Recursive Data-Driven Predictive Control

Verheijen, P.C.N.

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Electrical Engineering
Master Systems and Control
Control Systems Research Group

Recursive Data-Driven Predictive Control

Master Thesis

P. C. N. Verheijen
Student ID: 1391763

Committee:

Chairman:	Prof. Dr. P. M. J. Van den Hof
Member:	Dr. A. A. J. Lefeber (Erjen)
Member:	Dr. M. Lazar
Adviser:	Dr. G. Rodrigues Gonçalves da Silva

Supervisors:

Dr. M. Lazar
Dr. G. Rodrigues Gonçalves da Silva

Eindhoven, October 2021

Declaration concerning the TU/e Code of Scientific Conduct

I have read the TU/e Code of Scientific Conduct[i].

In carrying out research, design and educational activities, I shall observe the five central values of scientific integrity, namely: trustworthiness, intellectual honesty, openness, independence and societal responsibility, as well as the norms and principles which follow from them.

Date

18-10-2021

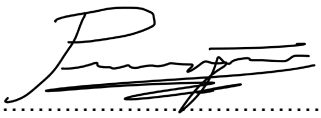
Name

Petrus Cornelis Nicolaas Verheijen

ID-number

1391763

Signature



Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Abstract

Model Predictive Control (MPC) is a control strategy where the control law optimizes the input based on the current system's response as well as the response in a foreseeable future. The advantages of this strategy is that it can respond to certain dynamics before they occur. However, to accurately predict this future response, an accurate description of the system must be provided. In the current industry, predictive controllers are tuned based on a model that represents the dynamics of the system at the time it is modelled. Classical system identification techniques in one hand, can be used to improve and/or assess model accuracy but still resort to a certain model definition. Off–line data–driven control design, on the other hand, removes the necessity of a model by directly describing the dynamics of the system as a relation of previously measured inputs and outputs. Adding to this data–driven control design, this thesis introduces a new approach to directly obtain the corresponding rate–based data–driven predictive controller of the system, which removes tracking offsets. Furthermore, while both system identification and data–driven control minimize modelling errors, the dynamics of any system are known to change over a substantial time–window. As the system dynamics change, the performance of the controller deteriorates. Regardless of how robust the controller is, at some point, it must be redesigned. An approach that circumvents the performance degradation issue through recursively updating the predictive controller is also proposed in this thesis. This is to ensure that in real–time, at any time, the dynamics of the predictive controller match those of the system it operates on. The recursive algorithm can either be combined with an off–line data–driven predictive controller which ensures the initial response is optimal, or can be used without any initial system knowledge as a plug–and–play predictive controller. The proposed methods will be tested on various simulated models as well as on a real–life Buck DC-DC Power converter.

Contents

Contents **iii**

List of Figures **iv**

1 Introduction **1**

2 Preliminaries **3**
2.1 Model Predictive Control 3
2.2 Subspace Identification of the Prediction matrices 5
2.3 Problem Statement 7

3 Integral–Action Data–Driven Predictive Control **8**
3.1 Adding Integral–Action to Data–Driven Predictive Controllers 8
3.2 Position control for a linear motor 12
3.3 Comparison of i-DPC with the state–of–art 14

4 Recursive Predictive Control **16**
4.1 Recursive Identification of the Prediction matrices 16
4.2 Recursive Predictive Control 18
4.3 Closed Loop Convergence Analysis 20
4.3.1 Using a continuously updating controller 22
4.3.2 Including a Strongly Persistently Exciting condition in predictive control problem 23
4.4 Illustrative example 26

5 Implementation on a DC-DC Buck Converter **29**
5.1 The Setup 29
5.2 Experiments 31

6 Conclusions **34**

Bibliography **35**

List of Figures

3.1 The linear motor setup 12
3.2 Linear motor example: closed–loop response comparison between SPC and i-DPC. 13
3.3 Linear motor example: estimation errors for *integral prediction matrices* computed using estimated SPC and i-DPC prediction matrices, respectively. 15
3.4 Linear motor example: prediction error comparison for different prediction horizons for *integral prediction matrices* computed using estimated SPC and i-DPC prediction matrices, respectively. 15

4.1 Control loop 21
4.2 Simulations on a linear actuator using an off–line i–DPC (blue) and the recursive i–RDPC (red) 28

5.1 Schematic of a DC-DC Voltage Buck Converter 29
5.2 The circuit in connection with the Arduino board 30
5.3 Buck converter controlled by an MPC with integral action 32
5.4 Buck converter controlled by the proposed off–line i–DPC 33
5.5 Buck converter controlled by the proposed recursive i–RDPC 33

Chapter 1

Introduction

Model predictive control (MPC) [1, 2] is a control strategy that employs a system model to predict its future behaviour. By predicting future system trajectories, MPC can anticipate for changes in the tracking reference and constraints. Computation of the MPC control law requires solving an optimization problem online, in order to minimize a cost function that penalizes the future tracking error. However, the performance of an MPC controller is highly depending on the model that is utilized for predictions, i.e. for building the so-called prediction matrices.

A prediction model is commonly obtained using either first principles or system identification techniques. Uncertainties caused by poor parameter and/or order estimations, noisy data and wear off the system over time influence the controller performance. In the last decades, eliminating the modelling step from the controller design procedure has gained increasingly interest in the field of control engineering [3]. This "Data–Driven" (DD) approach eliminates the modelling step by directly building the controller from data instead. DD predictive control design can be primarily categorized in two ways: a direct method that predicts the future states as a linear combination of previously measured input/output (I/O) data, or an indirect method which estimates the controller matrices (or prediction matrices) from input/output data. This input–output data can be either measured from an experiment session on the system prior to building the controller, or on–line while the system is controlled by the data–driven controller.

A direct approach to DD control is presented in [4] and further expanded in [5], [6] and [7]. This approach parametrizes the future output as a linear combination of past inputs and outputs, and future inputs using Hankel matrices of off–line measured input/output data and a set of coefficients that is computed on–line. This so–called Data–Enabled Predictive Control (DeePC) removes the necessity of computing any prediction model estimate. However the number of optimization variables depend on both the size of the measured input–output data matrices and the prediction horizon. Reference tracking in DeePC can be achieved by shifting the equilibrium [8] or including integral action [6]. Furthermore, [5] researched stability and robustness properties of the resulting controller, and more recently proposed a more practically feasible stability guarantee [9]. As this method maps measured data directly, it is more sensitive to noise. However [7] introduces an approach to reduce the influence of noise in the controller signal.

Indirect data–driven methods, such as Subspace Predictive Control (SPC), are presented in [10] and [11]. SPC is a control algorithm that estimates the controller matrices directly from off–line data, whilst simultaneously using on–line previously measured input/output data to estimate the current state of the system. The estimation procedure is based upon subspace identification, as presented in [12]. SPC shares the same tuning parameters as MPC, such as the output and

input weights and the (control/prediction) horizon. Therefore, tuning these parameters on a subspace predictive controller yields the same effects as doing so on an equivalent MPC. This makes SPC an intuitive and user–friendly data–driven control method. Furthermore, integral action can be implemented into the resulting controller by modifying the controller matrices after they are estimated, as shown in [11]. Instead of using on–line measured previous input/output data, [13] shows that if the states are measurable, the controller matrices can be completely computed off–line. This results in a data–driven controller that operates on–line like a model–based controller. Additionally, [14] introduced a method to validate the performance of the resulting controller.

In addition to DD controllers which are constructed using off–line data, [15, 16, 17] show that DD controllers can alternatively be constructed using on–line data. This removes the necessity of having to perform an off–line experiment and allows for automatic adjustments for time–varying systems. However, the initial small amount of present data means that initial control inputs are unreliable. Furthermore, as the data is collected in a closed–loop, it can not be assumed that the control input excites the system persistently, which is required to estimate the system dynamics accurately. To circumvent these issues, [15] suggests to run the system in open loop until a minimum amount of data has been gathered to compute the controller. A method to improve persistency of excitation is proposed in [16] which only updates the controller matrices if the newly fed input does sufficiently excite the system. Alternatively, [17] attempts to improve the excitation through the cost–function.

Therefore, we identify two main problems. First, proper reference tracking in a model–predictive controller is done by either including an input reference, or augmenting the model to include an integral action. The problem is however, that both of these methods require the model, which is unknown when building the controller directly from data. Secondly, the dynamics of all physical systems are prone to change over a substantial time–window. As off–line generated controllers (including data–driven ones) are defined from one batch of data, which represents the system at one time–instance, the resulting controller will lose performance. Because of this, it must be redesigned eventually. On–line data–driven controllers compensate for this behaviour by automatically updating their controller matrices. This additionally allows for true plug–and–play controllers. However, on–line DD controllers lack formal convergence guarantees and suffer in computational efficiency. Motivated by these problems, in this thesis the following research problems have been considered:

1. zero–offset reference tracking for a constrained output feedback data–driven predictive controllers, assuming only I/O data of the system is available,

2. performance degradation of a predictive controller in closed loop for time–varying or non–linear systems, again assuming only on–line I/O system data is available.

The thesis is structured as follows: the prerequisite information about MPC and SPC is explained in Chapter 2. Next, we introduce a novel method that allows proper reference tracking for the SPC control law in Chapter 3. The on–line data–driven controller research and the corresponding recursive data–driven predictive controller are introduced in Chapter 4. Here, we present our approach for an efficient on–line update rule and analyse convergence properties of the resulting algorithm in closed–loop. After the main part of the thesis, in Chapter 5, a real–life simulation on a DC–DC Voltage Buck–Converter will be presented with the new algorithms. The thesis is concluded in Chapter 6.

Chapter 2

Preliminaries

Model–predictive control is a control algorithm that computes the control input by minimizing a cost function over a finite horizon. This allows MPC to respond to predictable system behaviour and reference changes before they occur. In this chapter, we introduce the standard linear MPC algorithm. Furthermore, we also introduce how to build the same predictive controller from data using the indirect data–driven SPC approach.

2.1 Model Predictive Control

In this section the classical MPC algorithm will be presented. Consider a discrete, linear and time-invariant system:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \quad k \in \mathbb{N}, \end{aligned} \tag{2.1}$$

where $x(k) \in \mathbb{R}^n$ is the state, $u(k) \in \mathbb{R}^m$ is the input, $y(k) \in \mathbb{R}^q$ is the output and k denotes the discrete–time index. As we are dealing with sampled–data systems, it is a fair assumption to consider the feed through term $D = 0$. The control objective is to find an optimal control input sequence $U_k^* := U^*(x(k))$ that steers the output to the reference by minimizing the following cost function:

$$\begin{aligned} J(x(k), U_k) &:= (y_{N|k} - r_{N|k})^T Q (y_{N|k} - r_{N|k}) \\ &\quad + \sum_{i=0}^{N-1} \Big((y_{i|k} - r_{i|k})^T Q (y_{i|k} - r_{i|k}) + (u_{i|k} - r^u_{i|k})^T R (u_{i|k} - r^u_{i|k}) \Big) \\ &:= (Cx_{0|k} - r(k))^T Q (Cx_{0|k} - r(k)) + (Y_k - \mathcal{R}_k)^T \Omega (Y_k - \mathcal{R}_k) \\ &\quad + (U_k - \mathcal{R}^u_k)^T \Psi (U_k - \mathcal{R}^u_k), \end{aligned} \tag{2.2}$$

where $x_{0|k} = x(k)$, $u_{0|k} = u(k)$, $x_{i+1|k} = Ax_{i|k} + Bu_{i|k}$ and $y_{i|k} = Cx_{i|k}$. Moreover, $r_{i|k}$ and $r^u_{i|k}$ are the output reference and corresponding input reference, respectively. Furthermore,

$$\begin{aligned} Y_k &= \begin{bmatrix} y^T_{1|k} & y^T_{2|k} & \dots & y^T_{N|k} \end{bmatrix}^T, \qquad U_k = \begin{bmatrix} u^T_{0|k} & u^T_{1|k} & \dots & u^T_{N-1|k} \end{bmatrix}^T, \\ \mathcal{R}_k &= \begin{bmatrix} r^T_{1|k} & r^T_{2|k} & \dots & r^T_{N|k} \end{bmatrix}^T, \qquad \mathcal{R}^u_k = \begin{bmatrix} (r^u_{1|k})^T & (r^u_{2|k})^T & \dots & (r^u_{N|k})^T \end{bmatrix}^T, \\ \Omega &= \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & Q \end{bmatrix}, \qquad \Psi = \begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix}. \end{aligned} \tag{2.3}$$

Above Q and R are positive definite symmetric weighting matrices of appropriate dimensions, and N is the prediction horizon (we assume that a control horizon equal to the prediction horizon is used for brevity).

The future predicted outputs Y_k can be rewritten to a function of the future outputs U_k and the current state $x(k)$:

$$\begin{aligned} Y_k &= \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} U_k \\ &= \Phi x(k) + \Gamma U_k. \end{aligned} \tag{2.4}$$

Substituting equation (2.4) into (2.2) yields:

$$\begin{aligned} J(x(k), U_k) := {} & U_k^T(\Psi + \Gamma^T\Omega\Gamma)U_k + 2U_k^T\Gamma^T\Omega(\Phi x(k) - \mathcal{R}_k) + (\mathcal{R}^u_k)^T\Psi\mathcal{R}^u_k - 2U_k^T\Psi\mathcal{R}^u_k \\ & + (Cx(k) - r(k))^TQ(Cx(k) - r(k)) + (\Phi x(k) - \mathcal{R}_k)^T\Omega(\Phi x(k) - \mathcal{R}_k). \end{aligned} \tag{2.5}$$

The unconstrained MPC control law (i.e., if output and input constraints are neglected) can be computed by taking the gradient of J with respect to U_k and setting it equal to 0, i.e.:

$$\nabla_{U_k} J(x(k), U_k) = 2(\Psi + \Gamma^T\Omega\Gamma)U_k + 2\Gamma^T\Omega\left(\Phi x(k) - \mathcal{R}_k\right) - 2\Psi\mathcal{R}^u_k = 0, \tag{2.6a}$$

which yields

$$U^*_k(x(k)) = -G^{-1}(F(\Phi x(k) - \mathcal{R}_k) - 2\Psi\mathcal{R}^u_k), \tag{2.6b}$$

where

$$G = 2(\Psi + \Gamma^T\Omega\Gamma), \quad F = 2\Gamma^T\Omega. \tag{2.6c}$$

When output and input constraints are added to the minimization of J, the constrained MPC control law is computed on–line, by solving a quadratic program, i.e.,

$$\begin{aligned} \min_{U_k} \ & \frac{1}{2}U_k^TGU_k + U_k^T(F(\Phi x(k) - \mathcal{R}_k) - 2\Psi\mathcal{R}^u_k) \\ \text{subject to:} \ & \mathcal{L}U_k \le \mathbf{b} - \mathcal{M}\Phi x(k), \end{aligned} \tag{2.7}$$

where the derivation of the matrices $\mathcal{L}$, $\mathcal{M}$ and the vector $\mathbf{b}$ is illustrated next. It is worth mentioning that the terms in (2.5) that do not depend on U_k are omitted in (2.7), as they do not influence the corresponding optimum.

Consider a set of given linear constraints in the outputs and inputs:

$$\begin{aligned} u_{min} \leq u_{i|k} \leq u_{max}, \quad \forall i = 0, 1, \cdots, N-1, \\ y_{min} \leq y_{i|k} \leq y_{max}, \quad \forall i = 1, 2, \cdots, N. \end{aligned} \tag{2.8}$$

Then these constraints can be expressed as a single inequality for each i as

$$\begin{bmatrix} 0 \\ 0 \\ -I_q \\ I_q \end{bmatrix} y_{i+1|k} + \begin{bmatrix} -I_m \\ I_m \\ 0 \\ 0 \end{bmatrix} u_{i|k} \leq \begin{bmatrix} -u_{min} \\ u_{max} \\ -y_{min} \\ y_{max} \end{bmatrix}, \quad \text{yielding:} \tag{2.9}$$

$$M_{i+1} y_{i+1|k} + E_i u_{i|k} \leq b_i, \quad \forall i = 0, 1, \cdots, N-1, \quad \text{and} \quad M_N y_{N|k} \leq b_N.$$

These constraints can be aggregated as follows

$$\begin{bmatrix} M_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M_N \end{bmatrix} \begin{bmatrix} y_{1|k} \\ \vdots \\ y_{N|k} \end{bmatrix} + \begin{bmatrix} E_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E_{N-1} \end{bmatrix} \begin{bmatrix} u_{0|k} \\ \vdots \\ u_{N-1|k} \end{bmatrix} \leq \begin{bmatrix} b_0 \\ \vdots \\ b_N \end{bmatrix}, \quad \text{or} \tag{2.10}$$

$$\mathcal{M} Y_k + \mathcal{E} U_k \leq \mathbf{b}.$$

Note that the definition of $\mathcal{M}$, $\mathcal{E}$ and $\mathbf{b}$ is not unique, any alternative notation that satisfies (2.8) can be used. Substituting the future predicted outputs (2.4) into equation (2.10) results in

$$\mathcal{L} U_k \leq \mathbf{b} - \mathcal{M} \Phi x(k), \quad \text{with} \quad \mathcal{L} = (\mathcal{M}\Gamma + \mathcal{E}). \tag{2.11}$$

For both the constrained and unconstrained controllers, the MPC control law is extracted from $U_k^*(x(k))$ as follows:

$$u_k^*(x(k)) = [I_m \quad \mathbf{0}_{(N-m)\times m}] U_k^*(x(k)). \tag{2.12}$$

Note that this implementation of reference tracking by including an input reference in the cost function is often overlooked. The benefit of this method is that the original model of the system does not need to be altered, therefore no further modification are required in the constraints or applying the actual input to the system.

The steps to build a model–based predictive controller for a linear, time–invariant system are presented [1, 2]. Next, we show how to build the same controller without a model by identifying Γ and $\Phi x(k)$ directly from data.

2.2 Subspace Identification of the Prediction matrices

Instead of identifying a model of the system to build the prediction matrices, this section will introduce a method that eliminates the model by identifying the prediction matrices directly from data. Equation (2.7), and the definitions of the matrices therein, show that to compute same quadratic program without a model, only an estimate of the "forced–response" matrix Γ and the "free–response" $\Phi x(k)$ is required. To identify these prediction matrices (Γ and Φ), we start by rewriting the inputs and outputs into an ARMAX model [12]. Hereby it is assumed the I/O data is collected in an off–line, open–loop experiment where the input satisfies the persistence of excitation condition. Let the Hankel matrix of a signal be defined as

$$\mathbf{z}_k := \begin{bmatrix} z(k) \\ z(k+1) \\ \vdots \\ z(k+N-1) \end{bmatrix}, \quad \mathbf{Z}_k := \begin{bmatrix} \mathbf{z}_k & \mathbf{z}_{k+1} & \mathbf{z}_{k+2} & \dots & \mathbf{z}_{k+L} \end{bmatrix}, \tag{2.13}$$

such that the I/O data can be structured as

$$\begin{aligned} \mathbf{U}_p &= \mathbf{U}_0, \quad \mathbf{Y}_p = \mathbf{Y}_1, \\ \mathbf{U}_f &= \mathbf{U}_N, \quad \mathbf{Y}_f = \mathbf{Y}_{N+1}, \end{aligned} \tag{2.14}$$

which represent the so–called "past" and "future" data. L is the measurement length, and bounds on L will be introduced later. The distinction between the vector notation Z_k and $\mathbf{z}_k$ is that the former presents future data predicted using data up to and including k and the latter presents known measured data. Following [10, 13, 15], the relation between these past and future data can be described as

$$\begin{aligned} \mathbf{Y}_p &= \Phi X_p + \Gamma \mathbf{U}_p, \quad \mathbf{Y}_f = \Phi X_f + \Gamma \mathbf{U}_f, \\ X_f &= A^N X_p + \mathcal{C} \mathbf{U}_p, \end{aligned} \tag{2.15}$$

where $X_p = \begin{bmatrix} x(0) & x(1) & \cdots & x(L) \end{bmatrix}$, $X_f = \begin{bmatrix} x(N) & x(N+1) & \cdots & x(N+L) \end{bmatrix}$, and $\mathcal{C}$ is the extended controllability matrix $\mathcal{C} = \begin{bmatrix} A^{N-1}B & \cdots & AB & B \end{bmatrix}$. According to [18], as long as $Nq \geq n$, it is guaranteed for an observable system that there exists a matrix M such that $A^N + M\Phi = 0$. Let $W \triangleq \begin{bmatrix} \mathbf{U}_p^T & \mathbf{Y}_p^T & \mathbf{U}_f^T \end{bmatrix}^T$; this allows us to rewrite $\mathbf{Y}_f$ without state information:

$$\begin{aligned} \mathbf{Y}_f &= \Phi A^N X_p + \Phi \mathcal{C} \mathbf{U}_p + \Gamma \mathbf{U}_f \\ &= -\Phi M (\mathbf{Y}_p - \Gamma \mathbf{U}_p) + \Phi \mathcal{C} \mathbf{U}_p + \Gamma \mathbf{U}_f \\ &= \begin{bmatrix} \Phi(\mathcal{C} + M\Gamma) & -\Phi M & \Gamma \end{bmatrix} W. \end{aligned} \tag{2.16}$$

For given W and $\mathbf{Y}_f$ we can obtain the least–squares estimated prediction matrices:

$$\begin{bmatrix} P_1 & P_2 & \Gamma \end{bmatrix} = \mathbf{Y}_f W^\dagger, \tag{2.17}$$

where $[.]^\dagger$ represents the Moore-Penrose pseudo-inverse, $P_1 = \Phi(\mathcal{C} + M\Gamma)$ and $P_2 = -\Phi M$. Hereby it should be noted that $L \geq (2m+q)N$ to ensure (2.17) has a solution. [12] Points out that if the input is uncorrelated with both the measurement noise and the process noise, the influence of noise will vanish when $L \to \infty$.

Since the state is not measurable, estimating Φ directly as in [13] is not possible. Fortunately, it is not required. Instead, $\Phi x(k)$ can be estimated on–line using the previous inputs and measured outputs as

$$\begin{aligned} \Phi x(k) &= \Phi A^N x(k-N) + \Phi \mathcal{C} \mathbf{u}_{k-N} \\ &= -\Phi M \Phi x(k-N) + \Phi \mathcal{C} \mathbf{u}_{k-N} \\ &= -\Phi M \mathbf{y}_{k-N+1} + \Phi(\mathcal{C} + M\Gamma) \mathbf{u}_{k-N} \\ &= \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix}. \end{aligned} \tag{2.18}$$

The on–line estimation of $\Phi x(k)$ depends on the unknown M and the accuracy of the estimate is not guaranteed to improve for any sequence length shorter than N. The data–driven predictive controller now resorts to solving the following quadratic program:

$$\begin{aligned} \min_{U_k} \quad & \frac{1}{2} U_k^T G U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2 U_k^T \Psi \mathcal{R}_k^u \\ \text{subject to:} \quad & \mathcal{L} U_k \leq \mathbf{b} - \mathcal{M} \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix}, \end{aligned} \tag{2.19}$$

where G, F and $\mathcal{L}$ are defined using the estimated Γ.

Remark 1. *Since $\Phi x(k)$ is computed using previous I/O data, it should be noted that due to this poor initial estimate, the quadratic program may not be feasible at start. This can be circumvented by relaxing the output constraints via soft–constraints. These can be added as*

$$\begin{aligned} \min_{U_k, \varepsilon_{sc}} \quad & \frac{1}{2} U_k^T G U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2 U_k^T \Psi \mathcal{R}_k^u + \frac{1}{2} \varepsilon_{sc}^T \Sigma \varepsilon_{sc} \\ \textit{subject to:} \quad & \mathcal{L} U_k - \mathcal{D} \varepsilon_{sc} \leq \mathbf{b} - \mathcal{M} \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix}, \end{aligned} \tag{2.20}$$

where ε_{sc} is a vector containing the soft–constraints, $\mathcal{D}$ is a matrix used to only relax desired constraints with soft–constraints, where $\mathcal{D}\varepsilon_{sc} \in \mathbb{R}^{2N(q+m)}$, and Σ is a diagonal matrix used to penalize the usage of non–zero soft–constraints (where the diagonal values of Σ are usually substantially larger than those in Ω or Ψ).

Remark 2. *In contrary to the SPC algorithm presented in [10], our implementation shifts the output data by 1. As we assume the system is strictly proper, this shift allows us to*

1. *directly identify the correct prediction matrices without removing the first row manually afterwards,*
2. *predict the future outputs using the current output y_k, which results in a faster response to unpredictable dynamics.*

Furthermore, they suggested that by performing an Singular Value Decomposition (SVD) matrix reduction on P_1 and P_2 reduces noise influences in their estimations. However, for this reduction, the order of the system should be estimated. Improperly estimating this order can result in significantly worse performance.

2.3 Problem Statement

The data–driven control algorithm presented in Section 2.2 shows that a predictive controller can alternatively be defined directly from data. This removes the steps involving system identification and on–line state estimation and thereby simplifies the MPC design process. However, due to the lack of a model, zero offset in constant reference tracking problems are not guaranteed, because computing the correct input reference is not possible. Additionally, the algorithm assumes the prediction matrices are estimated from a linear time–invariant system, while in practice, this is almost never the case. While SPC is an optimal unconstrained controller, it is based on one finite data set, measured at one moment in time. This lead us to formulate the following two main problems treated in this thesis:

Problem 1: How to design a constrained output feedback data–driven predictive controller with zero offset only from I/O data and without any model knowledge of the system?

Problem 2: Using only I/O data, how to prevent performance degradation of a data–driven predictive controller when the system dynamics no longer match the estimated dynamics in the controller?

Chapter 3

Integral–Action Data–Driven Predictive Control

Embedding an integral action into a model–based predictive controller is known to improve reference tracking, and guarantee zero–offset tracking for constant set-points. Furthermore, it also removes effects from constant disturbances. Including this integral action transforms the controller into a rate–based controller, as it controls the rate of change in the input instead of the input itself. Because of this, the input reference corresponding to the output reference is no longer required.

3.1 Adding Integral–Action to Data–Driven Predictive Controllers

Rate–based controllers are to this end defined by an augmented state $x_I(k) := \begin{bmatrix}\Delta x(k)^T & y(k)^T\end{bmatrix}^T$, where $\Delta x(k) := x(k) - x(k-1)$ and an incremental input $\Delta u(k) := u(k) - u(k-1)$ [2]. With this state, we construct the augmented model as follows:

$$\begin{aligned} x_I(k+1) &= \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} x_I(k) + \begin{bmatrix} B \\ CB \end{bmatrix} \Delta u(k) \\ y(k) &= \begin{bmatrix} 0 & I \end{bmatrix} x_I(k). \end{aligned} \tag{3.1}$$

The cost function changes to:

$$\begin{aligned} J_{x,\Delta U_k} &:= (y_{N|k} - r_{N|k})^T Q (y_{N|k} - r_{N|k}) + \sum_{i=0}^{N-1} \Big((y_{i|k} - r_{i|k})^T Q (y_{i|k} - r_{i|k}) + \Delta u_{i|k}^T R \Delta u_{i|k} \Big) \\ &:= \Delta U_k^T (\Psi + \Gamma_I^T \Omega \Gamma_I) \Delta U_k + 2 \Delta U_k^T \Gamma_I^T \Omega (\Phi_I x_I(k) - \mathcal{R}_k) \\ &\quad + (y(k) - r(k))^T Q (y(k) - r(k)) + (\Phi_I x_I(k) - \mathcal{R}_k)^T \Omega (\Phi_I x_I(k) - \mathcal{R}_k), \end{aligned} \tag{3.2}$$

where $\Delta U_k = [\Delta u_{0|k}^T \dots \Delta u_{N-1|k}^T]^T$. The actual control input is obtained as $u(k) = u(k-1) + \Delta u^*(k)$. By substituting the integral model (3.1) into (2.4), we obtain

$$\Phi_I = \begin{bmatrix} CA & I \\ CA^2 + CA & I \\ \vdots & \\ \sum_{i=1}^{N} CA^i & I \end{bmatrix}, \quad \Gamma_I = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB + CB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{N-1} CA^iB & \sum_{i=0}^{N-2} CA^iB & \cdots & CB \end{bmatrix}. \tag{3.3}$$

Here, it stands clear that this augmented model does not physically exist. Therefore, it can not be directly measured. Using only input/output data measured on the physical system, how to design an output feedback data–driven predictive controller with zero tracking offset? Next, we state a result that allows us to derive these integral action prediction matrices directly from data by manipulating the input used in the experiment.

Theorem 1. *Let* $\Delta U_{i|k} = \begin{bmatrix} u_{0|k} - u(k-1) \\ u_{1|k} - u_{0|k} \\ \vdots \\ u_{i-1|k} - u_{i-2|k} \end{bmatrix}$ *and* $[\Phi]_i$, $[\Gamma]_i$ *(respectively* $[\Phi_I]_i$, $[\Gamma_I]_i$*) represent the corresponding row–blocks of* Φ, Γ *(respectively* Φ_I, Γ_I*) w.r.t. output* $y_{i|k}$*. Then it holds that*

$$y_{i|k} = [\Phi_I]_i x_I(k) + [\Gamma_I]_i \Delta U_{i|k} \tag{3.4a}$$
$$= [\Phi]_i x(k) + [\Gamma]_i U_{i|k}, \tag{3.4b}$$

for all $i = 1, \dots, N$.

Proof: Consider first the trivial case, $i = 1$, and let $y_{0|k} = CAx(k-1) + CBu(k-1)$. Then we have that

$$\begin{aligned} y_{1|k} &= \begin{bmatrix} CA & I \end{bmatrix} x_I(k) + CB\Delta U_{1|k} \\ &= CA\left(x(k) - x(k-1)\right) + y_{0|k} + CB\left(u_{0|k} - u(k-1)\right) \\ &= CAx(k) + CBu_{0|k} \\ &= [\Phi]_1 x(k) + [\Gamma]_1 U_{1|k}. \end{aligned} \tag{3.5}$$

Next, consider the case of an arbitrary $i > 1$ and assume that (3.4) holds for all $j = 1, \dots, i-1$. Then we have that

$$\begin{aligned} y_{i|k} &= \begin{bmatrix} \sum_{j=1}^{i} CA^j & I \end{bmatrix} x_I(k) + \begin{bmatrix} \sum_{j=0}^{i-1} CA^jB & \dots & CB \end{bmatrix} \begin{bmatrix} u_{0|k} - u(k-1) \\ \vdots \\ u_{i-1|k} - u_{i-2|k} \end{bmatrix} \\ &= CA^i(x(k) - x(k-1)) + \begin{bmatrix} CA^{i-1}B & CA^{i-2}B & \dots & CB \end{bmatrix} \begin{bmatrix} u_{0|k} - u(k-1) \\ \vdots \\ u_{i-1|k} - u_{i-2|k} \end{bmatrix} \\ &\quad + \begin{bmatrix} \sum_{j=1}^{i-1} CA^j & I \end{bmatrix} x_I(k) + \begin{bmatrix} \sum_{j=0}^{i-2} CA^jB & \dots & CB & 0 \end{bmatrix} \begin{bmatrix} u_{0|k} - u(k-1) \\ \vdots \\ u_{i-1|k} - u_{i-2|k} \end{bmatrix}. \end{aligned} \tag{3.6a}$$

By extracting the term with the largest power of A in each sum in $[\Phi_I]_i$ and $[\Gamma_I]_i$, the remaining terms form $[\Phi_I]_{i-1}$ and $[\Gamma_I]_{i-1}$ respectively. Furthermore, as the last entry of $[\Gamma_I]_{i-1}$ equals zero, the latter part of (3.6a) (starting at $\begin{bmatrix}\sum_{j=1}^{i-1} CA^j & I\end{bmatrix} x_I(k)$) equals $y_{i-1|k}$. Therefore,

$$\begin{aligned} y_{i|k} &= CA^i x(k) + \begin{bmatrix} CA^{i-1}B & CA^{i-2}B & \dots & CB \end{bmatrix} \begin{bmatrix} u_{0|k} \\ \vdots \\ u_{i-1|k} \end{bmatrix} \\ &\quad - CA^i x(k-1) - \begin{bmatrix} CA^{i-1}B & CA^{i-2}B & \dots & CB \end{bmatrix} \begin{bmatrix} u(k-1) \\ \vdots \\ u_{i-2|k} \end{bmatrix} + y_{i-1|k} \\ &= CA^i x(k) + \begin{bmatrix} CA^{i-1}B & CA^{i-2}B & \dots & CB \end{bmatrix} \begin{bmatrix} u_{0|k} \\ \vdots \\ u_{i-1|k} \end{bmatrix} - y_{i|k-1} + y_{i-1|k} \\ &= [\Phi]_i x(k) + [\Gamma]_i U_{i|k}. \end{aligned} \tag{3.6b}$$

Note that, we used that $y_{i|k-1} = y_{i-1|k}$, which is true assuming there is no model-plant mismatch. Hence, the equations (3.4) hold by induction for all $i = 1, \dots, N$.

The result of Theorem 1 indicates that it is possible to estimate the prediction matrices corresponding to the rate–based predictive control algorithm by manipulating the data fed to the system in the experiment. That is, the experiment is performed based on (3.4b), where the designed input is simply fed to the system; then, in the identification step, the vector $\Delta U_{i|k}$ is constructed from $u(k)$, from which the integral matrices in (3.4) can be identified using (2.17).

The implementation of the integral action DPC still requires estimation of $\Phi_I x_I(k)$. In this case, the unconstrained integral DPC law changes to

$$\begin{aligned} \Delta U_k^*(x) &= -G^{-1}F\left(\begin{bmatrix} P_{1_I} & P_{2_I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k\right) \\ &= -G^{-1}F\left(\Phi_I x_I(k) - \mathcal{R}_k\right), \end{aligned}$$

which is similar to the method used in Section 2.2. Here, P_{1_I} and P_{2_I} are obtained as (2.17), but using the differentiated input (3.4). Integral action also allows to assert directly rate–constraints on the input. As such, the constraints become:

$$\begin{aligned} M_{i+1} y_{i+1|k} + E_i u_{i|k} + K_i \Delta u_{i|k} \leq b_i, \text{ or} \\ \mathcal{M} Y_k + \mathcal{E} U_k + \mathcal{K} \Delta U_k \leq \mathbf{b}, \end{aligned} \tag{3.7}$$

where M_{i+1}, E_i and b_i are re–defined appropriately, in contrast with (2.9), to accommodate for the newly introduced rate constraints specified via K_i. Furthermore,

$$Y_k = \begin{bmatrix} P_{1_I} & P_{2_I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} + \Gamma_I \Delta U_k, \tag{3.8a}$$

$$U_k = \underbrace{\begin{bmatrix} I_m & & & 0 \\ I_m & I_m & & \\ \vdots & \vdots & \ddots & \\ I_m & I_m & I_m & I_m \end{bmatrix}}_{L_{tr}} \Delta U_k + \underbrace{\begin{bmatrix} I_m \\ I_m \\ \vdots \\ I_m \end{bmatrix}}_{\mathbb{1}_N} u(k-1). \tag{3.8b}$$

Substituting (3.8) in (3.7) yields:

$$\begin{aligned} \mathbf{b} &\geq \mathcal{M}\left(\begin{bmatrix} P_{1_I} & P_{2_I} \end{bmatrix}\begin{bmatrix} \Delta \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} + \Gamma_I \Delta U_k\right) + \mathcal{E}\left(L_{tr}\Delta U_k + \mathbb{1}_N u(k-1)\right) + \mathcal{K}\Delta U_k, \\ \left(\mathcal{M}\Gamma_I + \mathcal{E}L_{tr} + \mathcal{K}\right)\Delta U_k &\leq \mathbf{b} - \mathcal{E}\mathbb{1}_N u(k-1) - \mathcal{M}\begin{bmatrix} P_{1_I} & P_{2_I} \end{bmatrix}\begin{bmatrix} \Delta \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix}. \end{aligned} \tag{3.9}$$

Remark 3. *A solution for adding integral action to the SPC algorithm was presented in [11], which differs from our approach, as follows. Firstly, therein integral action is added only for a specific ARIMAX model structure, which allows using a discretized derivative of the output for feedback. This clearly is not extendable to general linear structures, as it will amplify the measurement noise. In contrast, we develop a data–driven analogue of the rate–based method (also called the velocity form model) [2] for adding integral action to model–based predictive control, which is applicable to general linear systems. Secondly, in [11], integral prediction matrices are derived based on algebraic manipulations on the estimated prediction matrices of SPC without integral action. In contrast, Theorem 1 shows that estimation of integral prediction matrices can be done directly from data, in the same way as estimation of normal prediction matrices, by suitable data manipulations.*

Let us now formalize the procedure to obtain the prediction matrices and implement the developed i-DPC algorithm:

Algorithm 1. i-DPC Scheme

1. Design an input signal $u(t)$ and perform an experiment on the plant to collect the output data $y(t)$.
2. Compute $\Delta u(k)$ and use it to build the matrices $\mathbf{U}_p$, $\mathbf{U}_f$ as in (2.14). Build also the output matrices $\mathbf{Y}_f$, $\mathbf{Y}_p$ from $y(k)$.
3. Apply (2.17) to obtain the respective prediction matrices $P_{1_I}, P_{2_I}, \Gamma_I$.
4. Choose the performance matrices Q, R and compute G and F in (2.6a).
5. Construct the constraint matrices as in (3.9).
6. Apply the predictive control law online via your favorite QP solver.

Remark 4. *Theorem 1 actually indicates a much broader result than just the SPC implementation. It suggests that if the difference of the input applied in the experiment is used in any identification/estimation procedure, the corresponding integral action augmented system estimate is obtained.*

Remark 5. ***i-DeePC***
Following Remark 4, it means that the rate–based DeePC controller, corresponding to the original DeePC controller [5], can be obtained as follows: Consider the following notation

$$H_L(z) := \begin{bmatrix} z_0 & z_1 & \cdots & z_{T-L} \\ z_1 & z_2 & \cdots & z_{T-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{L-1} & z_L & \cdots & z_{N-1} \end{bmatrix}.$$

Suppose the data sequences $\{u^d, y^d\}$ are obtained in an experiment session on an LTI system G, assuming u_k^d is persistently exciting upto order T. Then it can be proven that the sequence $\{\bar{u}, \bar{y}\}$ is also a trajectory of G if and only if there exists $\alpha \in \mathbb{R}^{T-L+1}$ such that [5]

$$\begin{bmatrix} H_L(u^d) \\ H_L(y^d) \end{bmatrix}\alpha = \begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix}.$$

Following Theorem 1, if we obtain Δu^d *from* u^d*, we can prove that the sequence* $\{\Delta\bar{u}, \bar{y}\}$ *is a trajectory of* G_I *if and only there exists* $\alpha \in \mathbb{R}^{T-L+1}$ *such that*

$$\begin{bmatrix} H_L(\Delta u^d) \\ H_L(y^d) \end{bmatrix} \alpha = \begin{bmatrix} \Delta\bar{u} \\ \bar{y} \end{bmatrix},$$

where G_I *is* G *augmented with an integral action.*

In the next section we provide an illustrative example of the proposed i-DPC, and provide a comparison to the alternatives.

3.2 Position control for a linear motor

Consider the simplified motion dynamics of a linear motor [19], as shown in 3.1:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -50 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} d(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t),$$

where the first state (the output) is the position and the second state is the velocity. The system is discretized with a sampling time of $T_s = 0.02$ s. The input force is constrained between -500 N $\leq u(k) \leq 500$ N, and displacement is limited between -0.165 m $\leq y(k) \leq 0.165$ m. In

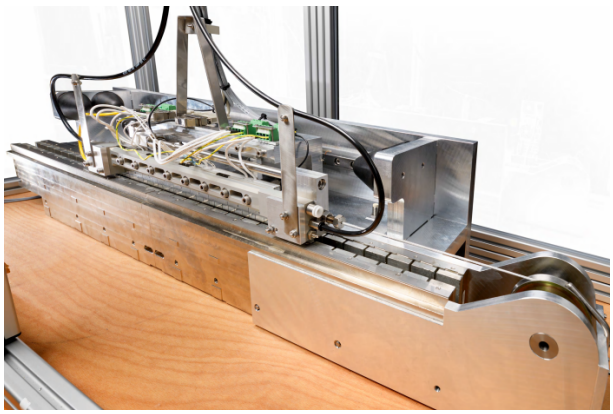


Figure 3.1: The linear motor setup

order to collect data from the system we design an experiment where the input is a PRBS signal with an amplitude of ± 80 N, clock sample 1 and length 2052 samples. The output is corrupted by a white noise sequence with variance $\sigma_e^2 = 0.005$.

We now compare the results of using a data-driven predictive controller without integral action (i.e., the SPC algorithm) with the developed i-DPC algorithm. For this case, the control performance parameters are given by $N = 15$, $Q = 6e5$, and $R = 0.05$. We then perform the estimation of the integral prediction matrices in (2.17) with $L = 2000$.

Fig. 3.2 presents the closed–loop simulation results for the developed i-DPC and the standard SPC as presented in [10]. Also, between $t = 2$ s and $t = 4$ s we apply a constant disturbance force of amplitude $d(k) = 30$ N.

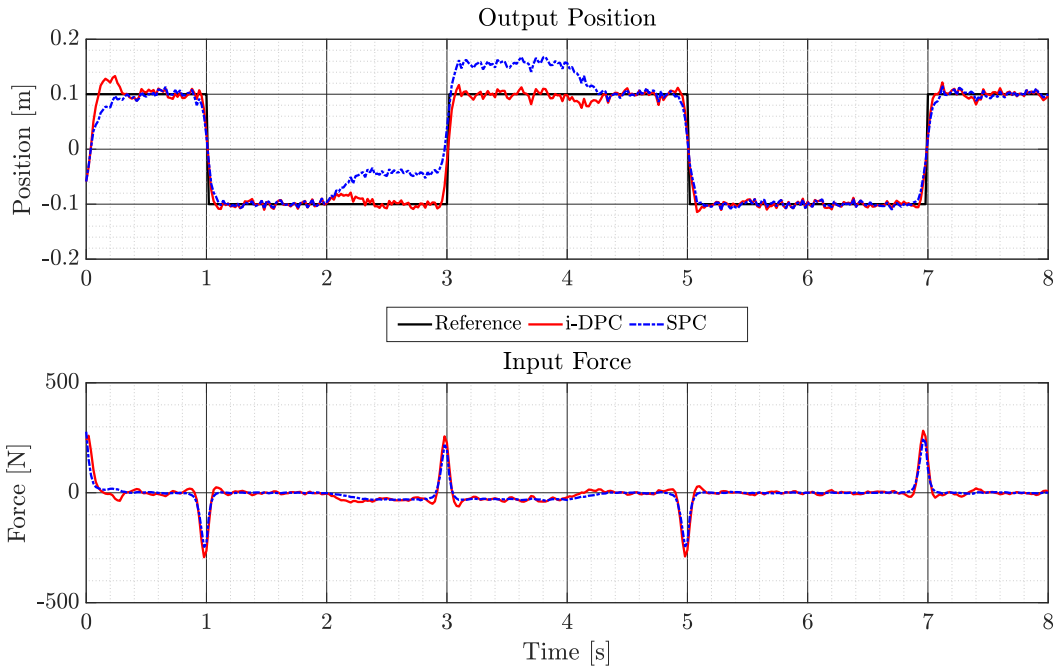


Figure 3.2: Linear motor example: closed–loop response comparison between SPC and i-DPC.

Due to the poor initial estimation, the initial response of both methods is a bit slower. Note that the proposed integral action implementation removes the effect of the disturbance completely, whereas the SPC data–driven controller without integral action fails to remove the offset.

3.3 Comparison of i-DPC with the state–of–art

Next, we compare i-DPC with the SPC algorithm with integral action added as in [11], which builds prediction matrices indirectly, by estimating non–integral prediction matrices first and then adding up individual terms in these matrices:

$$Y_k = \begin{bmatrix} y(k) \\ y(k) \\ \vdots \\ y(k) \end{bmatrix} + \begin{bmatrix} \bar{P}_{1I} & \bar{P}_{2I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}_{k-N} \\ \Delta \mathbf{y}_{k-N+1} \end{bmatrix} + \bar{\Gamma}_I \Delta U_k, \tag{3.10a}$$

where,

$$\begin{bmatrix} \bar{P}_{1I} & \bar{P}_{2I} \end{bmatrix} = L_{tr} \begin{bmatrix} P_1 & P_2 \end{bmatrix}, \quad \bar{\Gamma}_I = \Gamma L_{tr}, \tag{3.10b}$$

and the SPC cost function is modified appropriately using the new definition for Y_k. In [11], the prediction matrices are obtained using a QR-decomposition instead of the general least-squares solution. Furthermore, they use SVD to reduce the noise influence on P_1 and P_2. Only after that, the integral action is added as shown in (3.10). For the SVD we used the actual system order 2 (best case scenario). However, note that in practice, this is not known and must be estimated. The comparisons are made by taking the Frobenius norm between the estimated and model–based individual matrices $\{\hat{P}_{1_I} - P_{1_I}, \hat{P}_{2_I} - P_{2_I}, \hat{\Gamma}_I - \Gamma_I\}$ and also by computing the following prediction error criterion:

$$\varepsilon = \|\mathbf{Y}_f - \hat{\mathbf{Y}}_f\|_2. \tag{3.11}$$

The test sets for (3.11) consist of $\mathbf{Y}_f$ and $\mathbf{Y}_p$ block matrices containing $800 + N$ samples.

Figure 3.3 shows the error between the estimated matrices while Figure 3.4 portrays (3.11), both for an increasing horizon N. The Frobenius norm criterion yields comparable results, with estimation of P_{2_I} considerably improving for i-DPC with the horizon length. Most importantly, the prediction error criterion shows that the method developed in this paper for direct estimation of integral prediction matrices can significantly outperform the indirect approach of [11] for adding integral action to the SPC.

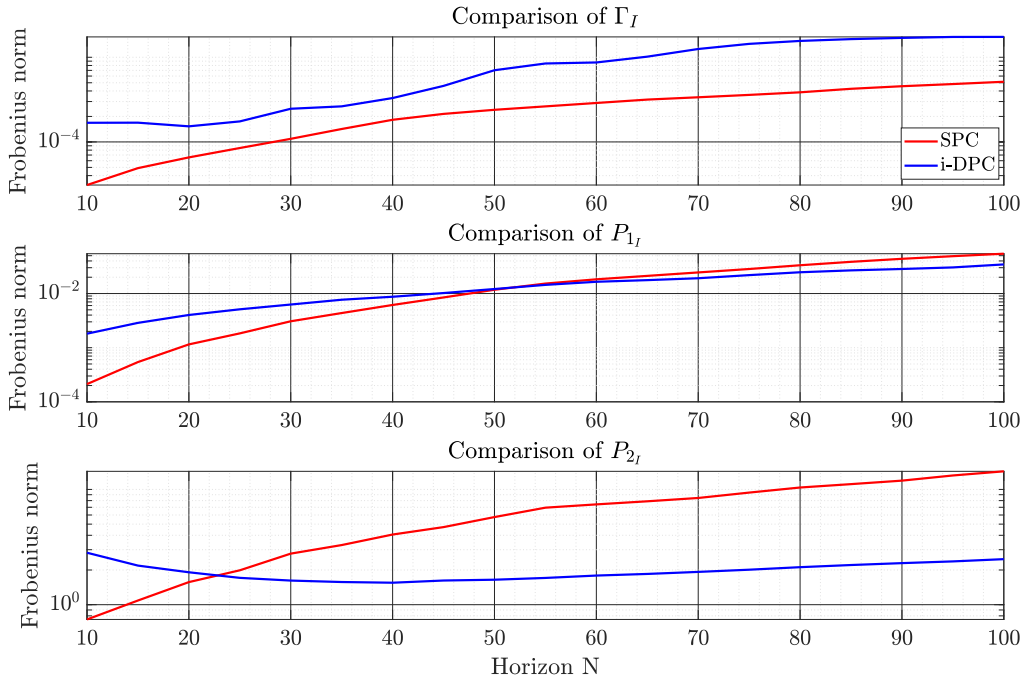


Figure 3.3: Linear motor example: estimation errors for *integral prediction matrices* computed using estimated SPC and i-DPC prediction matrices, respectively.

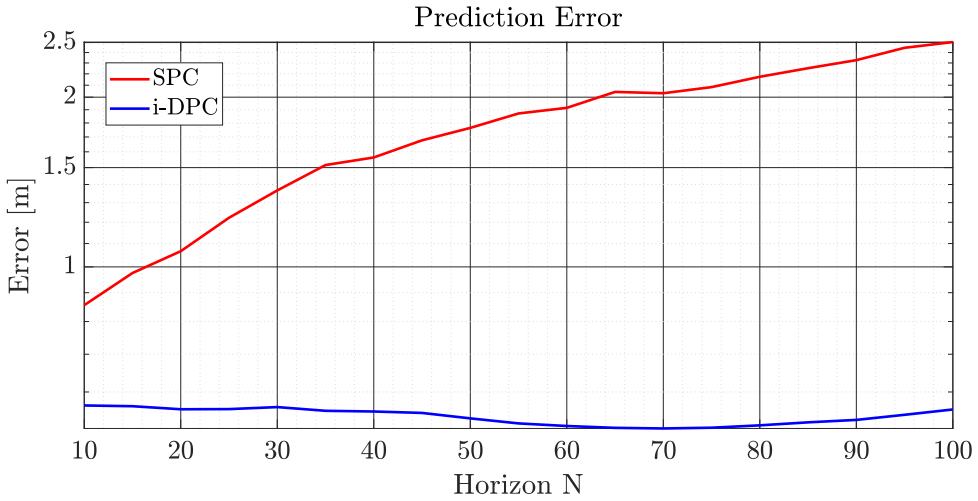


Figure 3.4: Linear motor example: prediction error comparison for different prediction horizons for *integral prediction matrices* computed using estimated SPC and i-DPC prediction matrices, respectively.

Chapter 4

Recursive Predictive Control

Off–line data–driven predictive control estimates the prediction matrices based on one set of data, measured at one time instance on the system. The benefit is that the experiment used to collect this data can be optimized for estimation. However, the resulting prediction matrices only provide an accurate prediction for the measured system dynamics, around the operating point of the system in the experiment. Therefore, time–varying or non–linear aspects of the system are not captured in the estimation. On–line data–driven control approaches eliminate this issue through continuously updating the controller with on–line measured input–output data [15, 16, 17, 20]. Theoretically, the controller should therefore always optimally control the system, at any time, around any operating point. However, it is not guaranteed that the on–line measured data is optimal for estimation. Tracking a steady reference means that the dynamics of the system are not frequently excited. Therefore, the resulting input/output data is generally very poor for estimation. This dual–problem between designing a controller that generates an input that is both suitable for reference tracking and identification can be solved in various ways. To address this problem, [20] included input disturbance with a chosen variance, but the controller only used the disturbance if the estimation error was satisfactory. Another option is to use a parameter that indicates how exciting the measured data is, and only update the measurement using exciting data samples [16]. Alternatively, [17] and [21] modify the cost function to automatically balance between estimation and tracking. [20] and [16] also use a recursive implementation of subspace identification for the prediction matrices, that uses the QR-decomposition as basis. This chapter is organized as follows. In Section 4.1 we will introduce an alternative recursive estimation procedure to estimate the prediction matrices on–line. We then use the recursive estimation algorithm with the off–line SPC control law to build a Recursive Data–Driven Predictive Controller (RDPC) in Section 4.2. Section 4.3 assesses the dual–problem of identification and control, and introduces a solution by modifying the predictive control problem.

4.1 Recursive Identification of the Prediction matrices

Recursive identification methods identify the parameter estimates computed recursively. This means that there is an previous estimate based on data up to $t-1$, and this estimate is updated using the newly measured sample. This is in contrary to off–line or batch methods, like SPC least–squares estimation introduced in Section 2.2, which identify the parameter estimate from one batch of data[22] [23]. In this section, the recursive least–squares (RLS) method is explained. RLS is the recursive implementation of the batch–wise least–squares estimation, as used in subspace identification.

Consider the following least–squares estimation as used in subspace identification (2.17):

$$\begin{bmatrix} P_1 & P_2 & \Gamma \end{bmatrix} = \mathbf{Y}_f \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}^T \left(\begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix} \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}^T \right)^{-1}. \tag{4.1}$$

Note that the pseudo–inverse $[\cdot]^\dagger$ has been replaced with the equivalent right–side expression ($ZZ^\dagger = ZZ^T(ZZ^T)^{-1}$). Let us introduce the following

$$\begin{aligned} \theta^T &= \begin{bmatrix} P_1 & P_2 & \Gamma \end{bmatrix}, \quad \phi_k = \begin{bmatrix} \mathbf{u}_{k-2N} \\ \mathbf{y}_{k-2N+1} \\ \mathbf{u}_{k-N} \end{bmatrix}, \\ W_L &= \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_L \end{bmatrix} = \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}. \end{aligned} \tag{4.2}$$

This further yields

$$\begin{aligned} \hat{\theta}_L &= \left(W_L W_L^T\right)^{-1} W_L \mathbf{Y}_f^T \\ &= \left(\sum_{t=1}^{L} \phi_t \phi_t^T \right)^{-1} \sum_{t=1}^{L} \phi_t \mathbf{y}_{t-N+1}^T. \end{aligned} \tag{4.3}$$

Equation (4.3) implies that the large Hankel matrices multiplications can be carried out as a sum of vector multiplications. In order to formulate the recursive expression, consider the new variable $\mathbf{X}_L$

$$\begin{aligned} \mathbf{X}_L^{-1} &:= \sum_{t=1}^{L} \phi_t \phi_t^T \\ &= \sum_{t=1}^{L-1} \phi_t \phi_t^T + \phi_L \phi_L^T \\ &= \mathbf{X}_{L-1}^{-1} + \phi_L \phi_L^T. \end{aligned} \tag{4.4}$$

Substituting (4.4) into (4.3) yields

$$\begin{aligned} \hat{\theta}_L &= \mathbf{X}_L \left(\sum_{t=1}^{L} \phi_t \mathbf{y}_{t-N+1}^T \right) \\ &= \mathbf{X}_L \left(\sum_{t=1}^{L-1} \phi_t \mathbf{y}_{t-N+1}^T + \phi_L \mathbf{y}_{L-N+1}^T \right), \end{aligned} \tag{4.5}$$

note that,

$$\hat{\theta}_{L-1} = \mathbf{X}_{L-1} \sum_{t=1}^{L-1} \phi_t \mathbf{y}_{t-N+1}^T,$$

therefore,

$$\begin{aligned}\hat{\theta}_L &= \mathbf{X}_L \left(\mathbf{X}_{L-1}^{-1} \hat{\theta}_{L-1} + \phi_L \mathbf{y}_{L-N+1}^T \right) \\ &= \mathbf{X}_L \left(\mathbf{X}_L^{-1} \hat{\theta}_{L-1} - \phi_L \phi_L^T \hat{\theta}_{L-1} + \phi_L \mathbf{y}_{L-N+1}^T \right) \\ &= \hat{\theta}_{L-1} + \mathbf{X}_L \phi_L \left(\mathbf{y}_{L-N+1}^T - \phi_L^T \hat{\theta}_{L-1} \right),\end{aligned} \tag{4.6}$$

which shows that to update $\hat{\theta}_L$ from $\hat{\theta}_{L-1}$, only the data sequence at time instant L is required. The final step is to remove the necessity of inverting $\mathbf{X}_L$ every iteration, which is unwanted as $\mathbf{X}_L$ is dimensionally the largest matrix in this algorithm, and computing the inverse is a time–consuming operation. This can be done by applying the Woodbury inversion lemma [24]:

$$\begin{aligned}\mathbf{X}_L &= \left(\mathbf{X}_{L-1}^{-1} + \phi_L \phi_L^T \right)^{-1} \\ &= \mathbf{X}_{L-1} - \frac{\mathbf{X}_{L-1} \phi_L \phi_L^T \mathbf{X}_{L-1}}{1 + \phi_L^T \mathbf{X}_{L-1} \phi_L}.\end{aligned} \tag{4.7}$$

Also notice that

$$\begin{aligned}\mathbf{X}_L \phi_L &= \mathbf{X}_{L-1} \phi_L - \frac{\mathbf{X}_{L-1} \phi_L \phi_L^T \mathbf{X}_{L-1} \phi_L}{1 + \phi_L^T \mathbf{X}_{L-1} \phi_L} \\ &= \frac{\mathbf{X}_{L-1} \phi_L}{1 + \phi_L^T \mathbf{X}_{L-1} \phi_L}.\end{aligned} \tag{4.8}$$

Therefore, the prediction matrices can be identified in an recursive manner through the following set of equations[22][23]:

$$\begin{aligned}e_k &= \mathbf{y}_{k-N+1} - \hat{\theta}_{k-1}^T \phi_k, \\ K_k &= \frac{\mathbf{X}_{k-1} \phi_k}{1 + \phi_k^T \mathbf{X}_{k-1} \phi_k}, \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + K_k e_k^T, \\ \mathbf{X}_k &= \mathbf{X}_{k-1} - K_k \phi_k^T \mathbf{X}_{k-1}.\end{aligned} \tag{4.9}$$

Here, $\hat{\theta}_0 = \mathbf{0}_{N(2m+q)\times Nq}$ and $\mathbf{X}_0 = \alpha I_{N(2m+q)}$, where $\alpha \gg 1$ to ensure the inverse approaches 0. Given the notation of ϕ_k in Equation (4.2), it should be noted that to initiate the recursive identification, enough data must have been gathered to completely build ϕ_k, in order to avoid leading zeros. The prediction matrices can be extracted from the estimation of θ_k, as seen in (4.2) (note the transpose). This recursive identification algorithm however still assumes the input persistently excites the system. In the next section, we continue by implementing this recursive identification algorithm within the data–driven predictive controller, and with that taking the first step into building the recursive predictive controller.

4.2 Recursive Predictive Control

The previous section introduced the adaptation of the classical recursive identification procedure as seen in [23] to identify the prediction matrices used in a data–driven predictive controller. This section we introduce a predictive controller, which updates its prediction matrices from on–line measured I/O data. As both the identification and the controller are already presented in (4.9) and (2.5), the basic recursive predictive controller without reference tracking can be formalized in the following algorithm

Algorithm 2. The RDPC scheme

1. Initialize the prediction matrix $\hat{\theta}_0$ and $\mathbf{X}_0$, weight matrices Ω and Ψ as well the constraint matrices $\mathcal{E}$ and $\mathcal{M}$,
2. Measure y_k and use that together with previous outputs and inputs to build ϕ_k and $\mathbf{y}_{k-N+1}$ as in (4.2),
3. Update the prediction matrix $\hat{\theta}_k$ as shown in (4.9),
4. Extract P_1, P_2 and Γ from $\hat{\theta}_k$ and use those to update the controller matrices G, F and $\mathcal{L}$,
5. Solve the quadratic program with your favourite QP solver,
6. Apply the first input to the system and repeat from step 2.

Theorem 1 states the relation that if the difference of the input is used in the estimation instead of the actual input applied to the system, the estimation will yield the system estimate augmented with an integral action. Hence, we can design the corresponding integral action recursive data–driven predictive controller (i-RDPC) as follows:

Algorithm 3. The i-RDPC scheme

1. Initialize the prediction matrix $[\hat{\theta}_I]_0$ and $\mathbf{X}_0$, weight matrices Ω and Ψ as well the constraint matrices $\mathcal{E}$ and $\mathcal{M}$,
2. Measure y_k and build $[\phi_I]_k = \begin{bmatrix}\Delta\mathbf{u}_{k-2N}^T & \mathbf{y}_{k-2N+1}^T & \Delta\mathbf{u}_{k-N}^T\end{bmatrix}^T$ and $\mathbf{y}_{k-N+1}$ as in (4.2),
3. Use $[\phi_I]_k$ to update the prediction matrix $[\hat{\theta}_I]_k$ as shown in (4.9),
4. Extract P_{1_I}, P_{2_I} and Γ_I from $[\hat{\theta}_I]_k$ and use those to update the controller matrices G, F and $\mathcal{L}$,
5. Compute ΔU_k^* by solving the quadratic program with your favourite QP solver,
6. Apply $u_k = u_{k-1} + \Delta u_k^*$ to the system and repeat from step 2.

Before applying either of the algorithms above, one must properly define the initial behaviour of the control loop. The loop can be initialized in two ways:

- **With a warm start**
  The recursive least–squares algorithm can be initialized using an off–line generated data–set, as how one would build the off–line SPC controller. Using this data, $\mathbf{X}_0$ can be correctly initiated as:
  $$\mathbf{X}_0 = \left(W_L W_L^T\right)^{-1}.$$
  Alternatively, using the definition of the prediction matrices, see Equation (2.16), a model–estimate of the system can be used to initialize $\hat{\theta}_0$. A correct initialization of $\mathbf{X}_0$ cannot be obtained, but α can be decreased based on the confidence in the model. As the controller can provide an initial prediction, the constraints can be satisfied and therefore no additional steps are required. However, the recursive estimation should only initiate once enough samples have been measured to construct ϕ_k with system data.

- **As a blank controller**
  In this scenario, the prediction matrices are initialized without any prior system knowledge. As such, two problems arise. First, the controller cannot guarantee the output constraints are satisfied. Secondly, the initial control input is zero. The second is a problem because the initial $N(2m+q)$ updates of the recursive estimation completely fit the current ϕ_k to the current $\mathbf{y}_{k-N+1}$. If the inputs in that ϕ_k are zero, it will fail to provide any estimates for P_1 and Γ and the controller cannot control the system. To circumvent this issue, the

system should be controlled in open loop until at least $2N$ samples have been collected, using any exciting, non-zero input. To circumvent the first problem, the system should be controlled in open loop until the estimate of $\hat{\theta}_k$ can be used to provide a proper prediction, which requires another $2N$ samples.

Even though the prediction matrices are updated continuously, in its current algorithm, the prediction matrices represent the system from the moment the controller starts. This means that if the system changes in this time–window, the prediction matrices will require a substantial amount of new exciting data to correctly "un-fit" the older system dynamics. To increase the speed to which the prediction changes to the new dynamics, a forgetting factor can be included [22]. Forgetting factors are factors that decrease the influence of older data over more recently measured data. The following two approaches are presented in [22]:

$$\begin{aligned} e_k &= \mathbf{y}_{k-N+1} - \hat{\theta}_{k-1}^T \phi_k, \\ K_k &= \frac{\mathbf{X}_{k-1}\phi_k}{\lambda + \phi_k^T \mathbf{X}_{k-1}\phi_k}, \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + K_k e_k^T, \\ \mathbf{X}_k &= \left(\mathbf{X}_{k-1} - K_k \phi_k^T \mathbf{X}_{k-1}\right)\frac{1}{\lambda}, \end{aligned} \tag{4.10}$$

where the forgetting factor λ is usually chosen close to 1 (for example 0.99 or 0.96). Smaller values for λ will reduce influence of older data faster but may jeopardize the current estimation. Due to this notation, λ has an exponential effect on $\mathbf{X}_k$. The second approach has a linear effect on $\mathbf{X}_k$:

$$\begin{aligned} e_k &= \mathbf{y}_{k-N+1} - \hat{\theta}_{k-1}^T \phi_k, \\ K_k &= \frac{\mathbf{X}_{k-1}\phi_k}{1 + \phi_k^T \mathbf{X}_{k-1}\phi_k}, \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + K_k e_k^T, \\ \mathbf{X}_k &= \mathbf{X}_{k-1} - K_k \phi_k^T \mathbf{X}_{k-1} + R_1, \end{aligned} \tag{4.11}$$

where the forgetting factor $R_1 \succ 0$ ensures the variance always stays above R_1 (here, $Z \succ 0$ and $Z \succeq 0$ denote the matrix Z is positive definite and positive semi-definite, respectively). In contrary to λ, R_1 allows for more freedom in its construction, thus bounding variances for some parameters less than others. Furthermore, due to the linear behaviour, it is significantly less aggressive, which is beneficial for an on–line identification.

Here the basic recursive predictive control algorithm is presented. While some initial problems are discussed, it is still not guaranteed whether the identification will converge to the actual prediction matrices. The next section will assess closed–loop convergence properties and introduce methods to improve the convergence of the estimation to the actual prediction matrices.

4.3 Closed Loop Convergence Analysis

Just as subspace identification, RLS guarantees convergence if the input is uncorrelated with the noise and is persistently exciting. This raises the question arises whether this identification works in a closed–loop presentation too. As the recursive identification is directly expressed from the standard, batch–wise, subspace identification of the prediction matrices, this section will use that notation for simplicity.

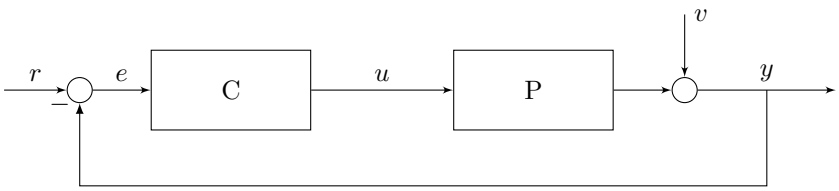


Figure 4.1: Control loop

Consider the closed–loop representation as shown in Figure 4.1. Given the plant model P_1, P_2 and Γ, and the controller matrices C_1, C_2 and C_3, the closed–loop dynamics can be expressed as:

$$\begin{aligned}\mathbf{Y}_f &= P_1\mathbf{U}_p + P_2\mathbf{Y}_p + \Gamma\mathbf{U}_f - P_2\mathbf{V}_p + \mathbf{V}_f,\\ \mathbf{U}_f &= C_1(\mathbf{R}_p - \mathbf{Y}_p) + C_2\mathbf{U}_p + C_3(\mathbf{R}_f - \mathbf{Y}_f),\end{aligned} \tag{4.12}$$

where $\mathbf{V}_p$ and $\mathbf{V}_f$ are Hankel matrices of the measurement noise v. $\mathbf{R}_p$ and $\mathbf{R}_f$ are the Hankel matrices containing the reference. For clarity of this derivation, only the estimation of Γ is considered. In an open–loop case, Γ can be estimated using the following projection [12]:

$$\begin{aligned}\mathbf{W}_p &= \begin{bmatrix}\mathbf{U}_p^T & \mathbf{Y}_p^T\end{bmatrix}^T,\\ \hat{\Gamma} &= \mathbf{Y}_f\mathbf{W}_p^{\perp}\left[\mathbf{U}_f\mathbf{W}_p^{\perp}\right]^{\dagger},\end{aligned} \tag{4.13}$$

where $[.]^{\perp}$ represents the orthogonal null projection, i.e. $Z^{\perp} := I - Z^T(ZZ^T)^{-1}Z$. If the noise v is uncorrelated with the input u (as in the case of open–loop experiments), the noise influence of the estimation of Γ vanishes as $L \to \infty$. However, when u is computed by a controller in closed–loop, u will be correlated with the noise v, thus the convergence guarantee no longer hold. Instead, it will approximate to the following:

$$\begin{aligned}\mathbf{Y}_f\mathbf{W}_p^{\perp} &= \Gamma\mathbf{U}_f\mathbf{W}_p^{\perp} - P_2\mathbf{V}_p\mathbf{W}_p^{\perp} + \mathbf{V}_f\mathbf{W}_p^{\perp},\\ \mathbf{U}_f\mathbf{W}_p^{\perp} &= C_1\mathbf{R}_p\mathbf{W}_p^{\perp} + C_3\mathbf{R}_f\mathbf{W}_p^{\perp} - C_3\mathbf{Y}_f\mathbf{W}_p^{\perp}.\end{aligned} \tag{4.14}$$

To remove the direct relation of $\mathbf{U}_f$ in $\mathbf{Y}_f$ and vise-versa, substitute the equations in (4.14) into each other:

$$\begin{aligned}\mathbf{Y}_f\mathbf{W}_p^{\perp} &= \Gamma(C_1\mathbf{R}_p + C_3\mathbf{R}_f - C_3\mathbf{Y}_f)\mathbf{W}_p^{\perp} - P_2\mathbf{V}_p\mathbf{W}_p^{\perp} + \mathbf{V}_f\mathbf{W}_p^{\perp},\\ \mathbf{U}_f\mathbf{W}_p^{\perp} &= C_1\mathbf{R}_p\mathbf{W}_p^{\perp} + C_3\mathbf{R}_f\mathbf{W}_p^{\perp} - C_3(\Gamma\mathbf{U}_f - P_2\mathbf{V}_p + \mathbf{V}_f)\mathbf{W}_p^{\perp},\end{aligned} \tag{4.15}$$

which rewrites to

$$\begin{aligned}\mathbf{Y}_f\mathbf{W}_p^{\perp} &= (I + \Gamma C_3)^{-1}\left[\Gamma C_1\mathbf{R}_p + \Gamma C_3\mathbf{R}_f - P_2\mathbf{V}_p + \mathbf{V}_f\right]\mathbf{W}_p^{\perp},\\ \mathbf{U}_f\mathbf{W}_p^{\perp} &= (I + C_3\Gamma)^{-1}\left[C_1\mathbf{R}_p + C_3\mathbf{R}_f + C_3P_2\mathbf{V}_p - C_3\mathbf{V}_f\right]\mathbf{W}_p^{\perp}.\end{aligned} \tag{4.16}$$

As such, the approximate estimate of $\hat{\Gamma}^{cl}$ is computed as:

$$\begin{aligned}\hat{\Gamma}^{cl} &= \mathbf{Y}_f\mathbf{W}_p^{\perp}\left[\mathbf{U}_f\mathbf{W}_p^{\perp}\right]^{\dagger}\\ &= (I + \Gamma C_3)^{-1}\left[\Gamma C_1\mathbf{R}_p + \Gamma C_3\mathbf{R}_f - P_2\mathbf{V}_p + \mathbf{V}_f\right]\mathbf{W}_p^{\perp}\\ &\quad\times\left(\left[C_1\mathbf{R}_p + C_3\mathbf{R}_f + C_3P_2\mathbf{V}_p - C_3\mathbf{V}_f\right]\mathbf{W}_p^{\perp}\right)^{\dagger}(I + C_3\Gamma).\end{aligned} \tag{4.17}$$

From (4.17) two limit cases can be identified.

$$\begin{aligned} &1) \text{ if } v_k \to 0: && \hat{\Gamma}^{cl} \to \Gamma, \\ &2) \text{ if } (\mathbf{R}_f + \mathbf{R}_p)\mathbf{W}_p^{\perp} \to 0 \text{ (insufficient excitation)}: && \hat{\Gamma}^{cl} \to -C_3^{-1}. \end{aligned} \tag{4.18}$$

These two cases means that $\hat{\Gamma}^{cl}$ will either converge to the actual system or to the inverse of the controller, which are equivalent to the two limit cases in the classical system identification theory, although proven in the frequency domain [22]. The identifications of P_1 and P_2 will converge following the same limit cases, however they approach $-C_3^{-1}C_1$ and $-C_3^{-1}C_2$ respectively if the reference is insufficiently exciting. In an on–line scenario, the reference is generally not a persistently exciting signal. To circumvent this issue, we introduce tools to improve the accuracy of the prediction matrices.

Remark 6. *In this derivation, process noise is omitted. However, as the state is expressed as a linear combination of previous inputs and outputs, the effects of process noise can be directly expressed through $-P_2\mathbf{V}_p$. Even though that will affect $\mathbf{Y}_f$, it has no effect on the outcome of this analysis.*

Remark 7. *The result from Equation (4.17) suggest that if a closed–loop experiment is required to collect data for the off–line SPC controller, only a persistently exciting reference is required.*

4.3.1 Using a continuously updating controller

According to [25], the estimation of $\hat{\theta}$ will converge to the plant if one of the following situations apply:

1. A sufficiently exciting reference signal is present;
2. C is a controller of sufficiently high order;
3. C is a controller that switches between several settings during the experiment.

For the last situation, these include non–linear and/or time–varying controllers. Furthermore, these situations only apply if the actual system can be described by the chosen model structure. Along with that, the estimation may not be consistent if the noise model is misspecified. To reflect this theory to the proposed controller, the following can be stated:

1. No assumptions can be made on the reference;
2. As the controller is computed using the last N inputs and outputs, the controller has an order of N. However, as it at the same time, controls the next N inputs, the order of the controller is insufficient;
3. Due to the inclusion of constraints, the predictive controller becomes a non-linear controller. Furthermore, as the controller is continuously updated using the recursive estimation, it is time–varying as well;
4. We assume the system can be described in the model structure, however as no noise model is specified, it is unsure if consistency can be achieved.

This suggests that $\hat{\theta}$ converges to the plant dynamics as long as the controller sufficiently changes. However, that immediately imposes a conundrum. If $\hat{\theta}$ converges, the controller converges too, therefore the controller no longer changes, and thus the convergence of $\hat{\theta}$ is no longer guaranteed.

The result of this is that $\hat{\theta}$ will converge to the system dynamics initially, but once the output settles to the reference, the input generated by the controller will not sufficiently excite the system. As $\hat{\theta}$ continuously updates and depending on how low the covariance matrix $\mathbf{X}_k$ is, the estimation might slowly diverge, provided that the corresponding controller still tracks the reference. This results in loss of prediction performance and constraint assessment, two key features of a predictive controller.

4.3.2 Including a Strongly Persistently Exciting condition in predictive control problem

The previous subsection elaborated on the free benefits of using a recursively updating controller. However, as it also showed that this may not be sufficient, a stronger guarantee is required. Hence, we will introduce a few methods in order to ensure a strongly persistently exciting (SPE) input is also applied to the system, while still solving the cost function optimization. The signal u is said to be persistently exciting up to order n if the matrix [23][26]

$$\bar{R}_n = \begin{bmatrix} R_u(0) & R_u(1) & \dots & R_u(n-1) \\ R_u(1) & R_u(0) & \dots & R_u(n-2) \\ \vdots & \ddots & \ddots & \vdots \\ R_u(n-1) & \dots & R_u(1) & R_u(0) \end{bmatrix}, \tag{4.19}$$

is non-singular, with $R_u(i) = \bar{E}u(t)u(t-i)$ being the correlation function at lag i. The signal is strongly persistently exciting if [21]

$$\rho_{\text{lb}} \preceq \bar{R}_n \preceq \rho_{\text{ub}}, \quad \rho_{\text{lb}} \succeq 0, \tag{4.20}$$

where $Z \preceq X$ means that $X - Z$ is positive semi-definite. In practice, $\bar{R}_n$ can be estimated for u as

$$\bar{R}_N = \frac{1}{L} \sum_{i=N}^{L} \mathbf{u}_{k-i} \mathbf{u}_{k-i}^T, \tag{4.21}$$

or, in a matrix form as

$$\begin{aligned} \mathbf{U}_{Hp} &= \begin{bmatrix} \mathbf{u}_{k-N-L} & \mathbf{u}_{k-L-N+1} & \dots & \mathbf{u}_{k-N} \end{bmatrix}, \\ \bar{R}_N &= \mathbf{U}_{Hp} \mathbf{U}_{Hp}^T. \end{aligned} \tag{4.22}$$

In (4.21), L represents an SPE horizon and by imposing $L \geq N$, one guarantees that $\bar{R}_N$ is singular if u is persistently exciting over this horizon. By keeping L small, one ensures that the controller is continuously computing an SPE input. Additionally, as this excites the output in real–time, it also ensures the estimation is closer to the actual system dynamics. Furthermore, as the cost function computes the new inputs, the SPE condition should apply to these as well. In order to write this condition in a way that we can use to solve the predictive controller optimization, consider first the shifted matrix:

$$\begin{aligned} \mathbf{U}_H &= \left[\begin{array}{ccc|cccc} u(k-N-L) & \dots & u(k-N) & u(k-N+1) & u(k-N+2) & \dots & u_{0|k} \\ u(k-N-L+1) & \ddots & u(k-N+1) & u(k-N+2) & \ddots & \ddots & u_{1|k} \\ \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \vdots \\ u(k-L-1) & \dots & u(k-1) & u_{0|k} & u_{1|k} & \dots & u_{N-1|k} \end{array}\right] \\ &= \begin{bmatrix} \mathbf{U}_{Hp} & \mathbf{U}_{Hf} \end{bmatrix}, \end{aligned} \tag{4.23}$$

where $L \geq 0$. This allows us to write the following optimization problem:

$$\begin{aligned} \min_{U_k} \quad & \frac{1}{2} U_k^T G U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2 U_k^T \Psi \mathcal{R}_k^u \\ \text{subject to:} \quad & \mathcal{L} U_k \leq \mathbf{b} - \mathcal{M} \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} \\ & - \operatorname{eig}(\mathbf{U}_H \mathbf{U}_H^T) \leq -\rho_{\text{lb}} \mathbb{1}_N. \end{aligned} \tag{4.24}$$

It should be noted that the upper bound of the eigenvalues is not imposed, as it is assumed the controller seeks to minimize U_k. With the inclusion of the constraint on the eigenvalues, this becomes a non-linear problem. Non-linear problems cannot be solved with a standard QP–solver and require alternative, less efficient solvers. Given that this solver has to compute the eigenvalues of a matrix possibly multiple times, it is not expected this control law will run efficiently. Because of this, we introduce a few alternative implementations that can be computed faster.

- **The Determinant**
  The determinant of a matrix is equal to the product of the eigenvalues. Furthermore, for any non-singular matrix the determinant is non-zero. Asserting a lower bound on the determinant therefore asserts a non-zero bound on the eigenvalues. However, it does not assert an exact lower bound on the eigenvalues in general. Furthermore, to find a suitable ρ for this method, one should consider that the value of the determinant scales exponentially with the prediction horizon. This changes the cost–function to:

$$\begin{aligned} \min_{U_k} \quad & \frac{1}{2} U_k^T G U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2 U_k^T \Psi \mathcal{R}_k^u \\ \text{subject to:} \quad & \mathcal{L} U_k \leq \mathbf{b} - \mathcal{M} \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} \\ & - \det(\mathbf{U}_H \mathbf{U}_H^T) \leq -\rho_{\text{lb}}. \end{aligned} \tag{4.25}$$

  Do note that as $\mathbf{U}_H \mathbf{U}_H^T$ is positive semi-definite, the determinant of this matrix must be greater than or equal to zero.

- **The Trace**
  At the other end of the determinant, is the trace. The trace is the sum of the diagonal values, and the sum of the eigenvalues as well. Computing the trace results in a quadratic equation with respect to U_k. This does still means standard QP solvers cannot be used, but Quadratically Constrained Quadratic Program (QCQP) solvers are widely available, and rather efficient. If the gradient of the constraints with respect to U_k can be provided, the QCQP problem can run faster and give more reliable results [27]. Fortunately, the gradient of the trace with respect to U_k can be easily computed. Suppose $\mathbf{U}_H^i$ is the ith row of $\mathbf{U}_H$. Then the values on the diagonal of $\mathbf{U}_H \mathbf{U}_H^T$ can be expressed as

$$[\mathbf{U}_H \mathbf{U}_H^T]_{ii} = \|\mathbf{U}_H^i\|_2.$$

  Given that each $\mathbf{U}_H^i$ contains the first i values of U_k, $\nabla_{U_k} \|\mathbf{U}_H^i\|_2$ results in

$$\nabla_{U_k} \|\mathbf{U}_H^i\|_2 = 2 \begin{bmatrix} \mathbb{1}_{1\times i} & \mathbf{0}_{1\times N-i} \end{bmatrix} U_k,$$

  and the gradient of the trace of $\mathbf{U}_H \mathbf{U}_H^T$ with respect to U_k can be computed by summing the gradients for each value on the diagonal

$$\nabla_{U_k} \operatorname{tr}(\mathbf{U}_H \mathbf{U}_H^T) = 2 \begin{bmatrix} N & N-1 & \dots & 1 \end{bmatrix} U_k.$$

It must be noted that the value of the trace is generally dominated by the larger eigenvalues, and the value of the trace cannot determine if $\mathbf{U}_H\mathbf{U}_H^T$ is singular. To prevent the trace from considering non-zero constant inputs as "persistently exciting", the average of the signal u can be removed as:

$$\bar{R}_N = \sum_{i=N}^{L} (\mathbf{u}_{k-i} - \mathbb{1}_{N\times 1}\mu(\mathbf{u}_{k-i}))(\mathbf{u}_{k-i} - \mathbb{1}_{N\times 1}\mu(\mathbf{u}_{k-i}))^T, \tag{4.26}$$

where $\mu(\mathbf{z})$ is the average of the sequence $\mathbf{z}$ (which is equivalent to the expected value). This can be computed in a matrix form as

$$\begin{aligned} \mu(\mathbf{u}_{k-i}) &= \frac{1}{N}\mathbb{1}_{1\times N}\mathbf{u}_{k-i} \\ (\mathbf{u}_{k-i} - \mathbb{1}_{N\times 1}\mu(\mathbf{u}_{k-i})) &= (I - \frac{1}{N}\mathbb{1}_{N\times N})\mathbf{u}_{k-i} = T_\mu \mathbf{u}_{k-i} \\ \bar{R}_N &= T_\mu \mathbf{U}_H \mathbf{U}_H^T T_\mu^T. \end{aligned} \tag{4.27}$$

As T_μ is only a linear transformation, the gradient is multiplied with this matrix too. Therefore it maintains the computational efficiency. This changes the cost–function to:

$$\begin{aligned} \min_{U_k} \quad & \frac{1}{2}U_k^T G U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix}\begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2U_k^T \Psi \mathcal{R}_k^u \\ \text{subject to:} \quad & \mathcal{L}U_k \leq \mathbf{b} - \mathcal{M}\begin{bmatrix} P_1 & P_2 \end{bmatrix}\begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} \\ & - \mathrm{Tr}(T_\mu \mathbf{U}_H \mathbf{U}_H^T T_\mu^T) \leq -\rho_{\mathrm{lb}}. \end{aligned} \tag{4.28}$$

- **Excite in the least exciting direction**
  Instead of improving the SPE by asserting bounds on the eigenvalues of $\mathbf{U}_H\mathbf{U}_H^T$, [17] introduced a method that steers U_k to the least exciting direction. Consider the following SVD:

$$\begin{aligned} \mathbf{U}_{Hp} &= U_S \Sigma_S V_S^T \\ &= \begin{bmatrix} c_{s_0} & c_{s_1} & \dots & c_{s_N} \end{bmatrix} \begin{bmatrix} \sigma_{s_0} & & & 0 & \dots & 0 \\ & \ddots & & \vdots & & \vdots \\ & & \sigma_{s_N} & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} v_{s_0} \\ \vdots \\ v_{s_L} \end{bmatrix}^T, \end{aligned} \tag{4.29}$$

  which shows that the matrix can be expressed as a set of orthonormal directions with each a corresponding magnitude. Note that only the previous inputs are used to compute this SVD ($\mathbf{U}_{Hp}$). Here, the direction with magnitude σ_{s_0} is excited the most, while the direction with magnitude σ_{s_N} is the least excited direction (or, if $\bar{R}_n$ is singular, not excited at all). With respect to the actual system, consider that the direction of N inputs represents a combination of individual inputs. The least exciting direction expresses a combination of inputs that has not been applied to the system in the given horizon. This results to minimizing the following criterion:

$$\min_{U_k} (\rho c_{s_N} - U_k)^T (\rho c_{s_N} - U_k), \tag{4.30}$$

  where ρ denotes the excitation level, which is bounded as $0 \leq \rho \leq \max \|U_k\|$, where $\max \|U_k\|$ can be computed from the given input constraints. Including (4.30) into (2.19)

yields:

$$\begin{aligned} \min_{U_k} \quad & \frac{1}{2}U_k^T(G+2)U_k + U_k^T F(\begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix} - \mathcal{R}_k) - 2U_k^T(\Psi\mathcal{R}_k^u + \rho c_{s_N}) \\ \text{subject to:} \quad & \mathcal{L}U_k \leq \mathbf{b} - \mathcal{M} \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_{k-N} \\ \mathbf{y}_{k-N+1} \end{bmatrix}. \end{aligned} \tag{4.31}$$

The benefit of this method, is that the problem remains a quadratic problem. However, there is much less control on how persistently exciting U_k really is. By applying the next input to the system, the SVD of the next $\mathbf{U}_{Hp}$ changes, and it is not guaranteed the current least exciting direction is the previously least exciting direction.

Remark 8. *When including the SPE term to the constraints, including a soft–constraint in the optimization problems (4.24), (4.25) and (4.28) is mandatory. This is to prevent that given the previous inputs, no U_k exists that satisfies the constraint and makes the problem infeasible.*

Remark 9. *The methods introduced above will steer the controller to generate a SPE input. However, this SPE input is not optimal for tracking and will introduce unwanted tracking offsets. Therefore, the controller should switch between computing an input suitable for estimation and an input suitable for reference tracking. While not necessary, one could disable the recursive estimation updates as long as the controller is not generating an SPE signal. This prevents however possible detection of system–controller mismatches, but at the same time, prevents the controller from adjusting based on data that is mostly noise as well. For this, we propose a rough metric ϵ_k that can be used to assess how useful the data was in the estimation. This is obtained through the a posteriori error $\bar{e}_k$ and the a priori error e_k, as follows:*

$$\begin{aligned} e_k &= \mathbf{y}_{k-N+1} - \hat{\theta}_{k-1}^T \phi_k, \\ \bar{e}_k &= \mathbf{y}_{k-N+1} - \hat{\theta}_k^T \phi_k \\ &= \mathbf{y}_{k-N+1} - \hat{\theta}_{k-1}^T \phi_k - e_k K_k^T \phi_k \\ &= e_k(1 - K_k^T \phi_k), \qquad \textit{or alternatively:} \\ \bar{e}_k &= e_k(\frac{\lambda}{\lambda + \phi_k^T \mathbf{X}_{k-1} \phi_k}) = e_k \epsilon_k. \end{aligned} \tag{4.32}$$

The resulting ϵ_k is a scalar between 0 and 1 that weights the a priori error e_k to the a posteriori error $\bar{e}_k$. If ϵ_k approaches 0, the current ϕ_k is completely fitted into the estimation, if ϵ_k approaches 1, the current data did not change $\hat{\theta}_k$. ϵ_k can therefore be used in 2 ways: it can either be used to determine whether the current ϕ_k adds useful data, where it can skip the update if not, or can be used to assess if the last SPE inputs did still provide useful data for the estimation, and disable the control law that emphasizes on persistency of excitation if so. As long as new unique data is fed to the recursive estimation, ϵ_k will increase. However, this data does not need to contain system dynamics. Therefore, it is important to ensure that the initial on–line data is by default persistently exciting. Fortunately, as the controller will change substantially initially, the results in Subsection 4.3.1 suggest that the input is by default persistently exciting.

4.4 Illustrative example

Consider again the linear actuator introduced in Section 3.2. In this simulation, we compare the off–line i-DPC with the recursive i-RDPC algorithm. To illustrate the effect of a parameter

change in the system, at $t = 10s$, the friction coefficient doubles. The simulation will be corrupted with a white noise sequence with variance $\sigma_e^2 = 0.005$.

For both controllers, the following parameters have been used: $R = 0.05$, $Q = 3\text{e}4$ and $N = 15$. In the recursive controller, we initialize $[\hat{\theta}_I]_0 = 0$ and $\mathbf{X}_0$ as an identity matrix with gain $\alpha = 1\text{e}6$. Furthermore, a forgetting factor $R_1 = 1\text{e-}5I$ is included to incorporate future model changes. The recursive controller generates an SPE input by forcing the next input to excite in the least excited direction, using $\rho = 2.5\text{e}5$, see equation (4.31).

The off–line i-DPC controller is built using the same open–loop experiment as in Section 3.2. The recursive controller runs in open loop between 0 and $4N$ samples, using a white noise sequence with a variance of $\sigma_u^2 = 5$ as input. The recursive identification initiates after $2N$ samples have been collected. From $t = 1.2s$ to $t = 4s$, the loop closes and the controller runs with the SPE conditions, after $t = 4s$ it loses this condition and the quadratic program is equal to that of the off–line i-DPC.

Aside from the output and input plots, the prediction error has been added as well. This prediction error is the norm of the difference of the predicted output using the identified prediction matrices compared to the actual future outputs, that is computed noise–free (similar to (3.11)). The simulation results are shown in Figure 4.2.

The effect the SPE condition in the recursive controller can be observed in the output and input plots. After $t = 4s$, when the SPE condition is dropped, the controller tracks the system accordingly and the prediction error converges to that of the off–line i-DPC. After the system changes at $t = 10s$, both controllers show a significant increase in the prediction error and tracking error. However, while this error does not reduce for the i-DPC, the recursive i-RDPC adjusts the prediction matrices to this new model and recovers optimal tracking.

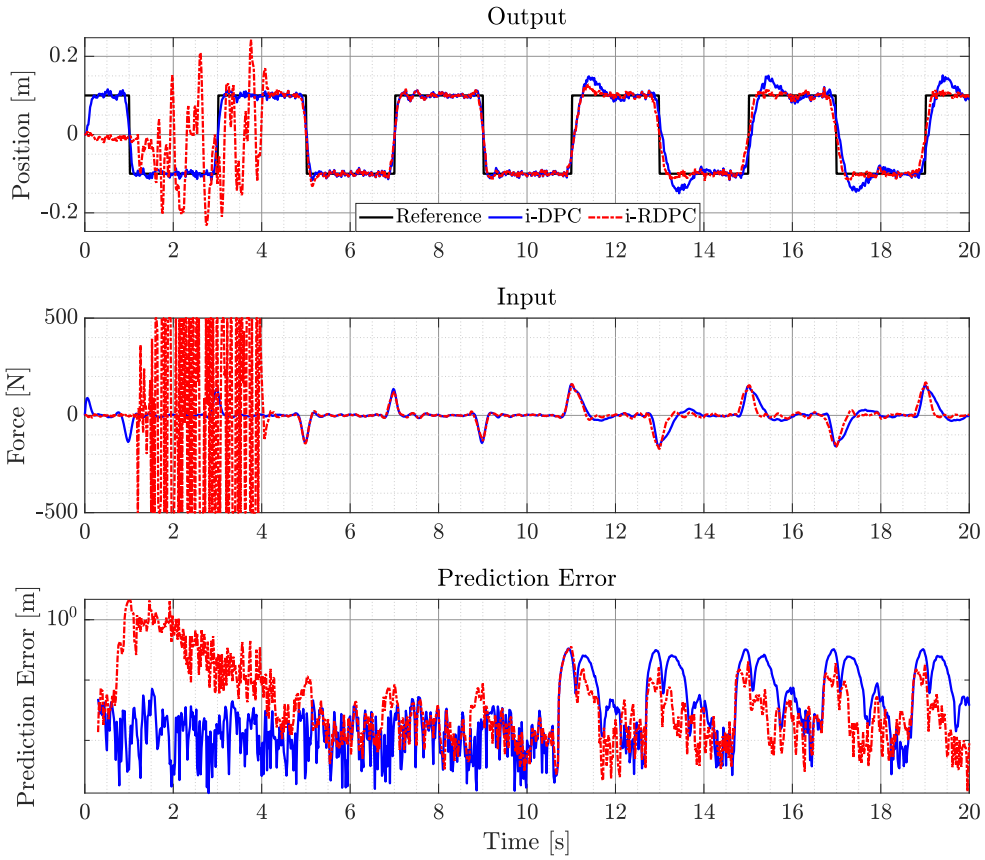


Figure 4.2: Simulations on a linear actuator using an off–line i–DPC (blue) and the recursive i–RDPC (red)

Chapter 5

Implementation on a DC-DC Buck Converter

Data–driven controllers are built without a model, directly from measured system input/output data. Hence, DD controllers avoid the difficult step of creating a model, and the inaccuracies caused by poor parameter estimation. However, this advantage cannot be illustrated through a simulation, as it simulates the controllers on a model. In the simulation, the best performance the DD controller can achieve, is the performance of an equivalent model based controller. As such, in this chapter, we illustrate the resulting DD controllers on a real system instead. Due to the circumstances in which this graduation project is executed, the system used will be a crudely built DC-DC Voltage buck converter. The setup and the controller are introduced in section 5.1, followed by the results of the experiment in section 5.2.

5.1 The Setup

A DC-DC Voltage buck converter is a step down voltage regulator. As it operates as a switched–mode power supply, buck converters are very energy efficient. Figure 5.1 shows the

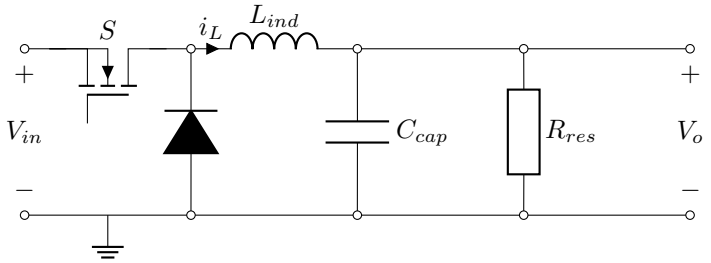


Figure 5.1: Schematic of a DC-DC Voltage Buck Converter

schematic of the buck converter used. In the setup, the switch S is controlled by a pulse–width modulated (PWM) signal, where $u(t)$ is the duty cycle.

This system will be controlled using an Arduino Uno as shown in Figure 5.2, which provides both the PWM signal and the input voltage V_{in}. Considering that the Arduino Uno runs on a 16MHz microprocessor, it is doubtful the Arduino can solve a quadratic program in a feasible

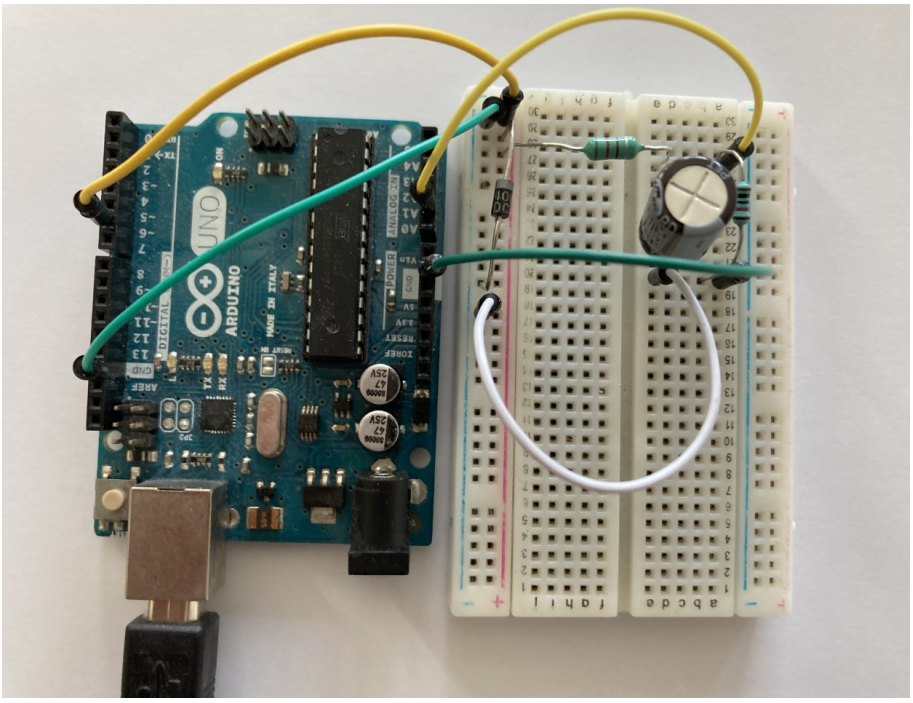


Figure 5.2: The circuit in connection with the Arduino board

sampling time (note that in general, buck converters are fast). Instead, the Arduino will run as an interface, where the actual control input will be computed by a python script. The communication is specifically designed to be as minimal as possible, requiring only 2 bytes to send a new control input, and 1 bytes send + 2 bytes received to obtain the last measurement. Thanks to this control setup, the controller has a sampling frequency of 250Hz. This is limited due to a latency in Windows between when USB data is received and when it is read in the python script. To compensate for this lower sampling rate, the buck converter is built using a capacitor and an inductor with a relatively high capacitance and inductance. Otherwise, the controller would simply control something without any measurable dynamics. In this setup, the following component values are used: $R_{res} = 10\text{k}\Omega \ \pm 1\%$, $C_{cap} = 470\mu\text{F} \ \pm 20\%$, $L_{ind} = 830\mu\text{H} \ \pm 10\%$ and $V_{in} = 5\text{V}$.

Furthermore, to ensure the switching behaviour of the PWM input does not cause unwanted harmonics in the output, the PWM frequency is increased to 62500Hz (instead of the default 960Hz). The output $y(t) = V_o$ is measured using an 10bit Analog-to-Digital-Converter (ADC), yielding a resolution of 4.9mV. Together, this imposes the following constraints on the system:

$$\begin{aligned} 0 \leq & u(t) \leq 1 \\ 0 \leq & y(t) \leq V_{in} \end{aligned} \tag{5.1}$$

Next, we continue with the experiments on different predictive controllers and discuss the results.

5.2 Experiments

To ensure zero–offset reference tracking, every controller tested on the system will include an integral action. Let us introduce the controllers.

- A **model–based predictive controller.** This is designed from the following model

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix}V_o(t)\\ i_L(t)\end{bmatrix} &= \begin{bmatrix}-\frac{1}{R_{res}C_{cap}} & \frac{1}{C_{cap}}\\ -\frac{1}{L_{ind}} & 0\end{bmatrix}\begin{bmatrix}V_o(t)\\ i_L(t)\end{bmatrix} + \begin{bmatrix}0\\ \frac{V_{in}}{L_{ind}}\end{bmatrix}u(t),\\ y(t) &= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}V_o(t)\\ i_L(t)\end{bmatrix},\end{aligned} \tag{5.2}$$

  or, equivalently,

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad y(t) = Cx(t).$$

  This model is discretized using a zero-order holder with a sampling time of $T_s = 4$ms. Using this model, an integral action MPC is designed, by obtaining the augmented state space matrices (A_I, B_I and C_I) as in (3.1) and building the cost function as show in Equation (3.2). To obtain the state estimate, an Luenberger observer is built on the integral action model as follows:

$$\hat{x}_I(k+1) = (A_I - L_I C_I)\hat{x}_I(k) + B_I\Delta u(k) + L_I y(k), \tag{5.3}$$

  where the observer gain L_I is obtained using the LQR command in MATLAB[1].

  Remark 10. *Due to the high component values, the current in the circuit can become significantly larger than the saturation current on the Arduino (20mA). If the current drawn in the circuit is larger than the saturation current, the voltage of the Arduino is automatically reduced to prevent damaging the board. This effect causes a non-linear behaviour in the response that is not visible in the model.*

- An **off–line i-DPC controller**. Prior to the actual control experiment, an open–loop experiment is performed to obtain system input/output data. This experiment has a length of 2000 samples, using a RBS input between $[0, 1]$ with clock period 1. Using this off–line experiment data, the controller is applied to the system in closed–loop following the steps in Algorithm 1.

- The **i-RDPC controller**, following the steps explained in Algorithm 3. The prediction matrices are initialized as zero matrices, and $\mathbf{X}_0 = 1\text{e}^6 I_{3N}$. Because of this, the experiment starts in open–loop, using an RBS input between $[0, 1]$ and clock period 1, until $5N$ samples have been collected. After $3N$ samples, the recursive estimation initiates. Furthermore, no forgetting factor is included and no SPE method has been included into the control law.

All three controllers are designed using the same controller tuning parameters: $Q = 1$, $R = 100$ and $N = 15$. The goal is to track a square wave reference with period of $4s$ between 1V and 4V for 10 seconds.

The closed–loop experiments with each controller on the system are portrayed in figures 5.3, 5.4 and 5.5. Next to the output and input plots, a zoom section is provided that zooms in at the rising flank at $t = 4s$. Thanks to the integral action, all three of the controllers show excellent tracking performance, as the output oscillates between 2-4 steps of the ADC. Even though all

[1] `Li = dlqr(Ai', Ci', 1000*eye(3), 1)';`

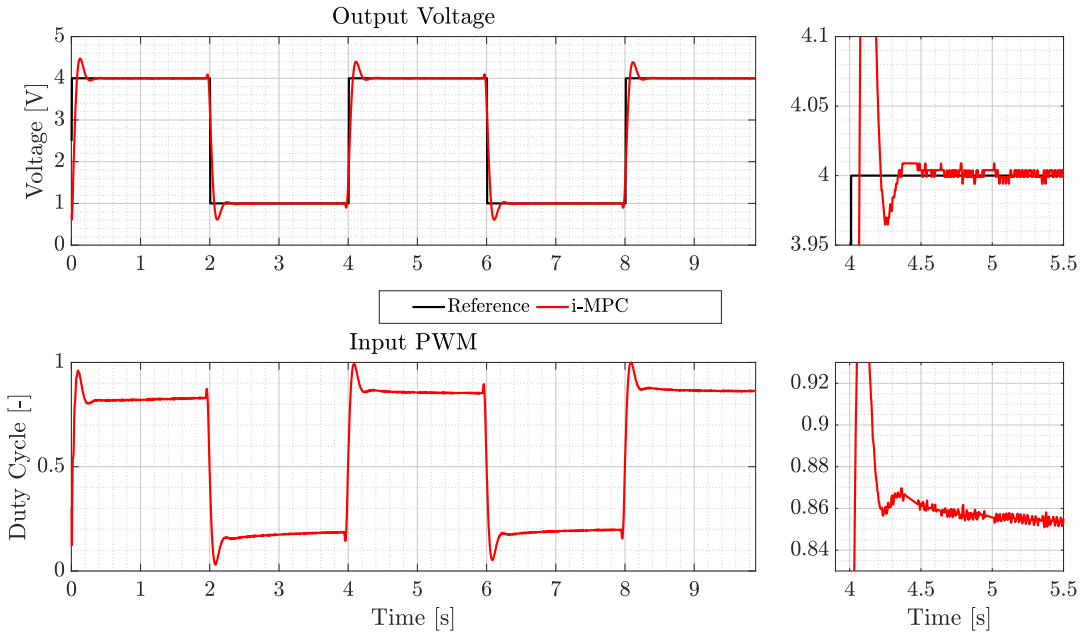


Figure 5.3: Buck converter controlled by an MPC with integral action

controllers show some overshoot, the magnitude thereof varies significantly. The model based controller shows the largest overshoot. A likely explanation for this is that when disregarding the fact that the current saturates, the system has a very oscillatory response. As this non-linear behaviour is not included into the model, the controller attempts to dampen the response. Thanks to the prior open–loop experiment, the i–DPC controller shows the best performance. The overshoot is minimal, while maintaining the same rise time as the other two controllers. The controller performance of the plug–and–play i-RDPC show that the initial input and the initial conditions of the controller generate a sufficient input signal to properly identify the prediction matrices. However, unlike the off–line i-DPC controller, there is still some overshoot and steady–state tracking error has a larger variance.

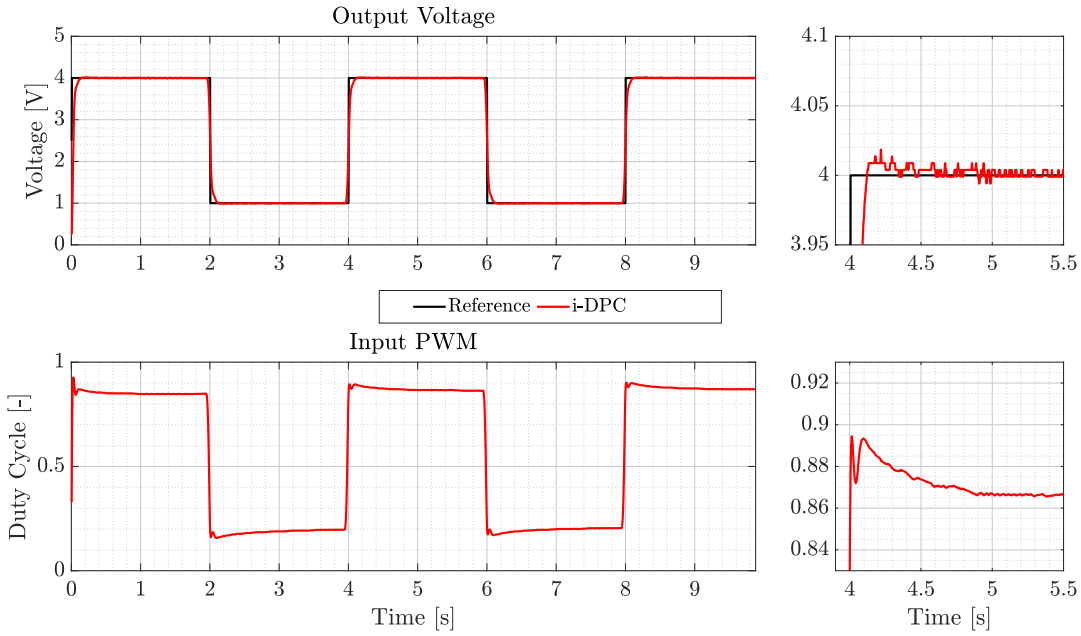


Figure 5.4: Buck converter controlled by the proposed off–line i–DPC

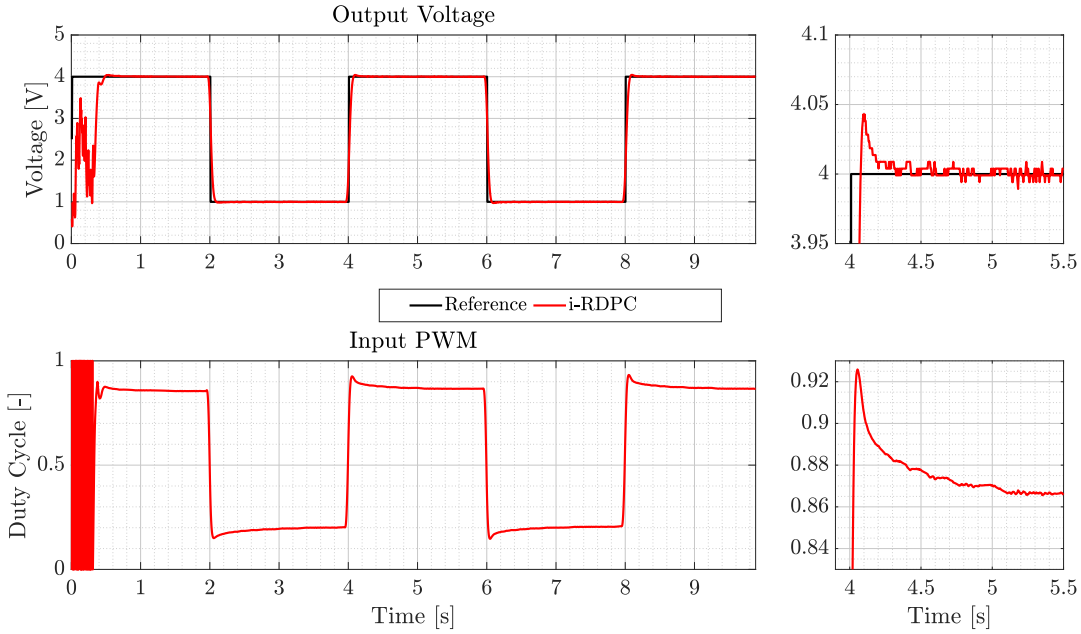


Figure 5.5: Buck converter controlled by the proposed recursive i–RDPC

Chapter 6

Conclusions

In this thesis we analysed two different problems arising in data–driven predictive control. The first problem is to achieve zero–offset reference tracking for constant references in a data–driven predictive controller, assuming only input/output data is available. Even then, the data–driven predictive controller is built from one finite set of data, measured at one time moment. This means that the resulting controller only operates optimally when the system dynamics match the observed dynamics in the data set of the experiment. Due to time–varying and/or non–linear behaviour in the system, this is almost never the case. The second considered problem is therefore how to prevent loss of performance due to mismatch in the estimated dynamics and the actual system dynamics, again assuming only input/output data is available and, without re-designing the controller.

In Chapter 3, we developed a new approach that obtains the corresponding integral action prediction matrices of the system by only modifying the input data used in the least–squares estimation. This allows us to build a rate–based data–driven predictive controller, which achieves zero–offset reference tracking for constant references by controlling the rate of change of the input instead of the input itself. Thanks to this, the resulting controller is also capable of removing the tracking offset caused by constant disturbances.

To prevent the loss of performance due to mismatch in the estimated dynamics and the actual system dynamics, in Chapter 4 we introduced a recursive data–driven predictive controller. This controller uses a recursive least–squares algorithm that updates the prediction matrices of the controller each time sample using on–line measured input/output data. Because of this, the prediction matrices and the corresponding controller are continuously updated to match the dynamics of the system in real–time. Besides that, we introduced several methods that enforce the controller to generate a strongly persistently exciting input, such that the recursive estimation is guaranteed to converge during closed–loop control. The recursive data–driven predictive controller can not only be used to continuously update a controller that is initialized off–line, but also without any initialization, as a plug–and–play controller.

The new control algorithms are simulated on a linear actuator model, and they are used in an experiment on a real–life DC-DC Voltage buck converter system. The results of the experiments illustrate the effectiveness of the proposed algorithms in terms of tracking performance, robustness, disturbance rejection, adapting to system changes and noisy measurements. In future work, it is a point of interest to develop a stability criterion and achieve recursive feasibility of the data–driven controllers (either recursive or off–line).

Bibliography

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, 2 ed., 2019. 1, 5

[2] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009. 1, 5, 8, 11

[3] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annual Reviews in Control*, vol. 43, pp. 1–64, 2017. 1

[4] I. Markovsky and P. Rapisarda, "Data-driven simulation and control," *International Journal of Control*, vol. 81, no. 12, pp. 1945–1959, 2008. 1

[5] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-Driven Model Predictive Control with Stability and Robustness Guarantees," *IEEE Transactions on Automatic Control*, 2020. 1, 11

[6] H. Yang and S. Li, "A Data–Driven Predictive Controller Design Based on Reduced Hankel Matrix," *10th Asian Control Conference*, pp. 1–7, 2015. 1

[7] J. Coulson and J. Lygeros and F. Dörfler, "Data-Enabled Predictive Control: In the Shallows of the DeePC," *18th European Control Conference*, p. 307–312, 2019. 1

[8] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-Driven Tracking MPC for Changing Setpoints," *ArXiv preprint: 1910.09443*, 2020. 1

[9] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "On the design of terminal ingredients for data–driven MPC," *IFAC PapersOnLine*, vol. 54, no. 6, pp. 257–263, 2021. 1

[10] W. Favoreel, B. de Moor, and M. Gevers, "SPC: Subspace Predictive Control," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, 1999. 1, 6, 7, 12

[11] R. Kadali, B. Huang, and A. Rossiter, "A Data Driven Subspace Approach to Predictive Controller Design," *Control Eng. Pract.*, vol. 11, no. 3, pp. 261–278, 2003. 1, 2, 11, 14

[12] P. van Overschee and B. L. de Moor, *Subspace Identification for Linear Systems*. Bosten MA: Springer, 1 ed., 1996. 1, 5, 6, 21

[13] G. R. Gonçalves da Silva, A. S. Bazanella, C. Lorenzini, and L. Campestrini, "Data-Driven LQR Control Design," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 180–185, 2019. 2, 6

[14] G. R. Gonçalves da Silva, A. S. Bazanella, and L. Campestrini, "One-shot data-driven controller certification," *ISA Transactions*, vol. 99, pp. 361–373, 2020. 2

[15] W. Aangenent, D. Kostic, B. de Jager, R. van de Molengraft, and M. Steinbuch, "Data-based optimal control," *Proc. Amer. Control Conf.*, vol. 2, pp. 1460–1465, 2005. 2, 6, 16

[16] X. Luo and Y. Song, "Adaptive Predictive Control: A Data–Driven Closed–Loop Subspace Identification Approach," *Abstract and Applied Analysis*, pp. 1–7, 2014. 2, 16

[17] R. Hallouzi and M. Verhaegen, "Persistency of excitation in subspace predictive control," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 11439–11444, 2008. 2, 16, 25

[18] R. K. Lim, M. Q. Phan, and R. W. Longman, "State estimation with ARMarkov models," *Dept. Mech. Aerosp. Eng. Tech. Rep.*, no. 3046, 1998. 6

[19] T. T. Nguyen, M. Lazar, and H. Butler, "Nonlinear model predictive control for ironless linear motors," in *IEEE Conference on Control Technology and Applications*, pp. 927–932, IEEE, 2018. 12

[20] N. A. Mardi and L. Wang, "Subspace-based model predictive control of time-varying systems," *48th IEEE Conference on Decision and Control*, pp. 4004–4010, 2010. 16

[21] B. A. Hernandez and P. A. Trodden, "Stabilizing predictive control with persistence of excitation for constrained linear systems," *ArXiv preprint: 1804.07506*, 2018. 16, 23

[22] T. Söderström and P. Stoica, *System Identification*. Hemel Hempstead, Hertfordshire, UK: Prentice Hall International, 1 ed., 1989. 16, 18, 20, 22

[23] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: The MIT Press, 1 ed., 1983. 16, 18, 23

[24] M. A. Woodbury, "Inverting modified matrices," in *Memorandum Rept. 42, Statistical Research Group*, p. 4, Princeton Univ., 1950. 18

[25] P. M. J. Van den Hof, "Closed-loop issues in system identification," *Annual Reviews in Control*, vol. 22, pp. 173–186, 1998. 22

[26] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. De Moor, "A note on persistency of excitation," *System & Control Letters*, vol. 54, pp. 325–329, 2005. 23

[27] MATLAB, "Nonlinear constraints." `https://mathworks.com/help/optim/ug/nonlinear-constraints.html`. Accessed: 2021-10-18. 24