Chapter 1

Introduction

1.1 Motivation

It is well known that repetition is the mother of learning. A student learning a piano piece repeat it till he or she can play it well, and, possibly, without errors. A gymnastic tries to make a perfect move by exercising it douses and hundreds of times. In both cases, the correct motion is learned and becomes ingrained into muscle memory. The converged muscle motion profile can bee seen as an open-loop control generated through repetition and learning. This type of learned open-loop control strategy is the essence of Iterative Learning Control (ILC).

Nowadays, more and more routine tasks are carried out by industrial machines or robots. The machines can achieve better precision, can perform dangerous or dirty work. It is often a cheap and effective solution for big industrial companies. ILC can be here of great use if a task needs to be done frequently. From each iteration, we can learn something to achieve better performance.

For example, let us imagine that a robot arm needs to take the same path, again and again, with high accuracy. We can possibly find a controller, which might run the robot arm along this route. However, to achieve good precision, a simple controller might not be enough, and we need to apply more complicated techniques to improve the result.

The other possibility is to take the result we get from the initial controller, and try to adjust it for the next trial. Improving the signal by each program run, we can achieve a perfect result with less effort if the signals are adapted with a proper algorithm. In this work, we consider some of such algorithms and see, that under certain conditions we can achieve monotonic convergence to an input signal, which renders an optimal tracking.

1.2 Problem Statement

There are many different ILC algorithms. One can find a good overview in [1]. The focus of this work is studying the so-called unit memory algorithms on the example of discrete-time dynamical systems over a finite time horizon. We also aim to prove robustness property for these algorithms despite the parameter uncertainty. Besides, we want to apply the algorithms we provide on discrete- and continuous-time systems. Especially, we face the issues that occur due to system discretization, the amount of data and numerical instability.

The unit memory algorithms are the ones, which can "remember" the last pass. Mathematically speaking, if u_k is the input signal and e_k is the tracking error at trial k, then

$$u_{k+1} = f(e_k, e_{k+1}, u_k) \tag{1.1}$$

where f is some function independent on k [2].

The l-input m-output discrete time systems can be written as

$$\begin{array}{rcl} x(t+1) &=& Ax(t) + Bu(t) \\ y(t) &=& Cx(t) + Du(t), \\ x(0) &=& x_0, \end{array} \tag{1.2}$$

over a time horizon $t = 0, 1, 2, \ldots, N$. A, B, C, and D denote here the matrices of fitting dimensions. We are interested in the minimizing of the error between output $y(\cdot)$ and the tracking signal $r(\cdot)$

$$e(t) = r(t) - y(t),\ t = 0, 1, 2, \ldots, N. \tag{1.3}$$

An uncommon contemplation in this work is the final time horizon. Together with discreteness of the the system it invites us to rewrite the system (1.2) in a more compact form. We "stock" the sequences

$$\{u(0), u(1), u(2), \ldots, u(N)\} \text{ and } \{y(0), y(1), y(2), \ldots, y(N)\}$$

as

$$u = \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(N) \end{pmatrix} \text{ and } y = \begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{pmatrix}.$$

The resulting vectors u and y we call supervectors.

Then the system (1.2) can be expressed as

$$y = Gu + d,$$

where

$$G = G(A, B, C, D) = \\ = \begin{pmatrix} D & 0 & \cdots & 0 & 0 & 0 \\ CB & D & \cdots & 0 & 0 & 0 \\ CAB & CB & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ CA^{N-2}B & CA^{N-3}B & \dots & CB & D & 0 \\ CA^{N-1}B & CA^{N-2}B & \dots & CAB & CB & D \end{pmatrix} \in \mathbb{R}^{m(N+1)\times l(N+1)}.$$

and

$$d = d(C, A, x_0) = \begin{pmatrix} Cx_0 \\ CAx_0 \\ CA^2x_0 \\ \vdots \\ CA^Nx_0 \end{pmatrix} \in \mathbb{R}^{m(N+1)}.$$

With other words, we put all the information about the system at times $t = 0, 1, \dots, N$ in one linear equation. This form already inspires some of ILC algorithms, for example Pseudo Inverse Model Algorithm

$$u_{k+1} = u_k + \beta GG^+ e_k,$$

where G^+ is the pseudo inverse of G and k denotes the trial of system run, $\beta \in (0, 2)$.

To make the notation less confusing, we denote with $u(t)$ the value of signal $u(\cdot)$ at time t, and with u the corresponding supervector form. u_k denotes the supervector at trial k of algorithm running.

Five different algorithms are illustrated in this work. Two of them are derived on considering of inverse model, two are inspired by steepest descent, and one is based on feedback design. The algorithms in this thesis are taken from [2]. However, some of them were modified to make them more practically usable.

Besides, we consider the robustness property of Unit Memory Algorithms despite the parametric uncertainty. Using Lyapunov techniques and Full-Block S-Procedure we derive a method, which allows us to check if the given algorithm stays stable if the parameters of the system are uncertain.

Furthermore, we discuss the issues appearing by the application of the algorithms.

With the supervector description, the ILC algorithms need an lot of computational memory and lead to significant numerical errors. We need to modify the algorithms in the way that we can apply them to the real problems.

1.3 Structure

This work consists of 6 chapters. The thesis is composed as follows: in the first chapter we in detail formulate the problem. After that, in Chapter 2, we recap the crucial basics for understanding the considered algorithms. In Chapter 3 we introduce the supervector notation and inspired by it, derive the ILC algorithms. In chapter 4 we succeed the requirements, under which the algorithms stay robust despite the parametric uncertainty. Finally, in chapter 5 we want to apply ILC algorithms to a real dynamical system. We face the issues the real problems yield, and suggest the remedies. The summary concludes this thesis, where further work is suggested.