast SVT without SVD. Identification is performed for different values of η. It is noticed that increasing the number of power iterations (η), does not have significant effect on the performance of identification. Like mentioned in [22], $\eta = 2$ is enough and will be used in this thesis for further implementations.

For performing system identification of the CD-player arm system for a range of values of λ, the time taken by this method (FRSVT) is compared with the time taken by the other two methods: SVT with SVD and Fast SVT without SVD. The time taken by each method is listed in Table 2-4. Although using SVD for obtaining SVT is still computationally faster, this method has the theoretical advantage of propagating the range space, of the matrix whose soft threshold has to be computed, over ADMM iterations.

Table 2-4: Evaluation of SVT methods

SVT method	Time taken in sec
SVT with SVD	15.37
Fast SVT without SVD	117.55
FRSVT	33.43

Since FRSVT offers range propagation over the iterations and since the time taken by FRSVT is comparable to that of computing SVT using SVD, we will be employing FRSVT for the computation of singular value thresholding to obtain the minimizer X^k in the iterations of ADMM in N2SID. For more analysis, further comparison of FRSVT and SVT using SVD is carried out, in Chapter 4, for different systems which are being recursively identified and controlled at each sampling time instant.

In this section, different parts of the N2SID algorithm were analyzed for potential improvements in the computational time. It was seen in Section 2-2-1 that terminating the ADMM algorithm early significantly reduced the computational time while not adversely affecting the VAF of the identification. It was also noted that performing singular value thresholding to obtain the minimizer X^k in ADMM was increasing the computational burden of the overall algorithm, because of which different alternatives to perform the thresholding were looked into. Out of the three alternative methods - QR-updating with Jacobi-type SVD, Fast SVT without SVD and FRSVT - the first method turned out to be less applicable to our problem. Out of the last two methods, which directly resulted in X^k, the method Fast SVT without SVD was computationally burdensome because of the rectangular size of the matrix $\mathcal{A}(\mathbf{x}^k) - \mathcal{B} + Z^{k-1}/t_p$, whose soft thresholding has to be computed. Evaluation of the last method – FRSVT – revealed its suitability to our application and was finalized to be employed in performing the SVT.

So far, the focus has been on reducing the computational time of N2SID so that each identification cycle is performed faster. Since our final goal is to identify a system recursively at each time instant, we now focus our attention on analyzing the recursion in greater detail. Furthermore, as elaborated in the next section, a method to recursively identify a system in

a fast manner is developed. The validity of this strategy and the conditions under which it will reduce the computational time are also looked into.

2-3 Tracking the underlying subspace

In Section 2-1, the procedure and details of identifying a system using N2SID algorithm was reviewed. This section deals with identifying the system, recursively, at each sampling time instant. We call the system identification performed at the i^{th} time instant as the i^{th} identification cycle. Each identification cycle is performed as described in Section 2-1.

For the i^{th} identification cycle, the N2SID problem, formulated as in Eq. (2-1.0.7), is denoted by $\mathcal{Q}^i$ as:

$$\mathcal{Q}^i : \min_{\hat{Y}_s^i, T_{u,s}^i, T_{y,s}^i} \quad \frac{\lambda}{N} \sum_{k=i-N+1}^{i} ||y^i(k) - \hat{y}^i(k)||_2^2 + ||\hat{Y}_s^i - T_{u,s}^i U_s^i - T_{y,s}^i Y_s^i||_\star \tag{2-3.0.1}$$

The corresponding optimization problem that is solved using ADMM is denoted by $\mathcal{P}^i$, which is defined in Eq. (2-3.0.2) as:

$$\begin{aligned} \mathcal{P}^i : \quad & \min_{\mathbf{x}, X} \quad f^i(\mathbf{x}) + g^i(\mathrm{X}) \\ & \text{subject to} \qquad \mathcal{A}^i(\mathbf{x}) - \mathrm{X} = \mathcal{B} \end{aligned} \tag{2-3.0.2}$$

From Section 2-1-1, we know that ADMM results in the optimal value of $\mathcal{P}^i$, denoted as p^{*i}, and the optimal value of the dual function, denoted as Z^i. Denote the corresponding values of primal variables as $\mathbf{x}^i, \mathrm{X}^i$. Note that the primal variables need not be the optimal values.

Having completed the i^{th} identification cycle, the next step is to perform system identification at the $(i+1)^{th}$ time instant - $\mathcal{Q}^{i+1}$. There are two ways to do this:

1 Repeat all the steps exactly like the last identification cycle. In doing so, the parameters are initialized with their default initial values as given Algorithm 1. This implies that the system, although identified just one time instant before, is now treated like a new system.

2 Utilize the system information obtained from the last identification cycle to initialize parameters in the algorithm. The underlying reason for using past information is the assumption that it helps improve the convergence speed, which is desirable in an online recursive algorithm.

Intuitively, the second approach promises to improve computation time by utilizing information from the previous identification cycle. To this end, Section 2-3-1 outlines a mathematical approach to utilize this information for generating initial conditions of the parameters. Theoretical guarantees on the choice of the initial conditions are developed. The strategy is then implemented and the results are analyzed in Section 2-3-2.

2-3-1 Using information from the last identification cycle

From Section 2-1, since the functions $f^i(\mathbf{x})$ and $g^i(\mathrm{X})$ depend on U_s^i and Y_s^i, the optimization problem $\mathcal{P}^i$, given in (2-3.0.2), directly depends on the system measurements. The optimization problems $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$ differ by one time instant measurement and the difference between the problems is proportional to the difference between the measurement values. To see this clearly, we analyze the difference term by term.

Consider the problem $\mathcal{P}^i$ from Eq. (2-3.0.2) and denote the functions $g^i(\mathrm{X})$ by T_1^i and $f^i(\mathbf{x})$ by T_2^i, i.e.,

$$\begin{aligned} T_1^i &:= ||\mathcal{A}^i(\mathbf{x}) - \mathcal{B}||_\star \\ T_2^i &:= (1/2)(\mathbf{x} - a^i)^T \mathcal{C}(\mathbf{x} - a^i) \end{aligned}$$

The difference between T_2 in $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$ is given by:

$$T_2^i - T_2^{i+1} = (1/2)((a^i)^T \mathcal{C} a^i - (a^{i+1})^T \mathcal{C} a^{i+1} + \mathbf{x}^T \mathcal{C}(a^{i+1} - a^i) + ((a^{i+1})^T - (a^i)^T)\mathcal{C}\mathbf{x})$$

It can be recollected, from Section 2-1-1, that a^i is a column vector consisting of the system output measurements from time instant $(i - N + 1)$ to i. Therefore each entry in the column vector $(a^{i+1} - a^i)$ is the difference between $y(k+1)$ and $y(k)$. Accordingly,

$$(a^i)^T \mathcal{C} a^i - (a^{i+1})^T \mathcal{C} a^{i+1} = \sum_{k=i-N+1}^{i} \lambda_k (y(k)^2 - y(k+1)^2)$$

where λ_k is the regularization parameter for the k^{th} measurement value. Hence we can conclude that $(T_2^i - T_2^{i+1})$ depends on $(y(k)^2 - y(k+1)^2)$ and $(y(k) - y(k+1))$.

Let us now analyze the term T_1. The difference between T_1 in $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$ is given by:

$$T_1^i - T_1^{i+1} = ||\mathcal{A}^i(\mathbf{x}) - \mathcal{B}||_* - ||\mathcal{A}^{i+1}(\mathbf{x}) - \mathcal{B}||_*$$

As the nuclear norm is a valid norm, it satisfies the reverse triangle inequality.

$$| \, ||\mathcal{A}^i(\mathbf{x}) - \mathcal{B}||_* - ||\mathcal{A}^{i+1}(\mathbf{x}) - \mathcal{B}||_* \, | \leq ||\mathcal{A}^i(\mathbf{x}) - \mathcal{A}^{i+1}(\mathbf{x})||_* \tag{2-3.0.3}$$

The linear map $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^{p \times q}$ is defined as given in Eq. (2-3.0.4), [11].

$$\mathcal{A}^i(\mathbf{x}) = \mathcal{H}_{\hat{y}} + \mathcal{H}_v V^i + \mathcal{H}_w W^i \tag{2-3.0.4}$$

where $\mathcal{H}_{\hat{y}}, \mathcal{H}_v, \mathcal{H}_w$ correspond to the optimization variables $\hat{Y}_s, T_{u,s}, T_{y,s}$ in Eq. (2-1.0.7) respectively, and are defined from $\mathbf{x}$. V^i and W^i are the Hankel matrices, as defined in Eq. (2-1.0.4), of input and output measurements, respectively, from time instant $(i - N + 1)$ to i.

Analyzing the right hand side of (2-3.0.3), we have,

$$||\mathcal{A}^i(\mathbf{x}) - \mathcal{A}^{i+1}(\mathbf{x})||_* = ||\mathcal{H}_v(V^i - V^{i+1}) + \mathcal{H}_w(W^i - W^{i+1})||_*$$

From Eq. (2-1.0.4) we have that each entry in $(V^i - V^{i+1})$ is the difference between $u(k)$ and $u(k+1)$. Similarly each term in $(W^i - W^{i+1})$ is the difference between $y(k)$ and $y(k+1)$. Hence, $(T_1^i - T_1^{i+1})$ depends on the difference between consecutive time instant measurements of the system input and output.

We make the assumption that the system to be identified is BIBO stable. Hence, for bounded inputs to the system, the outputs are bounded. From the analysis of the terms T_1 and T_2 we know that they depend on the difference between system measurements at consecutive time instants, and therefore $(T_1^i - T_1^{i+1})$ and $(T_2^i - T_2^{i+1})$ are bounded. From Eq. (2-1.0.10) and Eq. (2-1.0.11), the Augmented Lagrangian and hence the dual function is composed of the terms T_1 and T_2. Hence, the difference between dual functions of $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$ is bounded. Let this bound be denoted by ϵ. In other words,

$$|g_{t_p}^{i+1}(Z) \;-\; g_{t_p}^{i}(Z)| \;<\; \epsilon \tag{2-3.0.5}$$

Having solved $\mathcal{Q}^i$, we can now focus our attention on solving the system identification problem $\mathcal{Q}^{i+1}$. Given p^{*i} and Z^i from the i^{th} identification cycle $\mathcal{Q}^i$, the natural question that arises is

> *What can we say about the Z^{i+1} using the information that the dual functions of $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$ are less than ϵ apart and using the knowledge of Z^i?*

Theorem 2-3.2 provides an answer to this question. However, before getting to the theorem, we will state and prove a useful result.

Lemma 2-3.1. *Consider two convex functions $h_1 : \mathbb{R}^{p\times q} \to \mathbb{R}$ and $h_2 : \mathbb{R}^{p\times q} \to \mathbb{R}$ such that*

$$|h_1(x) - h_2(x)| \;<\; \epsilon \qquad \forall x \in \mathbb{R}^{p\times q} \tag{2-3.1.1}$$

Let

$$x_1^* = \underset{x\in\mathbb{R}^{p\times q}}{\operatorname{argmin}} \quad h_1(x)$$
$$x_2^* = \underset{x\in\mathbb{R}^{p\times q}}{\operatorname{argmin}} \quad h_2(x)$$

Define

$$\mathcal{M} := \{x \in \mathbb{R}^{p\times q} : h_1(x) - \epsilon < h_1(x_1^* + \epsilon)\}, \qquad \mathcal{M} \subset \mathbb{R}^{p\times q}$$

Then we have the following:

1 $x_1^* \in \mathcal{M}$

2 $x_2^* \in \mathcal{M}$

Proof. The first part of the lemma follows trivially from

$$h_1(x_1^*) - \epsilon < h_1(x_1^*) + \epsilon$$

We now proceed to prove $x_2^* \in \mathcal{M}$. From Eq. (2-3.1.1),

$$h_2(x) < h_1(x) + \epsilon$$

So,

$$h_2(x_1^*) < h_1(x_1^*) + \epsilon \tag{2-3.1.2}$$

Consider $\tilde{x} \in \mathbb{R}^{p \times q}$ such that

$$h_2(\tilde{x}) \quad > \quad h_1(x_1^*) + \epsilon$$

From Eq. (2-3.1.2),

$$h_2(\tilde{x}) > h_2(x_1^*)$$

$\implies$ $\tilde{x}$ is not x_2^*

$\implies$ x_2^* must satisfy the condition

$$h_2(x_2^*) < h_1(x_1^*) + \epsilon \tag{2-3.1.3}$$

From Eq. (2-3.1.1) we have,

$$h_1(x) - \epsilon < h_2(x), \ \ \forall x \in \mathbb{R}^{p \times q} \tag{2-3.1.4}$$

From Eq. (2-3.1.3) and Eq. (2-3.1.4),

$$x_2^* \in \mathcal{M}$$

□

Theorem 2-3.2. *Consider the concave functions* $g_{t_p}^i :\in \mathbb{R}^{p \times q} \to \mathbb{R}$ *and* $g_{t_p}^{i+1} :\in \mathbb{R}^{p \times q} \to \mathbb{R}$*, as defined in Eq. (2-1.0.11). We know from Eq. (2-3.0.5):*

$$|g_{t_p}^i(Z) - g_{t_p}^{i+1}(Z)| < \epsilon$$

Define

$$\mathcal{P}_Z^i : \max_Z \ \ g_{t_p}^i(Z) \tag{2-3.2.1}$$

$$\mathcal{P}_Z^{i+1} : \max_Z \ \ g_{t_p}^{i+1}(Z) \tag{2-3.2.2}$$

Let

$$Z^i = \arg\max_Z \quad g^i_{t_p}(Z)$$
$$Z^{i+1} = \arg\max_Z \quad g^{i+1}_{t_p}(Z)$$

Then we have:

$$Z^i \in \mathcal{M}_Z$$
$$Z^{i+1} \in \mathcal{M}_Z$$

where,

$$\mathcal{M}_Z := \{Z \in \mathbb{R}^{p\times q} : g^i_{t_p}(Z) - \epsilon < g^i_{t_p}(Z^i) + \epsilon\}$$

Proof. From [23] we know that minimization of a convex function is equivalent to maximization of a concave function. So,

$$Z^i = \arg\min_Z \quad (-g^i_{t_p}(Z))$$
$$Z^{i+1} = \arg\min_Z \quad (-g^{i+1}_{t_p}(Z))$$

From Lemma 2-3.1 it directly follows that:

$$Z^i \in \mathcal{M}_Z$$
$$Z^{i+1} \in \mathcal{M}_Z$$

□

Thus, it immediately follows that the solution of $\mathcal{P}^i_Z$ is an excellent choice of initial condition for the problem $\mathcal{P}^{i+1}_Z$.

We know, from Section 2-1-1, that ADMM proceeds by maximizing the dual function, and in the process, finds the minimizers of the primal variables in each iteration. Since the problem $\mathcal{P}^{i+1}$ from Eq. (2-3.0.2) is convex, $\mathbf{x}^i$, X^i are justified, by Theorem 2-3.2, to be used as the initial condition for solving $\mathcal{P}^{i+1}$. Thus, with $\mathbf{x}^i$, X^i and Z^i as the initial condition for solving the problem $\mathcal{P}^{i+1}$, we start in the same set which contains Z^{i+1}.

In this section, the problem of recursive system identification was taken up. It was shown that the information from the last identification cycle can be used to obtain a justified choice of initial condition for the current identification. In this way, information regaring the subspace describing the underlying system is passed on between consecutive identification cycles. So far, we have performed a theoretical anaysis of recursive system identification using N2SID. In the next section, N2SID is recursively implemented on a CD-player arm system by using the results of $\mathcal{Q}^i$ as the initial condition for $\mathcal{Q}^{i+1}$.

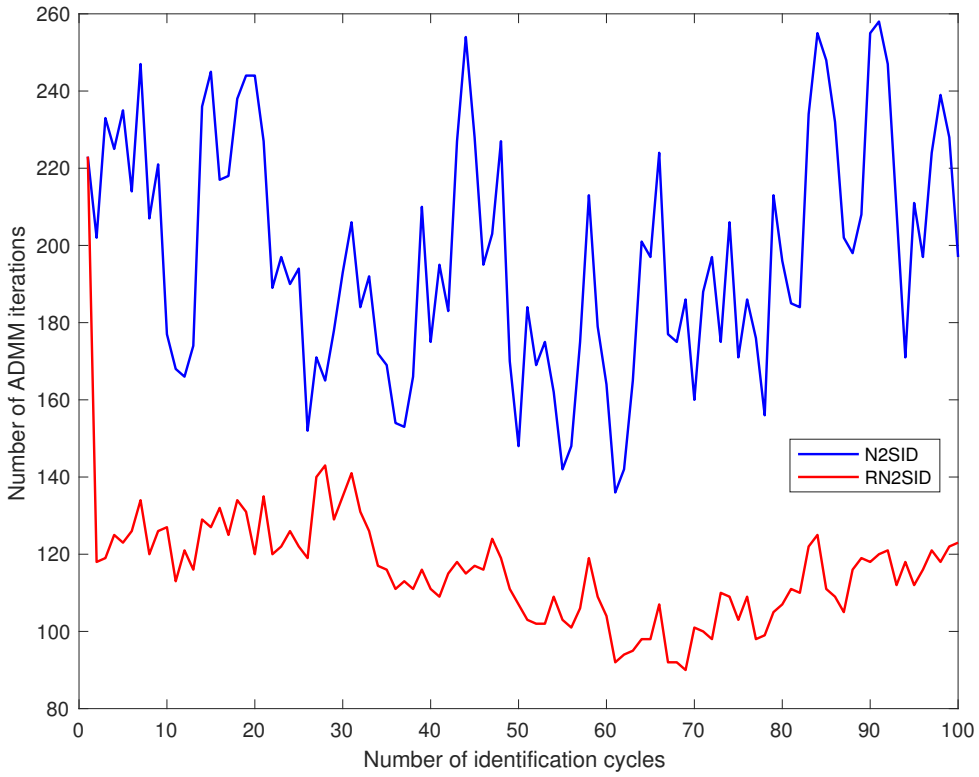


Figure 2-5: Number of iterations in ADMM for a CD player arm system

2-3-2 Results

In this section, system identification is recursively performed using N2SID to evaluate the theory developed in Section 2-3. In order to have clear view of the effect of the developed choice of initial condition, we implement N2SID as in [11], but for two different choices of initial condition. The first choice being the default initial condition as given in Algorithm 1, let us call it *Case 1*. The second choice is the result obtained from the identification $\mathcal{Q}^i - \mathbf{x}^i$, X^i and Z^i – let us call it *Case 2*. It is worth emphasizing here that the ADMM algorithm is not terminated early in this evaluation. This is to have a clear picture of how many iterations *Case 2* requires to converge. Also, in this evaluation, FRSVT is not used for the computation of Singular Value Thresholding in the computation of the minimizer X in ADMM.

The evaluation, as described, is carried out on the same CD-player arm system for 100 time instants, hence requiring 100 identification cycles. The total number of iterations required for convergence of ADMM in each cycle are compared and shown in Figure 2-5. Along with the number of iterations required, the corresponding VAF obtained in each cycle for both the cases are shown in Figure 2-6.

It is seen from the results that using the system information obtained from last identification cycle, noticeably decreased the number of iterations required to identify the system, while resulting in roughly the same VAF. The average time taken to perform one identification

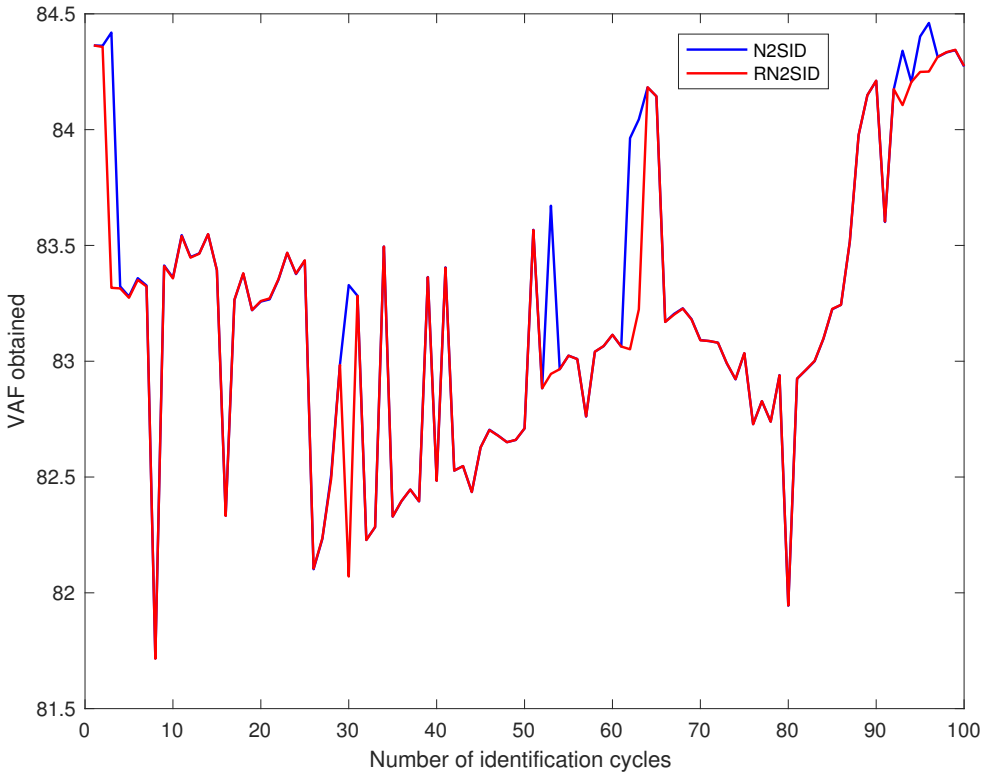


Figure 2-6: VAF obtained for 1000 identification cycles for a CD player arm system

Table 2-5: Comparison of results for different choice of initial condition

	Time, in sec	VAF	k_{ADMM}
Case 1	2.37	83.23	197
Case 2	1.39	83.17	116

cycle, for both cases, is listed in Table 2-5. Along with the average time taken, the average VAF obtained and the average number of ADMM iterations required, k_{ADMM}, to perform one identification cycle is tabulated for both the cases. With a justified choice of initial condition for $\mathcal{Q}^{i+1}$ (*Case 2*), the number of iterations required for ADMM to converge are significantly lesser than *Case 1*. The same is reflected through the time taken in both the cases: there is 41.35% reduction in the time taken to converge with the choice of initial condition justified in Section 2-3.

To summarize, using the results from $\mathcal{Q}^i$ as the initial condition for $\mathcal{Q}^{i+1}$ is an excellent choice for performing recursive subspace identification using N2SID. Having made some changes in N2SID algorithm in Section 2-2 to improve it's speed, the choice of initial conditions for its recursive implementation was justified and evaluated in this section. So far, all the improvements have only been evaluated individually. In the next section, their combined analysis is performed and the corresponding results are presented.

2-4 Performance analysis of Recursive N2SID

During the review of N2SID, in Section 2-1, it was noted that the N2SID problem, Eq. (2-1.0.7), is solved using the ADMM algorithm instead of SDP solvers mainly because of the attractive feature of ADMM to handle sub-differentiable convex problems with modest accuracy within a few iterations. ADMM was taken up in detail in Section 2-1-1 and it's convergence analysis, through the primal and dual residual, was performed. This revealed the slow convergence behavior of ADMM, as noted in [15]. From further analysis, it was concluded in Section 2-2-1 that early stopping of ADMM resulted in a significant reduction in computation time, while sacrificing minimally on accuracy. To further reduce the computational time of N2SID, few modifications in the ADMM algorithm were suggested in Section 2-2-2 and it was concluded that FRSVT will be employed to compute the singular value thresholding in the ADMM algorithm.

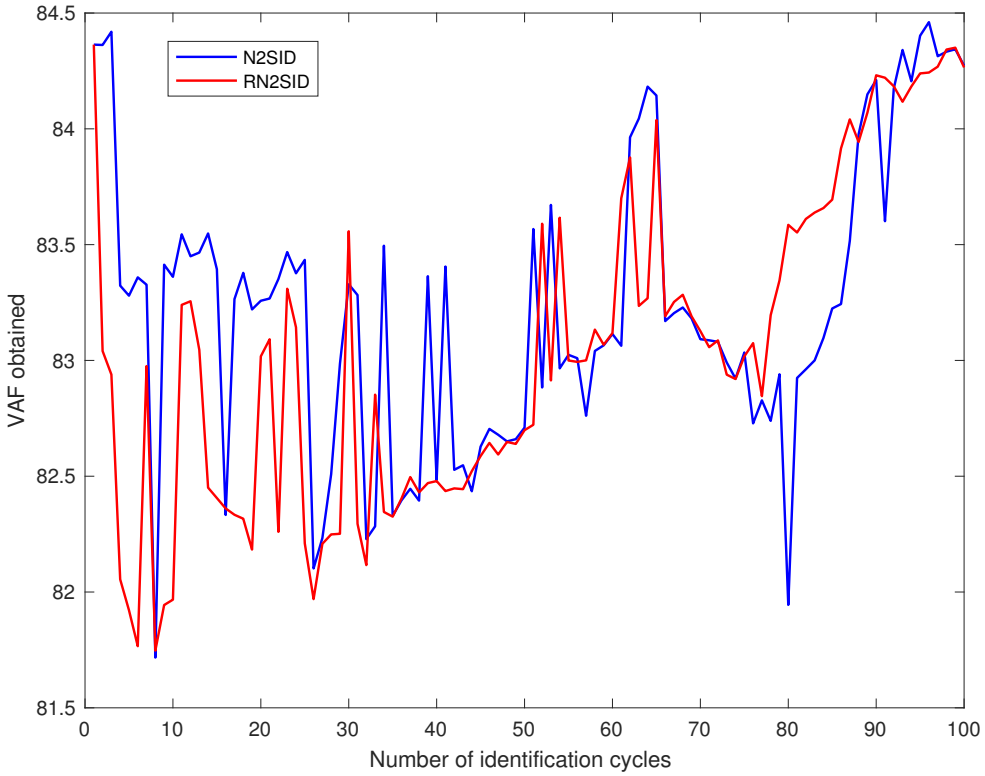


Figure 2-7: VAF obtained from recursive identification

In Section 2-3, the recursive N2SID algorithm was taken up and potential improvements in computation time were investigated. It was concluded that the information from the previous identification cycle can be used to select the initial condition for the current identification cycle. This resulted in a significant reduction in computation time.

As a summary, we list below all the modifications that we incorporate in the N2SID algorithm. The modified version is recursive and we call it Recursive N2SID, or RN2SID.

Modification 1: Early stopping of the ADMM algorithm.

Modification 2: Using the alternative method of Singular Value Thresholding - FRSVT, instead of conventional computation through the use of SVD.

Modification 3: Using the results of the previous identification as an initial guess for the current identification cycle.

RN2SID is now evaluated for the same CD-player arm system. Recursive identification using N2SID, as in [11], and RN2SID is performed for 100 time instants, with the length of each identification dataset being $N = 200$. The VAF obtained and the time taken, for both the approaches, in each identification cycle are shown in Figure 2-7 and Figure 2-8 respectively. The time taken by RN2SID is noticeably lesser than N2SID, with a VAF comparable to that obtained by N2SID. The average time taken by RN2SID to perform one identification cycle is 0.46 seconds.

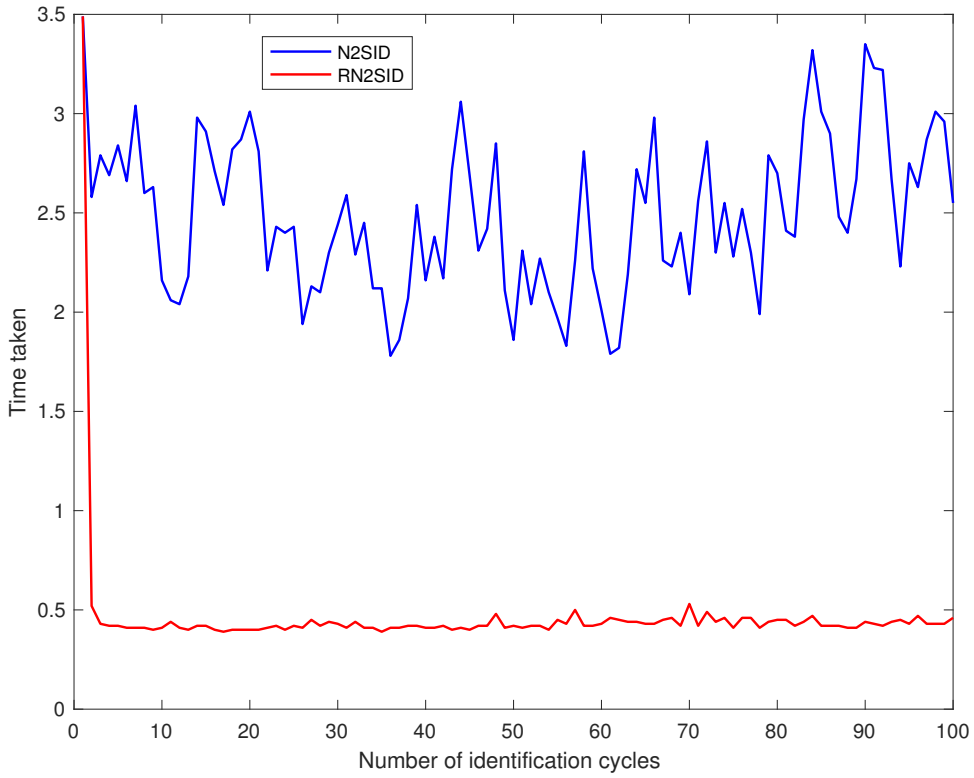


Figure 2-8: Time taken for recursive identification

The results obtained in this entire chapter are summarized in Table 2-6. In Section 2-1-2, we had seen that the N2SID algorithm was too slow to be recursively implemented on systems with sampling time lesser than 3 seconds. With all the modifications, the identification algorithm – RN2SID – demonstrated a significant reduction of computation time. All systems with a sampling time of higher than 0.5 seconds can now be identified using this novel technique.

Table 2-6: Evaluation of time taken for identification with each modification

	Time taken in sec
N2SID	2.98
Modification 1	0.62
Modification 2	3.53
Modification 3	1.39
RN2SID	0.46

As seen in the beginning of this Chapter, most subspace identification techniques realize the system matrices in the final step of the identification. However, this step of parameter estimation, adds to the computation time. If there is no need to know the system matrices explicitly, then this step can be skipped. In Chapter 3 we will formulate a control law that will not require knowledge of the system matrices, thus allowing us to skip the parameter estimation step.

Chapter 3

Controller Formulation

System identification is a desirable first step in the functioning of an online adaptive controller since it provides information about the dynamics of the system. Once the estimated model of the system is obtained, it is then used to compute a control signal which is supplied to the plant. In this chapter, we aim to design a controller which can utilize the system model obtained from the previous step of system identification and generate a control signal.

In general, the knowledge of system matrices plays a crucial role in the design of most standard controllers. In Chapter 2, it was seen that, the third step in system identification, namely Parameter Estimation, was responsible for explicitly extracting system matrices from the computed subspace (in our case, $T_{u,s}$ and $T_{y,s}$). However, Parameter Estimation is computationally expensive since the model order has to be estimated first. Gaining knowledge about the model order has to be done either through examination of dominant singular values or by using model order estimation criteria. These require operations like Singular Value Decomposition (SVD) which are computationally heavy. It is therefore desirable to design a controller directly in terms of the subspace obtained from identification, without ever explicitly extracting the system matrices. A Model Predictive Control (MPC) framework offers exactly this feature: it allows for controller design even when the system matrices are not explicitly known. Specifically, this is known as Subspace Predictive Control (SPC) [24].

In Section 3-1, we discuss how a MPC controller can be formulated directly in terms of the subspace acquired from N2SID. As it turns out, such a design requires an estimate of the initial state. Since we do not explicitly extract the system matrices, the initial state estimate is not available in our case. Therefore, in Section 3-2 we present the formulation of a MPC controller tailored for the requirements set by the N2SID algorithm.

3-1 Overview of MPC

When it was first deployed in the real world, MPC was mainly in process control industries to handle constraints, for example to enforce quality and production rates. In the early years, MPC used impulse and step response parameter based methods to model and control the

system under consideration. However, MPC as we see today, heavily relies on the state-space representation of the underlying system. A detailed review of the evolution of MPC was given in [4].

The concepts of MPC are particularly useful in our case because the system matrices are not explicitly computed in order to save computational time. An important concept that we borrow from MPC and use in the design procedure is the Receding horizon principle - computing the control action for the current time instant while taking into account future behavior. As it is shown further, this is crucial in the design process. We now look into the implementation details of Receding horizon principle along with the technique employed to formulate the controller directly in terms of the acquired subspace. To keep the equations simple, in order to understand the crux of the design technique, the explanation in this section is done through a simple example system which is noiseless.

Consider a discrete time system as given in Eq. (3-1.0.1).

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{3-1.0.1}$$

A typical method of designing a control law for such a system in the MPC framework involves stacking the outputs and inputs from the current time instant k until the Prediction horizon N_p. Stacked output Y can then be expressed as

$$Y = O_s x(k) + T_{u,s} U \tag{3-1.0.2}$$

where

$$U = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(N_p) \end{bmatrix} Y = \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(N_p) \end{bmatrix} O_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N_p} \end{bmatrix} T_{u,s} = \begin{bmatrix} D & 0 & \dots & 0 \\ CB & D & & \\ CAB & CB & \ddots & \vdots \\ \vdots & & & \\ CA^{N_p-1}B & \dots & CB & D \end{bmatrix}$$

For the 'stacked' system, the objective function is defined in terms of the requirements and constraints. For an unconstrained system with the objective of reference tracking, a general optimization function is defined as:

$$J = (Y_{ref} - Y)^T (Y_{ref} - Y) + \lambda U^T U \tag{3-1.0.3}$$

where Y_{ref} is a stacked vector of desired output values. The control law is derived as the argument U that minimizes the objective function given in Eq. (3-1.0.3). With an initial estimate of the state, the minimizing control input vector is derived at each time instant and only the current value is implemented. The process is repeated at each time instant and hence this principle is called Receding Horizon.

The important point to notice here is that the control law is in terms of $\tilde{C}$ and $\tilde{D}$, not directly in terms of system matrices A, B, C, D. This is precisely the feature of the controller that fits the requirements of our problem statement. It saves computation time not only by avoiding

the extraction of system matrices but also by avoiding the approximation of the model order. This feature is the backbone of controller design in SPC which was first introduced in [24].

In this example design procedure, we see that an estimate of the initial state is required for the derivation of the control value. However, since system matrices are not explicitly computed due to constraints on computational time, an estimate of the initial state is not available. In Section 3-2, a way to go around this problem is introduced and formulation of the controller for our case is developed.

3-2 Formulation of control law for recursive N2SID

In the last section, a way to formulate the MPC controller directly in terms of the available subspace was presented, which is appropriately called Subspace Predictive Control (SPC). The principles and techniques of the design example in the last section are extended here to derive a control law that uses the subspace resulting from the recursive N2SID method as given in Chapter 2. Therefore, the equations defining the system to be identified are reconsidered here and the control law is derived based on the same. The notations used in this section are the same as used in Chapter 2.

Consider Eq. (2-1.0.6) and for simplicity assume noise to be zero, hence having $\hat{Y}_s = Y_s$. From the solution of ADMM, and hence N2SID, we have the estimates of $T_{u,s}$ and $T_{y,s}$. From measurements of the system we have Y_s and U_s available for N time instants. The corresponding $O_s X$ is obtained from Eq. (2-1.0.6). We now have the data equation,

$$Y_s = O_s X + T_{u,s} U_s + T_{y,s} Y_s \tag{3-2.0.1}$$

However, we do not yet have an estimate for the initial state. Under the assumption that the system under consideration is finite dimensional, it is noted that the current output is influenced by finite number of past inputs. Hence it is reasonable to assume:

$$CA^i = 0 \qquad \forall \ i > (s-1) \tag{3-2.0.2}$$

This property is used to extend the observability matrix O_s and the acquired subspaces $T_{u,s}$ and $T_{y,s}$ to a "Band form". From identification, $T_{u,s}$, $T_{y,s}$ of block-size $s \times s$, and O_s of block-size $s \times 1$ are available. Under the assumption made in Eq. (3-2.0.2), $T_{u,s}$, $T_{y,s}$ and O_s are extended to block-size $N_p \times N_p$ and $N_p \times 1$ respectively. Let the extended forms be called $T_{u,band}$, $T_{y,band}$ and $O_{s,ext}$ which assume the band form as:

$$O_{s,ext} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad T_{u,band} = \begin{bmatrix} D & 0 & \dots & & 0 \\ CB & D & \dots & & \\ \vdots & CB & & & \vdots \\ & & & \ddots & \\ CA^{s-2}B & & & & 0 \\ 0 & CA^{s-2}B & \dots & & \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots CA^{s-2}B & \dots & D \end{bmatrix}$$

$$T_{y,band} = \begin{bmatrix} 0 & 0 & \dots & & 0 \\ CK & 0 & \dots & & \\ \vdots & CK & & & \vdots \\ & & & \ddots & \\ CA^{s-2}K & & & & 0 \\ 0 & CA^{s-2}K & \dots & & \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots CA^{s-2}K & \dots & 0 \end{bmatrix}$$

The extended matrices are partitioned according to past and future time instants of the input-output data. $T_{u,band}$ is partitioned as in Eq. (3-2.0.3), where the block-size of T_{u1} is $s \times s$ and T_{u4} is $(N_p - s) \times (N_p - s)$. Correspondingly, the sizes of T_{u2} and T_{u3} can be observed. $T_{y,band}$ is also partitioned in the same way.

$$T_{u,band} = \begin{bmatrix} T_{u1} & T_{u2} \\ T_{u3} & T_{u4} \end{bmatrix} \tag{3-2.0.3}$$

$O_{s,ext}$ is split as $\begin{bmatrix} O_{s1} \\ O_{s2} \end{bmatrix}$ where O_{s1} is of block-size $s \times 1$. From observation, O_{s2} is a zero vector.

Define a vector Y as $Y = \begin{bmatrix} Y_p \\ Y_f \end{bmatrix}$ where

$$Y_p = \begin{bmatrix} y(N-s+1) \\ \cdot \\ \cdot \\ y(N) \end{bmatrix} \qquad Y_f = \begin{bmatrix} y(N+1) \\ \cdot \\ \cdot \\ y(N_p) \end{bmatrix} \tag{3-2.0.4}$$

Similarly, define U as $U = \begin{bmatrix} U_p \\ U_f \end{bmatrix}$

The system from Eq. (3-2.0.1) can be extended to the band form, which can be expressed as:

$$\begin{bmatrix} Y_p \\ Y_f \end{bmatrix} = \begin{bmatrix} O_{s1} \\ O_{s2} \end{bmatrix} x(N-s+1) + \begin{bmatrix} T_{u1} & T_{u2} \\ T_{u3} & T_{u4} \end{bmatrix} \begin{bmatrix} U_p \\ U_f \end{bmatrix} + \begin{bmatrix} T_{y1} & T_{y2} \\ T_{y3} & T_{y4} \end{bmatrix} \begin{bmatrix} Y_p \\ Y_f \end{bmatrix} \tag{3-2.0.5}$$

From the expressions in the band form and Eq. (3-2.0.5), the following observations are made:

1 T_{u1} is the matrix $T_{u,s}$ that is obtained from identification.

2 T_{u2} is a zero matrix.

3 Vectors U_p and Y_p contain the last s measurement values that were used for identification.

4 Vectors U_f and Y_f denote the future input and output values.

5 As O_{s2} is a zero vector, Y_f is independent of the initial state.

Writing the equation for future output values Y_f Eq. (3-2.0.5), we have,

$$\begin{aligned} Y_f &= T_{u3}U_p + T_{u4}U_f + T_{y3}Y_p + T_{y4}Y_f \\ \implies Y_f &= (I - T_{y4})^{-1}(T_{u3}U_p + T_{u4}U_f + T_{y3}Y_p) \end{aligned} \tag{3-2.0.6}$$

The aim is to compute the optimal future input signal U_f such that Y_f tracks a reference signal R. This is achieved by minimizing the objective function J, as in Eq. (3-2.0.7), over the optimization variable U_f. Here, λ_p is the penalty on the control input. The argument U_f that achieves the minimum of the objective function is denoted by U_c. Y_f from equation Eq. (3-2.0.6) is substituted in Eq. (3-2.0.7). The minimum of the resulting objective function with respect to U_f is analytically found and is given in equation Eq. (3-2.0.8).

$$J = (Y_f - R)^T(Y_f - R) + \lambda_p U_f^T U_f \tag{3-2.0.7}$$

$$U_c = (C_2^T C_2 + \lambda_p I)^{-1} C_2^T (R - C_1) \tag{3-2.0.8}$$

where $C_1 = (I - T_{y4})^{-1}(T_{u3}U_p + T_{y3}Y_p)$ and $C_2 = (I - T_{y4})^{-1}T_{u4}$.

U_c is a vector containing the optimal control signal values from the current time instant $(N+1)$ till N_p. Adhering to the receding horizon concept of MPC, only the first value of this vector is given as a control input to the system. The corresponding output measurement $y(N+1)$ is recorded. The tuning parameters in this design are the prediction horizon N_p and the penalty parameter λ_p. Depending on the requirement of the desired system response, λ_p is tuned to result in an under-damped or damped response.

For the next time instant, $u(N+1)$ and $y(N+1)$ are appended to the identification datasets and the resulting Hankel matrices of input and output will correspond to the last N measurements. This is done by discarding the oldest measurement and using the latest measurement. Hence, as seen in Chapter 2, the $(i+1)^{th}$ system identification, $\mathcal{Q}^{i+1}$, will contain the measurements from $(i-N+2)$ to $(i+1)$. This ensures that the size of datasets used for identification will be constant at N.

Maintaining the recursive nature of the methodology, $\mathcal{Q}^{i+1}$ identification is performed. Then using the resulting subspaces, along with the measurements, the control law generation method, as described in this section, is repeated. The corresponding control value is supplied to the plant. This process of system identification and control is repeated at each sampling time instant. Since the entire procedure is recursive, it results in an inherently adaptive control framework.

To summarize, the methodology to formulate a control law in terms of the subspaces available from identification performed using N2SID was presented. It was seen how, with the assumption of the system being finite dimensional, the requirement of an estimate of the initial state can be bypassed to generate the control value. During the presentation of the methodology, it was also seen that the processes of system identification and control value computation are combined and executed at each sampling time instant to adaptively control the system in consideration. In the next chapter, the developed methodology is summarized and formulated into an algorithm. It is then implemented on different systems and the corresponding results are presented.

Chapter 4

Algorithmization and results

The aim throughout this thesis has been to develop an algorithm that is able to identify and control systems in the real world. To ensure appropriate and effective control, our framework must be able to react quickly to changing system data. Due to this, it is necessary to have an algorithm which has a low computational complexity. In a typical real world application, measurements are periodically taken using sensors which provide information regarding the current state of the system. In order to utilize this information efficiently, we must convert our ideas into a set of coherent logical operations which can act on this measured data and generate control signals.

The theory concerning the efficient identification of the system was developed in Chapter 2. Following this, the control framework was developed in Chapter 3. In this Chapter, we compile all the procedures outlined in the previous two chapters into an algorithm which can then be deployed on various systems. To this effect, we implement this algorithm both in simulation (in Section 4-2) and on a physical system (in Section 4-3).

4-1 Algorithm: Recursive N2SID with predictive control

In order to effectively identify and control, recursively, it is necessary to gain insight into the system in consideration. Like in Section 2-1, the system is identified using N2SID for a range of values of the regularization parameter λ. This step is needed to compute the value of λ that results in an estimated model that best describes the system. Once the estimated model is obtained, a simulation of the system under the effect of a MPC controller, as described in Chapter 3, is carried out for a range of penalty parameters λ_p. Depending on the nature of desired system response - underdamped or damped - an appropriate value of the penalty parameter is selected. Once the tuning parameters are fixed for the system under consideration, the real-time process of recursively identifying and controlling starts. This is carried out by implementing the algorithm - Recursive N2SID with predictive control - as given in Algorithm 4.

Algorithm 4 Recursive N2SID with predictive control

`Input:` $u, y, \lambda, R, \lambda_p$
`Initialize:` $\mathbf{x}^0 = 0, \mathtt{X}^0 = -\mathcal{B}, Z^0 = 0, i = 0$
while true **do**
$i = i + 1$
▷ Identify the plant
$[T^e_{u,s}, T^e_{y,s}, \mathbf{x}^i, \mathrm{X}^i, Z^i] =$ `RN2SID` $(u, y, \lambda, \mathbf{x}^{i-1}, \mathrm{X}^{i-1}, Z^{i-1})$
▷ Compute control value using estimated subspaces
$U_c =$ `SPC` $(T^e_{u,s}, T^e_{y,s}, u, y, R, \lambda_p)$
$u_c = U_c(1)$
▷ Execute control action on plant
$y_m = Cx(i) + Du_c + e(i)$
$x(i+1) = A_{sys}x(i) + B_{sys}u_c + Ke(i)$
▷ Acquire measurements from plant
$y_i = y_m$
$u_i = u_c$
▷ Discard the oldest and append the latest measurement
$u = [u(2) \quad u(3) \quad ... \quad u(N) \quad u_i]$
$y = [y(2) \quad y(3) \quad ... \quad y(N) \quad y_i]$

Having presented the algorithm, we now move on to validating it. This is carried out by implementing it on a range of systems - both in simulations and in real world.

4-2 Implementation in simulations

In Chapter 2, the recursive identification part of Algorithm 4 was tested on the CD-player arm system. In this section, we implement the entire algorithm and evaluate the performance of the controller on some systems in simulations. First and foremost, the entire algorithm is tested on two LTI systems and then on a Flutter model, which is a linear parameter varying system.

4-2-1 Simulations on LTI systems

The algorithm is first evaluated on the CD-player arm system, taken from [18], which has been used to test various modifications made in Chapter 2. It is an LTI system with two inputs and two outputs. As mentioned in Section 4-1, system identification using N2SID, in its original form as in [11], is first carried out for a range of regularization parameters λ. Then for the estimated system, simulations are carried out to determine the penalty parameter λ_p in the control law. The recursive process of identification and control, as given in the Algorithm 4, is carried out for 80 sampling time instants and the response of the system, for both the outputs, is given in Figure 4-1.

Similar testing is carried out on a system that is identified using data from a pH neutralization process that is carried out in a stirring tank. This dataset is also taken from [18]. This system has two inputs (one being the concentration of the acid solution and the other being

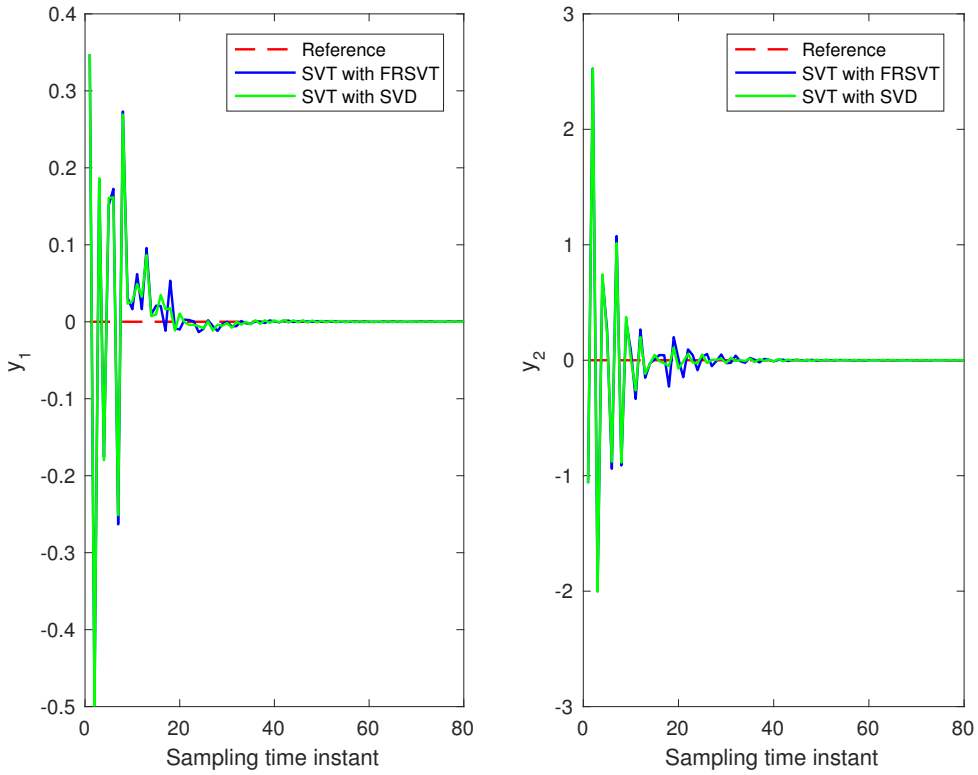


Figure 4-1: Output response of the CD-player arm system

the concentration of the base solution) and one output. The output is the pH of the resulting solution. The results of the implementation of Algorithm 4 for this system is shown in Figure 4-2.

For both the systems, the algorithm is implemented once with the Singular Value Thresholding in ADMM carried out using SVD and once with the alternative method FRSVT, given in Section 2-2-2. This is done in order to additionally evaluate FRSVT by comparing it with SVT using SVD, in terms of performance effects and the computational time taken. As can be seen in Figure 4-1 and Figure 4-2, using FRSVT does not negatively affect the performance of the host algorithm, N2SID in our case.

Table 4-1: Results of implementation of Algorithm 4

	λ	VAF	λ_p	t_{FRSVT} in sec	t_{SVD} in sec
CD-player arm system	4.28	84.36	190	0.7025	0.6513
pH neutralization process	0.4833	41.85	1.45	0.3835	0.3011

Table 4-1 lists out the numerical results of the implementation of Algorithm 4 on both the systems. The time taken per identification and control action, with FRSVT and SVD employed to compute singular value thresholding in ADMM in N2SID, are mentioned as t_{FRSVT} and t_{SVD} respectively. Furthermore, the time taken when FRSVT is used for Singular Value

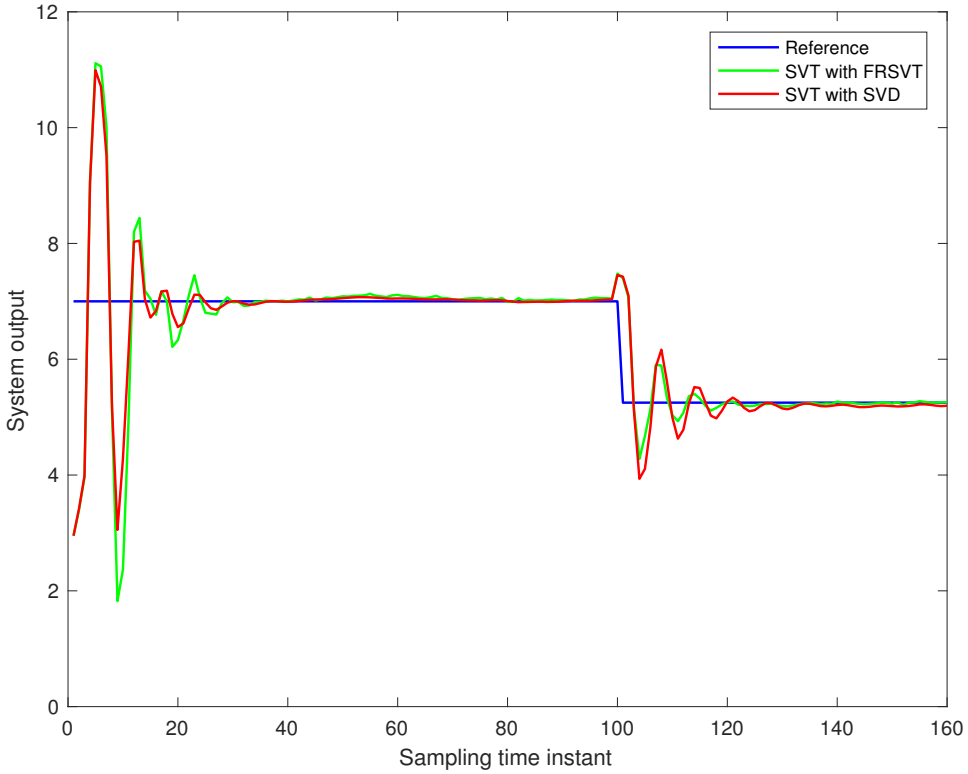


Figure 4-2: Output response of a pH neutralization process

Thresholding is comparable to that of using SVD.

It can be concluded that Algorithm 4 results in satisfactory control over LTI systems. However, in order to achieve control over an LTI system, there is not much usefulness in repeatedly identifying it. Nevertheless, this test is useful in validating the coherency of all the steps in the algorithm. We now move on from validating the algorithm on LTI systems to a parameter varying system.

4-2-2 Simulation on an LPV system

In the methodology presented in this work, the system is identified at each sampling time instant and the computed control value is such that it is based on the latest information about the system. Therefore, the developed work, as concisely given in Algorithm 4, is more meaningful when the system is changing and hence it is necessary to validate the method on LPV or LTV systems. In this section, a flutter model from [25], which varies with the wind speed is selected for testing the developed algorithm. It is a single input single output system; the output being the rotational pitch angle of the flutter.

Algorithm 4 is implemented on the flutter model while continuously varying the wind speed at each time instant. Recursive identification and control value computation is carried out at each sampling time instant and the result of the controlled output response is shown in

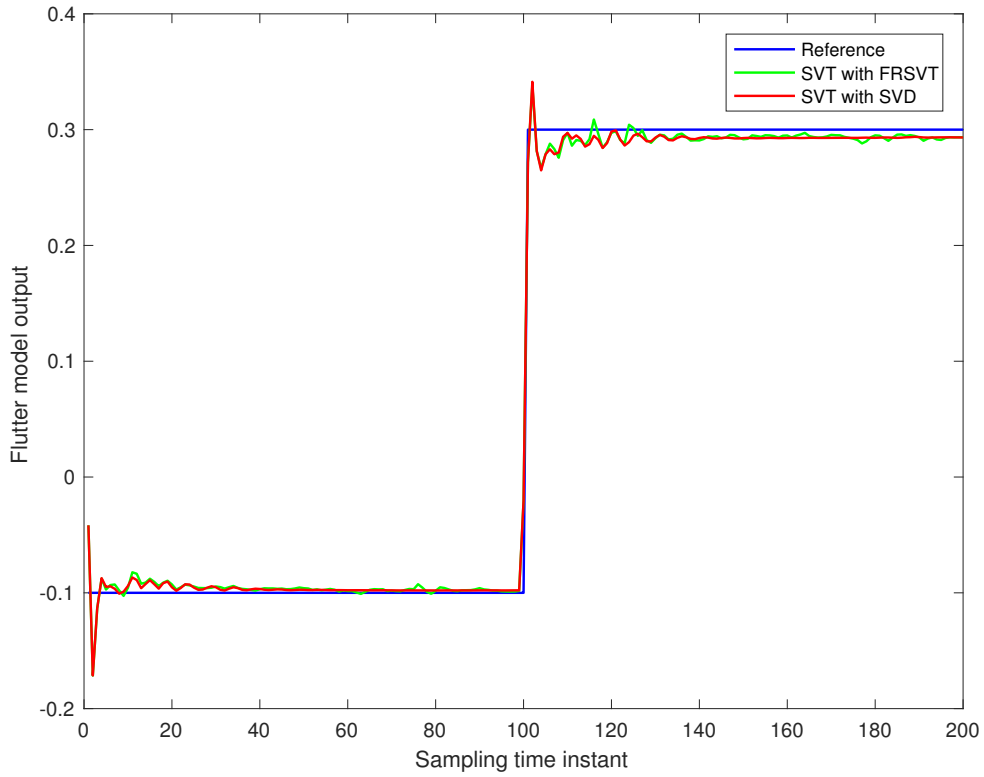


Figure 4-3: Output response of the LPV flutter model

Figure 4-3. The reference signal for the pitch angle is changed from -0.1 to 0.3 with the wind speed varying at the same time. The entire implementation is carried out for two different instances: one for Singular Value Thresholding carried out with FRSVT and the other for Singular Value Thresholding carried out with the conventional SVD, just like the analysis carried out for LTI systems. Time taken when FRSVT is used is 0.34 seconds whereas the time taken when SVD is used is 0.27 seconds. The effect of using FRSVT is almost the same as using SVD; FRSVT did not affect the performance of the host N2SID algorithm.

The performance of the SPC controller, which is designed as a part of the Algorithm 4, is compared with a PID controller. In order to do so, a PID controller is designed for the flutter model with a constant wind speed. The corresponding response of the system with under the effect of PID control is shown in Figure 4-4. The reference that is to be tracked is varied from -0.1 to 0.3. Note that this response is for the flutter model with a constant wind speed, hence acting like an LTI system.

The output response of the model under the effect of PID control when the flutter model is simulated with varying wind speed is shown in Figure 4-5. For comparison the response of the system under the control of SPC is also shown. The wind speed varied for this evaluation is the same as the wind speed varied for obtaining the results shown in Figure 4-3. While the control of PID was mostly satisfactory when the wind speed was constant, it is not so when the wind speed is varied. This implies that the PID controller is not able to account for the

changes in the system dynamics, whereas due to recursive system identification the changes in the system dynamics are well accounted for by the SPC.

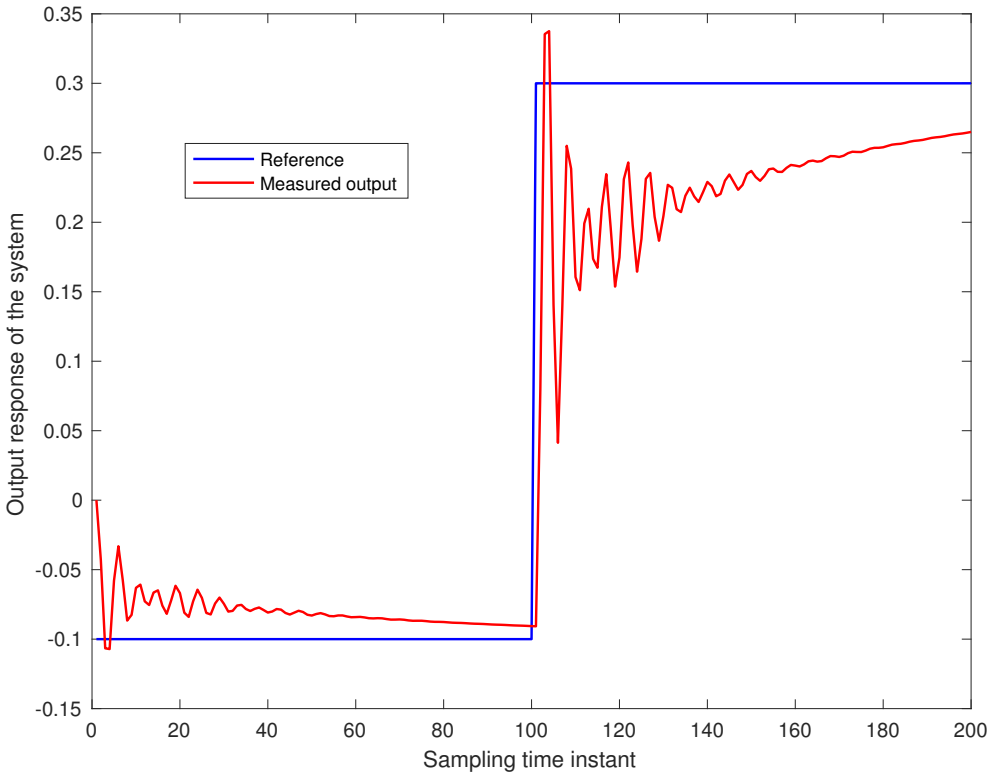


Figure 4-4: Output response of the flutter model when controlled using a PID controller

In this section, the results of implementation of Algorithm 4 on two LTI systems and one LPV system were presented. It was seen that the computational time required for the algorithm varied for systems, but it was always lesser than 0.7 seconds. Since the developed methodology performs recursive system identification, the method is more useful for changing systems. Hence it was validated on a flutter model, which is an LPV system that varies with the speed of wind. It was noted that while SPC accounted for the changes in the system dynamics and accordingly controlled the system, the PID control performed poorly when the system was changing with the wind speed. This result emphasized the inherent adaptive nature of the developed methodology. To further validate the work in this thesis, it is necessary to evaluate it on a physical system, which is performed in the next section.

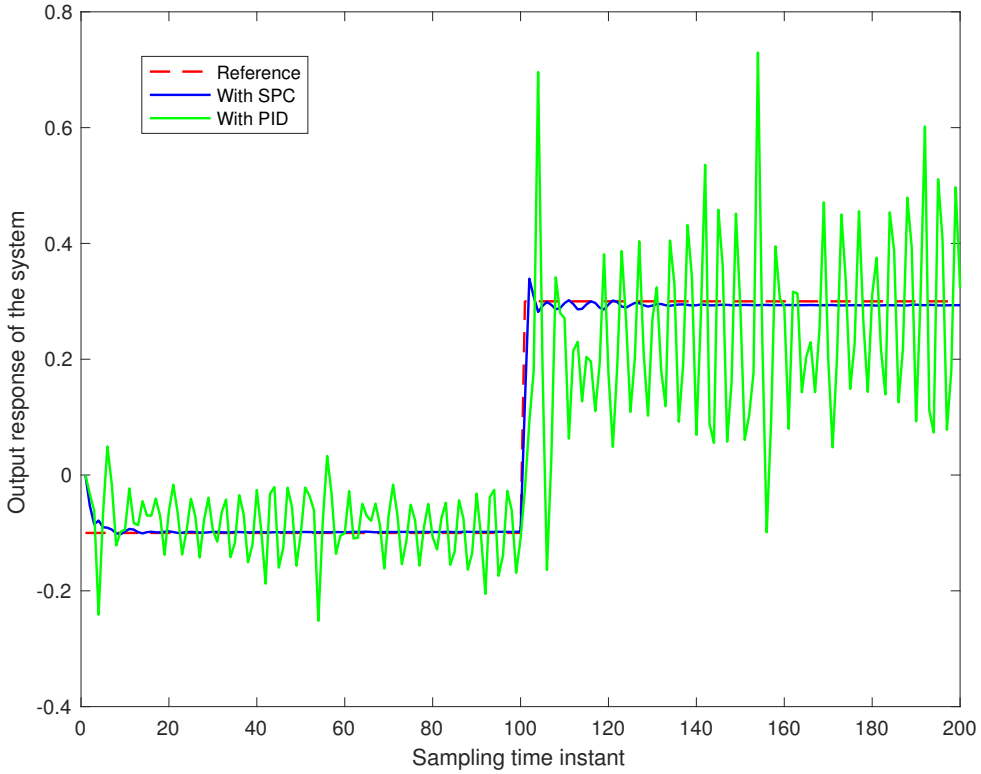


Figure 4-5: Comparison of PID and SPC controllers through output response of the flutter model

4-3 Implementation on a physical system

The developed methodology, having been tested on different LTI and LPV systems in simulations, is now tested on a real physical system. The selected setup is a Cart-Beam system - a cart that horizontally moves on a bar and a flexible beam that is fixed on the cart, see Figure 4-6. The position of the cart on the bar can be measured and manipulated directly, whereas the movement of the beam can only be indirectly controlled through the cart. The system is a single input two output system and the aim here is to control the deviation of the beam from the vertical position. Although the beam is flexible, it's physical properties are such that it can deviate only little from the vertical axis. Due to the physical construction of the setup, the movements of the cart and the beam are in opposite directions.

When a constant sinusoidal wave is given as the input, the cart moves sinusoidally along the bar. However, the system has an inherent drift causing the cart to slowly drift even with a constant sine wave input; this is shown in Figure 4-7. As a result, there is drift in the beam movements as well. Since our aim is to control the beam angle, it is necessary to compensate for the drift. This is done by controlling the position of the cart using an external PID controller, leaving the beam angle output free. The results of this technique are shown in Figure 4-8. Since the control of the cart position is not the aim here, the lag in the cart following the sinusoidal reference is not a problem. It is important to notice the absence of

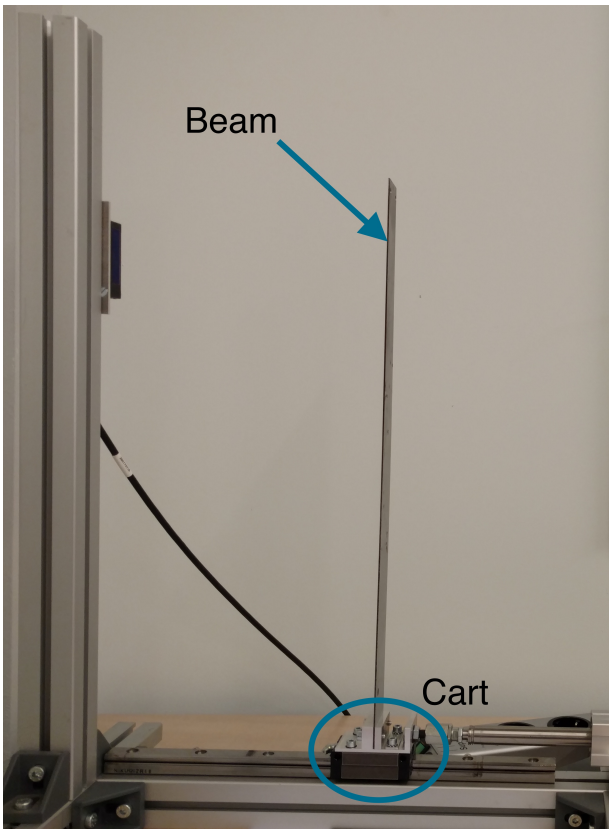


Figure 4-6: The Cart-Beam system

the drift in the system.

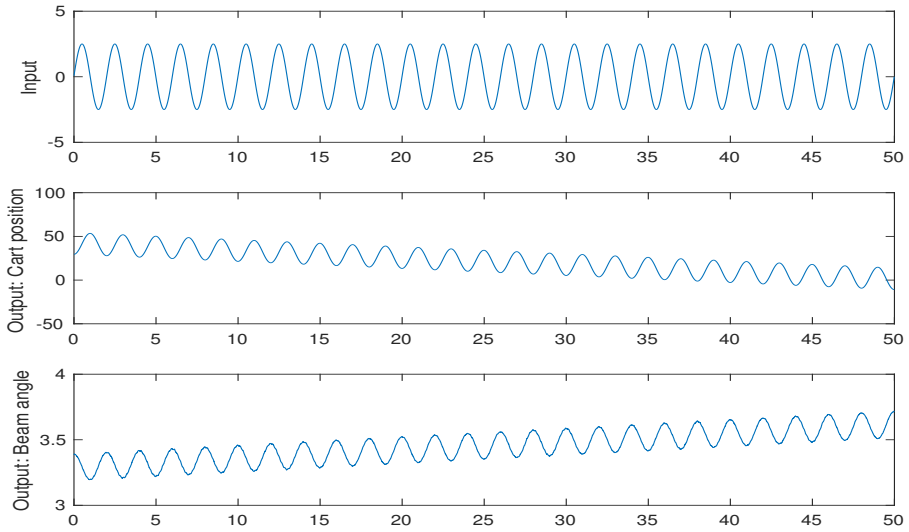


Figure 4-7: Presence of drift in the output response of the Cart-beam system

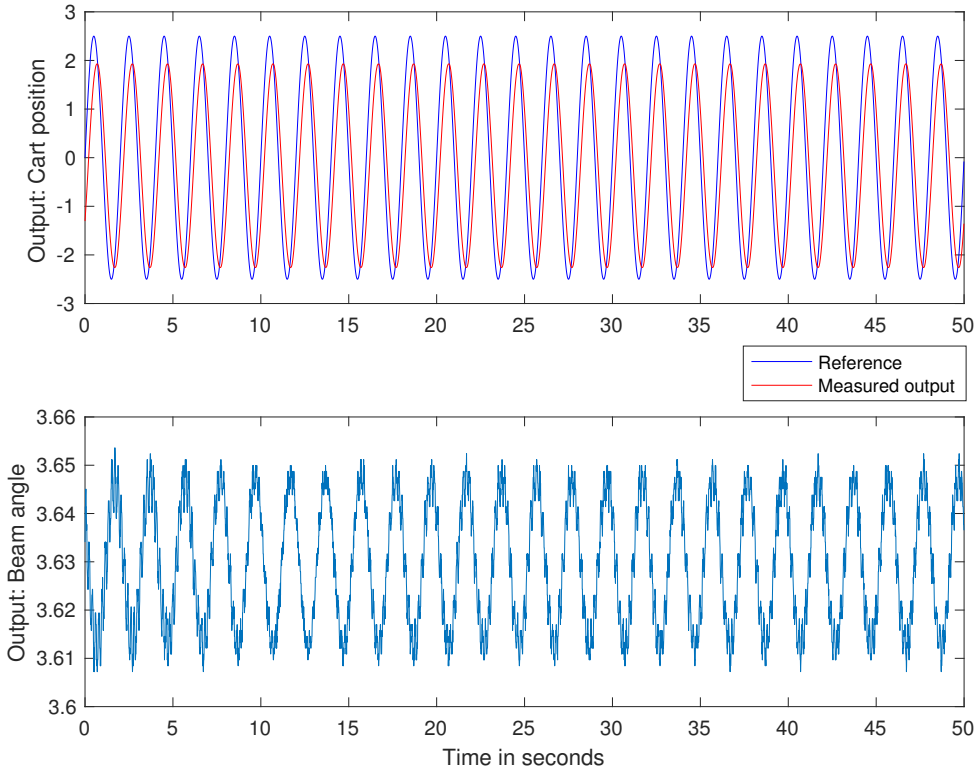


Figure 4-8: Output response of the Cart-beam system with the inner-loop controller

The system can then be treated as single input single output system with the input acting as a reference for the cart position and the output being the beam angle. This is concisely drawn in Figure 4-9. The resulting SISO system, let us call it an Inverted Beam System, is then identified and controlled using the methodology developed in this thesis.

However, the methodology cannot be used as it is because the computational time required to implement the method is still too high for the Inverted Beam system, whose sampling time is 0.01 seconds. If the system is run with sampling time higher than 0.01 seconds, the dynamics of the system are not captured. Therefore, it is necessary to reduce the computational time of the method to less than 0.01 seconds.

In order to go about this, we make use of the fact that the system is LTI. System identification is first carried out for a range of regularization parameters, of which the value resulting in the model best describing the system is chosen. Then using the estimated model, the controller is recursively implemented on the physical setup.

A varying pulse signal is given as the reference for the beam angle. Intuitively, the SPC controller (see Figure 4-9) should generate a signal (which acts as a reference for the cart position) such that the cart position moves in the form of a square wave in the direction opposite to that of the reference for the beam angle.

From the simulations that are carried out on the estimated model to select the penalty parameter, λ_p, for the controller, the value of penalty is chosen to be 5×10^{-6}. The signal

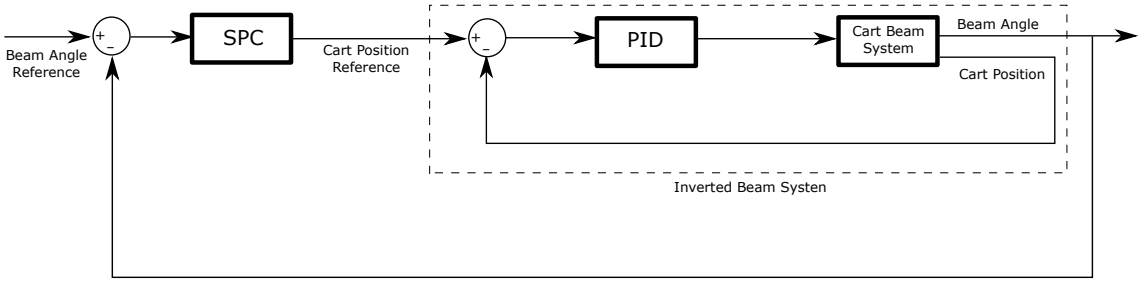


Figure 4-9: Block diagram of the Inverted beam system

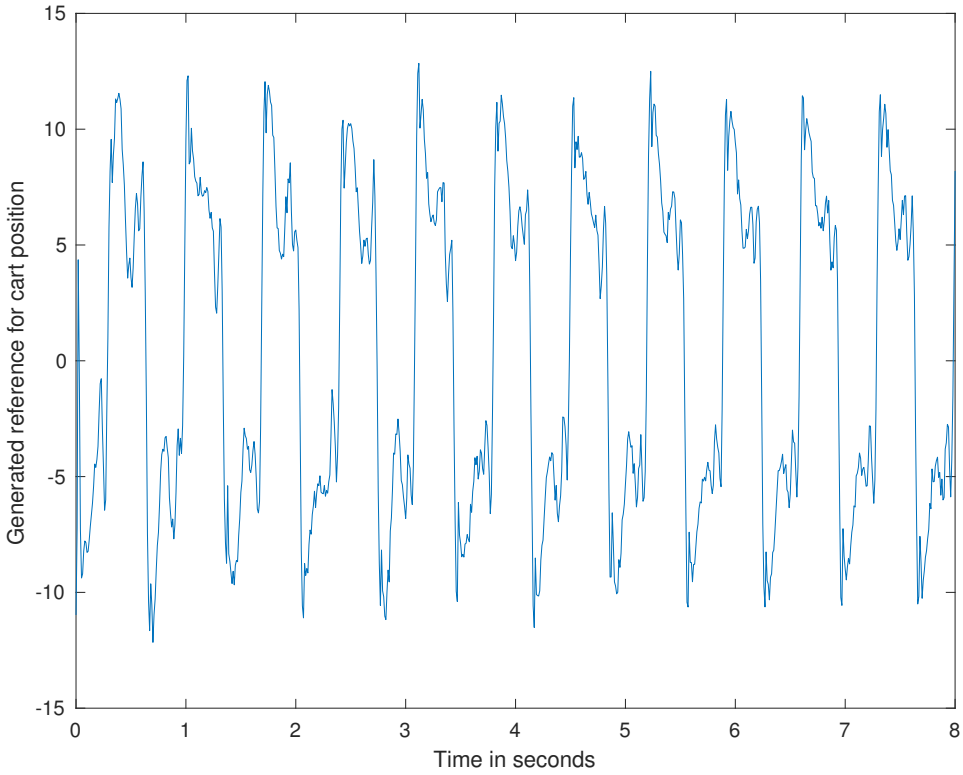


Figure 4-10: Output of the SPC controller

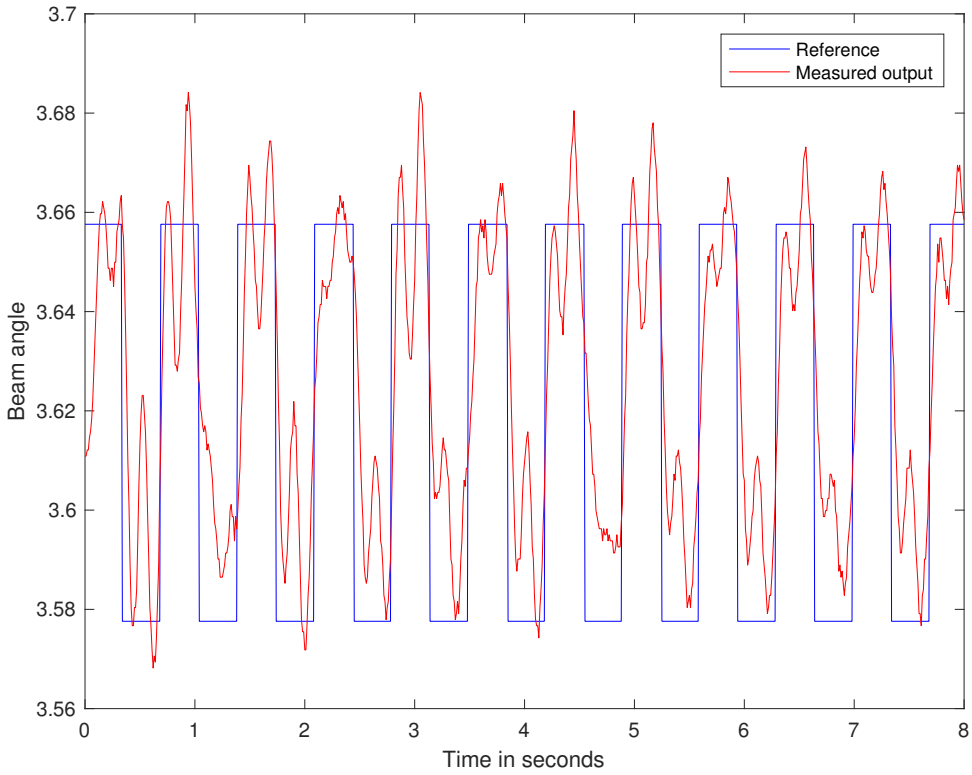


Figure 4-11: Controlled response of the beam angle

generated from the SPC controller is shown in Figure 4-10. It can be seen that just from the offline estimated model, the SPC controller is able to generate a roughly square wave.

The resulting measured response of the Inverted beam system is shown in Figure 4-11. The time taken to compute the control value for each sampling time instant is 3.22×10^{-4} seconds, which is well within our requirements.

To summarize, the methodology developed in this work was algorithmized in Section 4-1, which was then tested through implementation on different systems - LTI, LPV and a physical system. From the evaluated results, it can be concluded that the methodology proved to provide effective control for all the systems that were considered.

Chapter 5

Conclusion

The work presented in this thesis can be summarized with the conclusions drawn below.

- Nuclear norm based system identification methods are too slow to be implemented online. It was concluded that significant reductions in the computational complexity are required to allow implementations in adaptive control algorithms.
- It was concluded that when the ADMM algorithm is terminated early, there is a significant reduction in computation time with negligible loss in accuracy of identification.
- For recursive identification, it was concluded that the system information from the previous identification can be directly used to compute the intial condition for the current identification cycle. Such a choice is theoretically justified and results in significant performance gains.
- With the algorithm RN2SID, the time taken per identification was reduced from 2.98 seconds to 0.46 seconds for a particular plant model. Similar performance gains have been observed for other plants.
- The methodology to formulate a control law without explicit knowledge of system matrices and initial state estimate was presented.
- The run time of the overall algorithm – recursive system identification and control value computation – varied from a maximum of 0.7 seconds (CD-player arm) to a minimum of 0.34 seconds (Flutter model).
- The developed algorithm was able to effectively account for the changing system dynamics in the LPV system – Flutter model – and supply the appropriate control value. While the PID controller could satisfactorily control the Flutter model for constant wind speed, it failed to achieve control when the system parameters were varied.
- Although the computational speed of RN2SID was reduced, it was still greater than 0.01 seconds. Due to this limitation, it could not be implemented as a whole on the physical

system (Cart-Beam system). Instead, offline identification results of the Cart-Beam system were used for online control. This demonstrates the flexibility of the proposed algorithm for various circumstances.

Future work:

- Techniques to further reduce the computation time will be investigated in the future. A further reduction in computation time will allow the proposed algorithm to be implemented on a larger variety of systems.
- The penalty parameter λ_p in the SPC will be adaptively tuned instead of being manually tuned.
- Testing will be carried on real world LPV systems to test the capabilities of the proposed algorithm and to gain further insights on potential improvements.
- The limitations of the algorithm will be mathematically derived. The proposed algorithm will then be tested rigorously on Linear Time Varying systems and Non-Linear systems.

Bibliography

[1] Cara, E and Zuazua, E, "Control theory: History, mathematical achievements and perspectives," *Boletín de la Sociedad Española de Matemática Aplicada, 26, 79-140.*, 2003.

[2] Åström, K. J., "History of Adaptive Control," *Encyclopedia of Systems and Control*, 2014.

[3] Gevers, M, "A personal view on the development of system identification," *IEEE Control Systems, Vol 26, Issue 6, 93-105*, 2006.

[4] Telsang, B, "Literature Survey on Recursive Subspace Identification with Predictive Control, a nuclear norm approach," 2016.

[5] Qin, J, "An overview of subspace identification," *Computers and chemical engineering 30.10 (2006): 1502-1513*, 2006.

[6] Verhaegen, M and Dewilde, P, "Subspace model identification part I: The output-error state space model identification class of algorithms," *International Journal of Control, 56, 1187-1210*, 1992.

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

[8] Fazel, M, Hindi, H, and Boyd, S, "Rank Minimization and Applications in System Theory, year = 2004, journal = American Control Conference, Proceedings of the 2004 4, 3273-3278,"

[9] Fazel, M, Hindi, H, and Boyd, S, "A rank minimization heuristic with application to minimum order system approximation," *Proceedings of American Control Conference, 4734-4739*, 2001.

[10] Hansson, A, Liu, Z, and Vandenberghe, L, "Subspace system identification via weighted nuclear norm optimization," *IEEE Decision and Control, 3439 - 3444*, 2012.

[11] Verhaegen, M and Hansson, A, "N2SID: Nuclear Norm Subspace Identification," *CoRR, http://arxiv.org/abs/1401.4273*, 2015.

[12] Ljung, L, "On the use of Regularization in System Identification," *Department of Electrical Engineering, Linkoping University*, 1992.

[13] Verhaegen, M and Verdult, V, *Filtering and System Identification.* 2007.

[14] Verhaegen, M and Hansson, A, "Nuclear Norm Subspace Identification (N2SID) for short data batches," *IFAC World Congress*, 2014.

[15] Boyd, S, Parikh, N, Chu, E, Peleato, B, and Eckstein, J, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, year = 2011, journal = Foundations and Trends in Machine Learning, Vol. 3, No. 1, 1âĂŞ122,"

[16] Komodakis, N and Pesquet, J, "Playing with Duality: An Overview of Recent Primal-Dual Approaches for Solving Large-Scale Optimization Problems," *IEEE Signal Processing Magazine (Volume:32, Issue: 6)*, 2015.

[17] Boyd, S and Vandenberghe, L, *Convex Optimization.* 2004.

[18] Moor, B, "DaISy: Database for the Identification of Systems," *http://homes.esat.kuleuven.be/ smc/daisy/*, 2012.

[19] Moonen, M, Van Dooren, P, and Vandewelle, J, "A Singular Value Decomposition updating algorithm for subspace tracking," *Siam Journal on Matrix Analysis and Applications, Vol. 13, No. 4, pp. 1015-1038*, 1992.

[20] Cai, J. F and Osher, S, "Fast Singular Value Thresholding without Singular Value Decomposition," 2010.

[21] Golub, G and Van Loan, C, *Matrix Computations.* 1996.

[22] Oh, T, Matsushita, Y, Tai, Y, and Kweon, I, "Fast Randomized Singular Value Thresholding for Nuclear Norm Minimization,"

[23] Van den Boom, T and De Schutter, B, *Lecture notes Optimization in systems and control.* 2015.

[24] Favoreel, W, Moor, B, and Van Overschee, P, "Model-free subspace-based LQG-design," *Proceedings of ACC, Vol 5, 3372-3376*, 1999.

[25] Visser, M, Navalkar, S, and Van Wingerden, J, "LPV Model Identification for Flutter Prediction: A Comparison of Methods, year = 2015, journal = 1st IFAC Workshop on Linear Parameter Varying Systems LPVS âĂŞ Grenoble, France,"

Glossary

List of Acronyms

SID	Subspace IDentification
MIMO	Multi Input Multi Output
PEM	Prediction Error Methods
SISO	Single Input Single Output
BIBO	Bounded Input Bounded Output
N4SID	Numerical algorithms for Subspace State Space System Identification
MOESP	MIMO Output-Error State Space model identification
SVD	Singular Value Decomposition
SVT	Singular Value Thresholding
FRSVT	Fast Randomized Singular Value Thresholding
AIC	Akaike Information Criterion
N2SID	Nuclear Norm based Subspace Identification
RN2SID	Recursive Nuclear Norm based Subspace Identification
COD	Complete Orthogonal Decomposition
SDP	Semi-Definite Programming
ADMM	Alternating Direction Method of Multipliers
MPC	Model Predictive Control
LTI	Linear Time-Invariant
LTV	Linear Time-Varying

LPV Linear Parameter-Varying

PID Proportional Integral Derivative

VAF Variance Accounting For

SPC Subspace Predictive Control