# LEARNING FEED-FORWARD CONTROL

– THEORY, DESIGN AND APPLICATIONS –

This dissertation has been completed in partial fulfillment of the requirements of the Dutch Institute of Systems and Control DISC for graduate study.

ISBN 90-36514126

Printed by Drukkerij Twente Hengelo B.V.

# LEARNING FEED-FORWARD CONTROL

## – THEORY, DESIGN AND APPLICATIONS –

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 25 februari 2000 te 15.00 uur.

door

**Wubbe Jan Roelf Velthuis**

geboren op 1 december 1970
te Stadskanaal

Dit proefschrift is goedgekeurd door
prof. dr. ir. J. van Amerongen, promotor
dr. ir. T.J.A. de Vries, assistent-promotor.

*Voor Femke en Job*

# Table of Contents

# Voorwoord

Een promotie is als een puzzel. Door de hulp van anderen kom je tot oplossingen die je in je eentje niet gevonden zou hebben. Allen die geholpen hebben mijn puzzel op te lossen wil ik daarvoor danken.

Je naaste collega's bepalen in grote mate het plezier dat je hebt in je werk. Erik, met jou had ik mij geen betere collega kunnen wensen.
Naast leuke collega's, is een begeleider bij wie je elk ogenblik van de dag kunt aankloppen voor raad, één van de meest belangrijke zaken voor een AIO. Theo, je was er altijd en ik heb je hulp als erg waardevol ervaren. Jij hebt me geleerd dat een opgeruimde kamer je werk ten goede komt. Ik hoop dat we onze samenwerking in de toekomst kunnen voortzetten.
Voor zijn goede aanwijzingen voor het schrijven van dit proefschrift wil ik graag Job bedanken.

Zonder de bijdragen van de volgende studenten zou dit onderzoek er anders uit hebben gezien. Bas, Douwe, Jaap, Jacques, Koen, Lars, Levi, Menno, Patrick, Pieter, Rene, Roy, Tom en Toon, bedankt.

Dit onderzoek werd mede mogelijk gemaakt door de volgende bedrijven: Philips CFT, Fokker Control Systems en Nefit-Fasto. Hun hulp en uitdagingen heb ik zeer op prijs gesteld.

Werk is meer dan werk alleen. De gezelligheid en de ondersteuning van alle (oud-) medewerkers van het lab en CLP hebben de afgelopen 4 jaar tot een erg leuke periode gemaakt.

Als laatste bedank ik mijn ouders, schoonouders en bovenal Femke en mijn *eigen* Job.

Wubbe Jan

# 1 Introduction

## 1.1 Why Learning Control?

One of the competitive tools manufacturers have at their disposal is the product performance [Dibb *et al.*, 1991]. Especially in markets involving high-tech products it is of vital importance to manufacture products that have a superior performance. This is also observed in the market for products we will consider in this thesis, i.e. electro-mechanical motion systems. From a mechatronic point of view, the performance of electro-mechanical motion systems can be improved by changing the mechanical design and/or the controller. Consider, for example, robot manipulators, where the motion accuracy depends on the stiffness and the inertia of the system. If the manipulator fails to meet specifications, its stiffness can be increased or it's inertia can be reduced by changing the mechanical structure or by application of new materials. Changing the controller can be done by either tuning the parameters of the existing controller or by designing a new one. Since an improved controller merely requires software changes and in some situations addition of sensors, this measure is relatively easy to implement, compared to structural adaptations. In this thesis we will focus on increasing the performance by means of control.

The design of a controller is generally based on a model of the plant. A more accurate model of the plant will result in a controller with a better performance. When modelling a plant, the following problems can be encountered [Harris *et al.*, 1993]:

- *The system is too complex to understand or to represent in a simple way.*
- *The model is difficult or expensive to evaluate.* The characteristics of some (non-linear) effects may be hard to obtain, e.g. friction.
- *The plant may be subject to large environmental disturbances, which are difficult to predict.*
- *The plant parameters may be time-varying.*

Adaptive control [Amerongen, 1982; Åström and Wittenmark, 1989; Mareels and Polderman, 1996] can offer a solution when the structure of the model of the plant dynamics and the disturbances that act on it is available, but the values of some of the parameters cannot be determined [Landau, 1979]. When the model is not available or when many parameters cannot be determined, learning control may be considered.

## 1.2 What is Learning Control?

Learning controllers are often pictured as a close representation of the human control system and thus would possess human like properties. In this thesis we don't study learning controllers from this biological point of view, but rather adopt the following definition.

**Definition 1.1 (Learning controller)** *A learning controller is a control system that comprises a function approximator of which the input-output mapping is adapted during control, in such way that a desired behaviour of the controlled system is obtained.*

**Definition 1.2 (Function approximator)** *A function approximator is an input-output mapping determined by a selected function $F(\cdot,\boldsymbol{w})$, of which the parameter vector $\boldsymbol{w}$ is chosen such that a function $f(\cdot)$ is "best" approximated.*

**Remark 1.1 (Learning and adaptive control)** In this sense, adaptive control can be viewed as a form of learning control in which a function approximator is used that can only approximate a limited class of target functions. In general, a learning controller will contain a function approximator for a much richer class of target functions.

A wide variety of function approximators can be used, such as neural networks [Hertz *et al.*, 1991; Haykin, 1994; Veelenturf, 1995] neuro-fuzzy networks (also known as adaptive fuzzy logic controllers) [Brown and Harris, 1994; Jang and Sun,

1995], look-up tables [Moore, 1999], etc. Roughly speaking, the function approximator can be used in two ways. *Firstly*, the function approximator can be used to generate (part of) the control signal. Learning takes place by adapting the parameter vector of the function approximator in such way that some cost function containing the control error is minimised. This is known as direct learning control [Ng, 1997]. *Secondly*, the function approximator can be used to learn a model of the plant under control, which is adapted such as to minimise a cost function of the prediction error. On the basis of the learned model, a controller is derived. This is known as indirect learning control [Ng, 1997].

From the time that the first learning controller was developed in 1963 until now, the field of learning control has grown tremendously. Many different controller structures have been proposed and their properties, such as stability and convergence rate, have been analysed both in practice and in theory. For an overview of the field of learning control we refer to [Hunt *et al.*, 1992; Prabhu and Grag, 1996; Agarwal, 1997; Ng, 1997]. However, in spite of all research, only few learning controllers have been applied in commercial products. Possible reasons are [Longman, 1998]:

- *Proof of stability is overrated.* A large part of the theoretical research on learning controllers is focused on stability. However, a stable learning controller does not necessarily yield a good learning transient response. In [Longman, 1998] the behaviour of a learning controller for a robot was observed by means of simulations. After performing a specific motion 6 times, the tracking error had decreased by a factor 2.8. When learning continued, the tracking error grew by a factor $10^{51}$ at 62.000 repetitions and finally decreased to a factor $10^{-18}$ at 250.000 repetitions. So, in spite of the fact that finally a small tracking error was obtained, this learning controller has no practical value because of the very large tracking error in between.
- *A zero tracking error should not be desired.* Some learning controllers attempt to obtain a zero tracking error. However, this requires large control signals at frequencies above the system bandwidth, which may be harmful for the actuators and is generally undesirable.
- *Wrong type of function approximator.* In the majority of learning controllers, the function approximator is implemented as a Multi Layer Perceptron (MLP) neural network [Poggio and Girosi, 1990] (see appendix A). As we will discuss later, this type of neural network is not especially suited for control.

Based on these considerations we can formulate the following properties that a learning controller should possess in order to become commercially interesting:

- *Easy to use in an existing control system.* This means that, as long as there is a good learning transient response, a minimum performance is guaranteed, i.e. that of the existing controller. Even during training the plant can be kept in operation, not resulting in loss of production.
- *Ability to make use of a priori knowledge of the plant.* Generally, designers and/or operators have some knowledge of the plant, e.g. in the form of a (simple) mathematical model in terms of equations, a Bode plot of the plant or a linguistic description of the plant behaviour. The learning controller should allow this type of knowledge to be incorporated in the controller design, in order to choose sensible controller parameters and to speed up learning.
- *The function approximator should be suited for control.* This means that:
  - *The memory space that is needed for implementation should be small.* In practice, the controller is implemented in software on an embedded computer. The amount of memory is limited. Therefore, the number of parameters the function approximator requires to approximate the control signal may not be too large.
  - *Calculation of the output of the function approximator and adaptation of the input-output relation must be done fast.* In a real-time environment, within one sample interval, the parameters of the function approximator are adapted *and* the output is calculated. Many electro-mechanical motion systems require a small sample time, leaving little time for calculations. Function approximators that involve a large number of complex calculations are therefore not suited for control.
  - *The learning mechanism should converge fast.* In order to keep the amount of time, in which the performance of the controlled system is sub-optimal, to a minimum, the learning mechanism should converge fast.
  - *The learning mechanism should not suffer from local minima.* When trapped in a local minimum, the learning mechanism assumes that the obtained values of the parameters of the function approximator, denoted by $\boldsymbol{w}_{loc}$, yield a minimum approximation error, denoted by $E(\boldsymbol{w}_{loc})$. This in spite of the fact that a $\boldsymbol{w}_{glob} \neq \boldsymbol{w}_{loc}$ exists, for which $E(\boldsymbol{w}_{loc}) > E(\boldsymbol{w}_{glob})$. In figure 1.1 a 1-dimensional example of such a phenomenon is presented. In the local minimum, $\boldsymbol{w}=\boldsymbol{w}_{loc}$, the gradient of the approximation error is equal to zero. Learning mechanisms that only use the gradient of the approximation error are unable to escape from the local minimum.

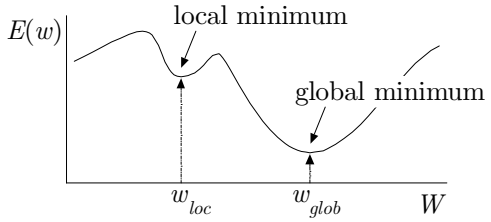



*Figure 1.1: Local minimum in the learning mechanism*

When the learning mechanism easily traps in a local minimum, it is difficult to train the learning controller such that a high performance is obtained.

- *The input-output relation must be locally adaptable.* In some function approximators, the input-output relation is adapted globally. This means that, if the value of one of the parameters of the function approximator is adapted, the input-output relation over the entire input domain is changed. Consider a learning controller that has been trained to perform several motions. When the learning controller is trained to perform a new motion, this involves adapting the parameters of the function approximator. Because the input-output relation is adapted globally, the previously learned control signals are changed, which may cause a loss of performance. Therefore, it is desirable that the input-output relation of a function approximator can be adapted locally. In this case, learning a new motion will not change the previously learned control signals.
- *The function approximator should possess a good generalising ability.* The generalising ability is the ability to produce a sensible output for an input that has not been presented during training but that is similar to training examples. When the function approximator has a good generalising ability, the learning controller will also obtain a high tracking performance for motions that are similar to the trained motions. It thus suffices to train the learning controller with a small number of characteristic training motions. When the function approximator has a poor generalising ability, the learning controller must be trained for each motion of interest, yielding an extensive training procedure.
- *The smoothness of the approximation should be controllable.* As argued before, the learning controller should only try to obtain a zero tracking error up to some frequency, since high-frequency control signals are undesirable. The user must be able to determine the maximum frequency of the output of the function approximator.

- *Good learning transient response.* The transient response of the learning controller should be such that the tracking error gradually converges to the desired value. Growing tracking errors in the intermediate phase of learning

might damage the plant. Furthermore, when this is the case, a minimum performance can no longer be guaranteed when the controller is used as an add-on device for an existing controller.

– *Long term stability has to be guaranteed.* Learning can be either performed continuously or only prior to operation. Continuous learning is required when the plant parameters change during operation, e.g. due to wear or environmental influences. In this case, one must be able to guarantee stable learning in spite of changing operation conditions.

## 1.3 Feedback Error Learning

An interesting learning controller for robot manipulators that have to track random paths, was proposed by Kawato [Kawato *et al.*, 1987; Miyamoto *et al.*, 1988] (figure 1.2). This controller is generally known as the Feedback Error Learning (FEL) controller.

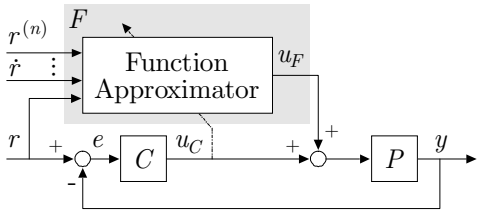



*Figure 1.2: Feedback Error Learning*

The learning control system consists of two parts:

– *The feed-forward controller*, denoted by $F$, i.e. a mapping $u_F = F(\boldsymbol{r})$. A conventional feed-forward controller can be used to compensate for the system dynamics and in this way obtains a high tracking accuracy. When the feed-forward controller is equal to the inverse plant, $F = P^{-1}$, the output of the plant, $y$, will equal the reference, $r$.

The plant, $P$, is always subject to disturbances. These disturbances can either have a stochastic or a reproducible nature. Reproducible disturbances reoccur in the same way when a specific motion is repeated. This means that they can be viewed as a function of the state of the plant, $\boldsymbol{x}$, see figure 1.3.

An example is position dependent friction. Like the plant dynamics, the reproducible disturbances can be compensated by a feed-forward controller.

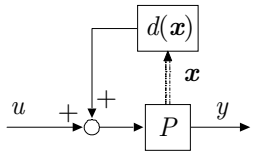



*Figure 1.3: Plant and reproducible disturbances*

In order to correctly compensate for the system dynamics and the reproducible disturbances, a detailed model is required. Inaccuracies in the model can result in a feed-forward controller that has a poor performance. When an accurate model is difficult to obtain, an alternative approach can be taken.

Instead of designing a feed-forward controller on the basis of a model, [Kawato *et al.*, 1987] proposed to implement the feed-forward controller as a function approximator, i.e. $u_F$=$F(\boldsymbol{r},\boldsymbol{w})$. During control, the input-output relation of the function approximator is adapted such that it learns the inverse plant and the compensation of the reproducible disturbances. The main difficulty is now to select a learning signal, which indicates how the input-output relation of the function approximator must be adapted. The learning signal can be obtained in a number of ways [Er and Liew, 1997]. [Kawato *et al.*, 1987; Kawato, 1990] showed that, when the output of the feedback controller is used as a learning signal (figure 1.2), the input-output relation of the function approximator converges to the inverse plant and the compensation of the reproducible disturbances.

The type of function approximator that Kawato used is a Multi Layer Perceptron (MLP) neural network (see appendix A). In case of the 3 Degrees Of Freedom (DOF) manipulator that Kawato used in his experiments, the input of the MLP consisted of the reference joint angles, $\boldsymbol{\theta}_d$, and their first and second order derivatives:

$$\boldsymbol{r} = \left[\theta_{d,1}\ \dot{\theta}_{d,1}\ \ddot{\theta}_{d,1}\ \theta_{d,2}\ \dot{\theta}_{d,2}\ \ddot{\theta}_{d,2}\ \theta_{d,3}\ \dot{\theta}_{d,3}\ \ddot{\theta}_{d,3}\right]^T \in \mathbb{R}^9 \tag{1.1}$$

while the output, $\boldsymbol{u}_F$ consisted of the motor torques, $\boldsymbol{\tau}$:

$$\boldsymbol{u}_F = \left[\tau_1\ \tau_2\ \tau_3\right]^T \in \mathbb{R}^3 \tag{1.2}$$

– *The feedback controller.* As stated, the feedback controller, $C$, provides the learning signal for the feed-forward controller. Furthermore, it determines minimum tracking performance at the beginning of learning. Finally, the feedback controller compensates random disturbances.

The FEL controller has been applied to a number of applications by others as well, for example:

– an automatic braking system for automobiles [Ohno *et al.*, 1994];
– control of a camera system [Bruske *et al.*, 1997];
– control of robot manipulators [Kim and Lee, 1996];
– welding [Tzafestas *et al.*, 1997].

The applications showed that the FEL controller considerably improved upon the performance of the feedback controller and that it is able to obtain a high tracking performance without extensive modelling. In [Kraft and Campagna, 1990; Kim and Lee, 1996; Tzafestas *et al.*, 1997] the behaviour of a FEL controller is compared to that of adaptive control systems. They concluded that, in case an accurate plant model is used in the adaptive controller, the tracking performance of the adaptive and the FEL controller are similar. Since the FEL controller converges slower than the adaptive controller, in this situation the latter is preferred. However, when an accurate plant model is unavailable, the adaptive controller fails to obtain a satisfactory tracking performance. The FEL does not suffer from this and still provides accurate tracking. This ability gives rise to the supposition that FEL control is suited for a wide range of applications, since in practice plants are often difficult to model accurately. The question remains whether the controller is commercially interesting. To answer this, we will evaluate if FEL has all the necessary features that were given in section 1.2:

– *Easy to use on an existing control system.* The only extension to the existing control system is the function approximator. When the control system is implemented in software, this requires minor changes and can be easily done.
– *Incorporation of prior knowledge in the design.* When the structure of the plant dynamics is known, the MLP network in the feed-forward controller, can be split up in several smaller MLP networks [Katic and Vukobratovic, 1995]. Each of these networks compensates for one specific part of the plant dynamics. Experiments showed that this speeds up learning considerably.
– *Stability is addressed.* It has been proven theoretically that for robot manipulators, the FEL controller converges [Kawato, 1990]. For other systems, stability has not been considered theoretically.

- *Good transient response.* During learning, the tracking error gradually converged to its minimum value. Like stability, the transient response has not been considered theoretically.
- *The function approximator is suited for control.* Much of the practical value of a learning controller depends on the type of function approximator that is used. In spite of the fact that the FEL controller finally achieves a high tracking performance, the learning behaviour is not optimal due to the MLP network [Dean *et al.*, 1991; Katic and Vukobratovic, 1995; Er and Liew, 1997]:
  - *Small memory requirement.* One of the nice properties of the MLP is that it is able to approximate high dimensional target functions with a low number of parameters. Therefore the amount of computer memory required for implementation is small.
  - *Computationally expensive.* Calculating the output of the MLP and adapting its weights involves a large number of complex computations. For some real-time control applications this type of neural network may therefore not be suited.
  - *The learning mechanism converges slowly and suffers from local minima.* The learning mechanism is easily trapped in a local minimum. Which local minimum the network weights end up in, depends on the initial weights of the network. Therefore, it is necessary to perform several training experiments with different sets of initial weights, to obtain an acceptable tracking accuracy.
  - *Good generalising ability.* An advantage of the fact that the input-output relation can only be adapted globally is that the MLP tends to have a good generalising ability when training is performed properly. When a motion system has to perform low-velocity motions, the FEL controller tends to have a poor performance. This is caused by the fact that an MLP network has difficulties with learning highly correlated data [Hrycej, 1997]. When the data is highly correlated, the network tends to fit the last presented data, resulting in a poor generalising ability.
  - *The smoothness of the approximation is not completely controllable.* The number of parameters of an MLP determines maximum accuracy of the approximation. It does not guarantee a certain smoothness. By learning, the MLP may approximate the target function very roughly in one part of the input domain and very accurately in another.

Looking at the properties above, we may conclude that, in case its learning behaviour is improved, FEL control has commercial potential. Different approaches exist to overcome the problems of the FEL controller. In the following, we will briefly present three methods. The first two methods alter the structure of the

learning controller, whereas in the last approach another type of function approximator is used.

Firstly, one can improve the learning behaviour by selecting different inputs for the function approximator. In [Gomi and Kawato, 1993; Er and Liew, 1997] the error signal was added as an input to the function approximator, changing the learning controller from a pure feed-forward controller into a feed-forward / feedback controller (figure 1.4). Experiments showed that this learning controller overcomes some of the problems of the original FEL controller.

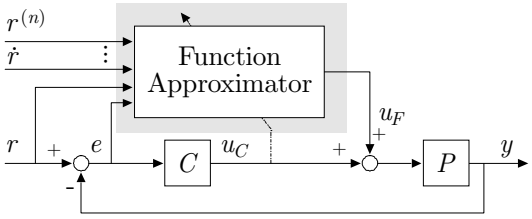



*Figure 1.4: Feedback Error Learning*

A second method is to use multiple feed-forward controllers, each trained to perform a specific task [Jacobs and Jordan, 1993]. A supervisory neural network learns which feed-forward controller is used for which task. This learning controller was tested on a manipulator that had to perform motions with objects of different weight. After learning, each of the feed-forward controllers had learned to move a specific object. The supervisory network had learned which feed-forward controller to use for which object.

Since the difficulties of FEL control are mainly caused by the MLP network, an obvious approach is to look for different function approximators. In [Kraft and Campagna, 1990; Ananthraman and Grag, 1993] the MLP network is replaced by a Cerebellar Model Articulation Controller (CMAC) network [Albus, 1975]. The CMAC network belongs to the class of neural networks that employ basis functions. In case of the CMAC network, the basis functions consist of piecewise polynomial functions that have a value unequal to zero on a compact part of the input space only. The basis functions are distributed such that at each point in the input space $\rho$ basis functions overlap. The parameter $\rho$ is known as the generalisation parameter and can be chosen by the designer. The output of the CMAC network is a weighted sum of the basis function evaluations. Learning is performed by adapting the weights of the network, not the basis functions themselves. All this yields the following improvements:

– *Faster convergence.* Since learning takes place locally, only a small number of weights is adapted, which results in a fast convergence.

– *Ability to learn correlated data.* The locations of the basis functions are fixed. This is beneficial for the learning of correlated data.

– *No local minima.* The learning mechanism does not suffer from local minima [Ng, 1997].

However, a disadvantage is that the controller designer has to choose the distribution of the basis functions. This requires some prior knowledge of the desired input-output mapping and tuning of the basis function distribution may be necessary before an acceptable performance is obtained. Experiments showed that replacing the MLP network by a CMAC network gives a superior learning behaviour and more accurate tracking.

## 1.4 Learning Feed-Forward Control

In this thesis a learning control system is considered that has the same structure as the FEL controller (figure 1.2). However, the feed-forward part of the learning controller is implemented as a B-spline network (BSN) (see appendix A) instead of an MLP. This type of FEL control has been named Learning Feed-Forward Control (LFFC) [Luenen, 1993; Starrenburg *et al.*, 1996]. The BSN approach is similar to the CMAC approach. Like the CMAC network, the BSN uses basis functions, known as B-splines, for approximation.

In the design of the LFFC the following parameters of the BSN have to be chosen:

1. *The inputs of the BSN.* For motion systems, the inputs of the BSN consist of the reference position, $r$, and derivatives thereof (see figure 1.2). Chapter 3 answers the question what inputs should be chosen such that the inverse system can be expressed as a function of the inputs.
2. *The B-spline distribution on each of the inputs of the BSN.* The output of a BSN is a weighted sum of B-spline evaluations. Therefore, the number of B-splines and their locations determine the accuracy of an approximation. Smooth target signals can be accurately approximated with a low number of "wide" B-splines. Strongly fluctuating signals require a large number of "narrow" B-splines (see figure 1.5).

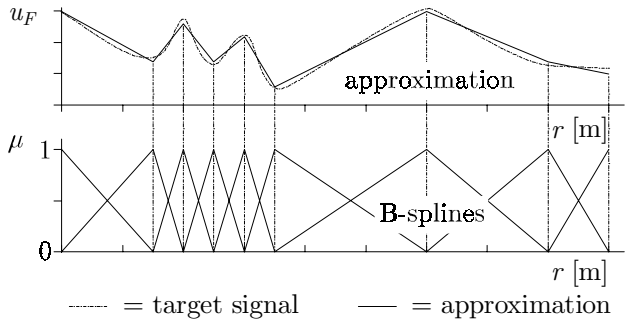



*Figure 1.5: BSN mapping*

A proper choice of the B-spline distribution can be made on the basis of knowledge of the inverse plant and of the disturbances, or by means of an iterative procedure that uses experimental data. These matters will be discussed in chapter 4.

3. *Selection of the learning mechanism.* Learning, i.e. adaptation of the network weights, can either be done after each sample, which is known as *on-line learning*, or after a motion has been completed, known as *off-line learning*. The on-line learning rule is

$$\Delta w_i = \gamma \mu_i(\boldsymbol{r}) e(\boldsymbol{r}) \tag{1.3}$$

and the off-line learning rule

$$\Delta w_i = \gamma \frac{\sum_{j=1}^{N_s} \mu_i(\boldsymbol{r}_j) e\left(\boldsymbol{r}_j\right)}{\sum_{j=1}^{N_s} \mu_i(\boldsymbol{r}_j)} \tag{1.4}$$

with,

$\boldsymbol{r}_j$: the input of the BSN. In LFFC the input consists of the reference position and derivatives thereof, $\boldsymbol{r}_j = \left\{r_j, \dot{r}_j, ...\right\}$;

$\mu_i(\boldsymbol{r}_j)$: the membership of the $i$-th B-spline, $\mu_i(\boldsymbol{r}_j) \in [0,1]$;

$\Delta w_i$: the adaptation of the weight of the i-th B-spline;

$\gamma$: the learning rate, $0 < \gamma \leq 1$;

$e(\boldsymbol{r}_j)$: the approximation error made by the network, in LFFC $e(\boldsymbol{r}_j)$ is the output of the feedback controller, $u_C$;

$N_s$: the number of input samples.

4. *Selection of a learning rate.* The larger the learning rate, the faster the convergence of the learning mechanism. However, a large learning rate makes the approximation more sensitive to noise and may cause instability.

By using a BSN, we have gained the following advantages:

- *No local minima.* The output of the BSN is a linear function of the weights. This means that the learning mechanisms given in (1.3) and (1.4) do not suffer from local minima. This implies that the initial weights of the BSN do not influence the final tracking accuracy.
- *Local learning.* Since B-splines have a compact support, the input-output mapping of the BSN can be adapted locally. Training a new motion does not unnecessarily affect the ability to track previously learned motions. Furthermore, due to the compact support of the B-splines only a small number of the weights contribute to the output. During training, only these weights need to be adapted. This in contrast to the MLP, where all network weights were changed during learning. Therefore, the BSN converges much faster than the MLP.
- *Tuneable precision.* The smoothness of the input-output relation is determined by the choice of the B-spline distribution. Choosing B-splines that have a more compact support, enables the BSN to more accurately approximate high-frequency data. When, due to actuator limitations for example, a more smooth approximation of the data is required in a specific part of the input space, B-splines should be used that have a larger support or a higher order. In case of the MLP, the user is not able to specify the smoothness in different parts of the input space, instead the MLP allocates its own resources.

The advantages listed so far also hold for the CMAC network. We have chosen to use the BSN for reasons of simplicity. In the CMAC network the basis-function distribution is such that at each input value, exactly $\rho$ basis function have an evaluation larger than 0. Several distributions satisfy this [Brown and Harris, 1994] each resulting in a different approximation. The user has to choose which distribution is most appropriate. Furthermore, refining an approximation by adding an extra knot (an extra basis function), results in a different basis function distribution. This means that the new basis function distribution is not able to exactly reproduce the approximation before refinement. This may be undesired. The distribution of the basis functions in a BSN is straightforward.

The principal drawback of BSNs (and of other NNs with basis functions) is that the number of network weights grows exponentially with the dimension of the input space. The number of network weights in an $N$-dimensional BSN is given by:

$$N_t = \prod_{j=1}^{N} N_j \tag{1.5}$$

Where $N_j$ is the number of B-splines defined on input $j$. Since accuracy considerations may require many B-splines and thus large values of $N_j$, e.g. when a highly non-linear function is to be approximated, these networks are impractical when the number of inputs is large. This so-called *curse of dimensionality* brings about the following problems [Brown and Harris, 1994; Bossley *et al.*, 1996]:

- *Large number of network weights.* If the plant dynamics have highly non-linear components, a highly non-linear function has to be mapped by the B-spline network. A network that is able to map these non-linearities accurately would take up a great deal of computer memory. In practice, memory resources are limited, therefore, network complexity and mapping accuracy have to be traded off against each other.
- *Large training sets.* When performing a certain reference motion, only the network weights indexed by the network inputs are updated. To adapt a large number of network weights therefore requires a large number of training motions. This will lead to large training times if the network is trained on-line, i.e. during control.
- *Poor generalising ability.* The compensation of a non-linearity may require narrow B-splines for reasons of accuracy. However, with narrow B-splines very different network output signals may occur with trajectories that are 'close to each other'. Hence the large training sets mentioned above all need to be presented before beneficial effects will become noticeable.

It can be seen that the curse of dimensionality may seriously harm the commercial value of the LFFC. Therefore, effort should be made to overcome the curse of dimensionality. In chapter 4 several methods are presented that may be able to do this.

## 1.5 Illustrative Application: a Linear Motor Motion System

An interesting application of learning control, is a permanent magnet synchronous linear motor. Linear motors [Nasar and Boldea, 1987; Basak, 1996] are often designed to perform linear motions with sub-millimetre accuracy, e.g. for laser-cutting, scanning, or pick-and-place tasks [Otten *et al.*, 1997]. The linear motor considered here is the so called Linear Motor Motion System (LiMMS), which is manufactured by Philips. The motor configuration consists of a base plate covered with permanent magnets, and a sliding part, the translator that holds the electric coils and their iron cores, see figure 1.6. By applying a three-phase current to three adjoining coils of the translator, a sequence of attracting and repelling forces between the poles and the permanent magnets will be generated. This results in a relative motion between the base plate and the translator. The basic behaviour of the motor is that of a moving mass; the mass of the translator with a (dummy) load, $m_L$, is 37 [kg].

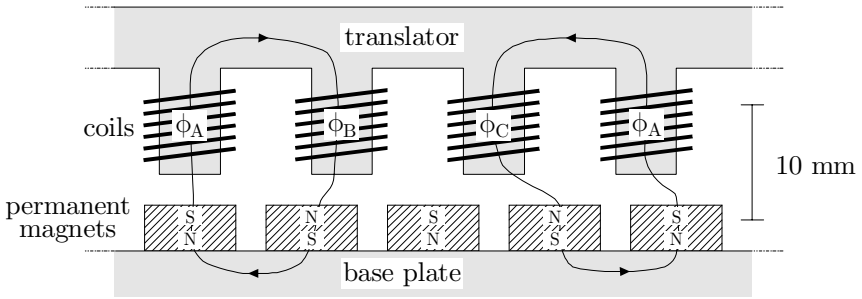



*Figure 1.6: Working principle of the LiMMS linear motor. The indicated lines are the flux-lines of the permanent magnets, $\phi_A$, $\phi_B$, and $\phi_C$ are the phases of the 3-phase motor current*

During operation the translator experiences a relatively high-frequency disturbance force, known as the force ripple. The force ripple is caused by two phenomenons:

– Firstly, a strong magnetic interaction exists between the permanent magnets on the base plate and the iron cores that are mounted in the coils of the translator. The iron cores are employed to obtain a high motor efficiency. This disturbance force, known as *cogging*, tries to align magnets and iron cores to stable ('detent') positions of the translator. The cogging force depends only on the relative position of the translator with respect to the magnets; it is independent of the motor current and is observed in a zero-current situation as well. For reasons of simplicity, in simulations the cogging force, $F_C$ [N], is modelled as a sinusoidal shaped input disturbance

with an amplitude of 10 [N] and a pitch of $1.6\,e^{-2}$ [m] that depends on the motor position, $y$ [m]:

$$F_C(y) = 10\sin\left(1.6e^{-2}y\right) \tag{1.6}$$

– Secondly, a force ripple can be generated by errors in the commutation, i.e. the way the current is applied to the coils. When moving a coil through a varying electro-magnetic field, a back EMF is generated that depends on the velocity of the coil in a linear way. If the current applied to the coils is not proportional to the back EMF a force ripple will result that depends on the position and the velocity of the translator. The back EMF can be computed from a detailed model of the translator and the magnets in the base plate. This requires the position and magnetic properties of the permanent magnets to be known accurately. In case of the LiMMS one has chosen to use relatively low-cost magnets that have large magnetic tolerances. Furthermore, the magnets are not placed with the highest possible accuracy. This makes it difficult to obtain the model needed to compute the commutation and thus a force ripple results. This effect has not been incorporated in the simulation model.

Other disturbances that act on the LiMMS, are friction phenomenons in the ball bearings that are present between the translator and the guiding rails it rests on. It is assumed that in the real set-up the friction characteristics can be described by the Stribeck curve [Armstrong-Helouvry *et al.*, 1994], i.e. the friction consists of Coulomb friction, stiction and viscous friction. In the simulation model, only viscous friction, $d_L$=10 [Nms$^{-1}$], is considered:

$$F_V(\dot{y}) = 10\dot{y} \tag{1.7}$$

This results in the simulation model given in figure 1.7.

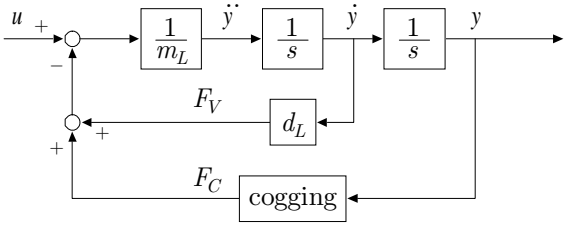



*Figure 1.7: Model of the LiMMS*

In chapter 2 and 3 this application will be used in simulation studies. Chapter 5 presents results with a real LiMMS.

## 1.6 Outline of the Thesis

Chapter 2: Repetitive motions

Long term stability is an essential property for a learning controller. To be able to analyse the stability of the LFFC, a special type of motion is considered, i.e. repetitive motions. In this case, the LFFC becomes similar to two other types of learning control: Iterative Learning Control (ILC) and Repetitive Control (RC). Rigorous stability analyses exist both for ILC and RC. Based on this, a stability analysis of LFFC will be performed. The stability analysis gives insight in the learning behaviour of LFFC and provides rules for a proper choice of the design parameters.

Chapter 3: Non-Repetitive Motions – Input Selection and Convergence

In this chapter we derive conditions that a motion system must fulfil, before it can be controlled by an LFFC. From this, we also gain insight in the proper choice of the inputs of the BSN. Stability of the LFFC will be addressed. The results of the stability analysis, performed in chapter 2, are used to examine stability of an LFFC that performs random motions. It shows that without additional stability measures, long term stability cannot be guaranteed. Therefore, we propose a new stability measure, referred to as regularisation, which is able to stabilise LFFC. Simulations are presented that confirm the operation of this.

Chapter 4: Non-Repetitive Motions – Parsimonious LFFC

It appears that in general the BSN will have multiple inputs. This may cause problems due to the curse of dimensionality. We will show that replacing the single BSN by multiple BSNs may overcome these problems. The resulting LFFC is known as a parsimonious LFFC. Finally, we show how the parsimonious LFFC can be trained.

Chapter 5: Design and Applications

Here, the theoretical insight gained in the previous chapters is used to formulate a design procedure for LFFC. This procedure will be used for two practical applications. The first system under control is the LiMMS, presented in section 1.5. Next, experiments will be performed with a part of a flight simulator, i.e. the Fokker Control Loading System.

Chapter 6: Discussion

We end up discussing the results of the previous chapters and draw conclusions.

# 2 Repetitive Motions

## 2.1 Introduction

It is of vital importance that a learning controller features long term stability. One way to guarantee this is to stop learning once the desired tracking performance has been obtained. However, during operation the plant characteristics may change due to e.g. wear or temperature influences. It may thus be desirable to learn continuously, such that the learning controller is able to adapt to the changing plant. Stability must then be ensured on the basis of a theoretical analysis. A large part of literature on learning control deals with stability. As stated in the previous chapter, not all analyses deal with stability from a practical point of view. Often the transient behaviour is not considered. This implies that large tracking errors are allowed in the intermediate phase of learning. In our opinion, a proper stability analysis should:

- *Indicate whether the learning controller is stable or not.*
- *Predict the transient response.* When the learning controller is applied as an add-on device for an existing controller, it is expected to improve the tracking performance. Growing tracking errors during learning may cause loss of production or may even harm the plant. To prevent this, the stability analysis should also indicate how the tracking error converges to its final value.
- *Give insight in the stability robustness.* As learning is performed continuously, the learning controller should remain stable even when the plant characteristics change. The stability analysis should indicate how much

the plant characteristics may change before the learning controller becomes unstable.

– *Give guidelines for the choice of values for the design parameters.* The values of the design parameters of the learning controller should be chosen such that it features fast and stable learning with a desired transient response. The stability analysis should not only indicate whether this is the case or not, but should also give guidelines on how the values should be chosen such that this is obtained.

The stability analysis of LFFC, presented in section 2.5, is based on the stability analyses of two other learning control schemes which are closely related to it, i.e. Iterative Learning Control (ILC) and Repetitive Control (RC). Both ILC and RC have been developed to control plants that have to perform one and the same motion over and over again. In section 2.2 and 2.3, we will shortly review ILC and RC. Furthermore, matters like stability, learning behaviour and design will be discussed. Section 2.4 will show that the structure of an LFFC that is designed for repetitive motions is similar to the structure of ILC and RC. Therefore, insight in the role of the design parameters of ILC and RC can be used in the stability analysis of LFFC, section 2.5. Finally, the results of the stability analysis will be verified by means of simulations in section 2.6.

## 2.2 Iterative Learning Control

### 2.2.1 Original ILC Scheme

Many industrial motion systems repeatedly perform a specific motion. Think, for example, of pick-and-place machines or welding robots. A learning control system that is suited for this type of applications is ILC [Arimoto *et al.*, 1984; Kawamura *et al.*, 1988; Moore, 1992,a; Moore *et al.*, 1992,b; Bien and Xu, 1998]. A more detailed description of the field of interest of ILC is given by the following axioms [Arimoto, 1998]:

1. A plant repeatedly performs a specific motion that ends in a fixed duration, $T_p>0$ [s].
2. The plant dynamics are time-invariant.
3. The desired output, $y_d(t)$ $(t\in[0,T_p])$, is given a priori.
4. For each trial, the initial states are the same.

5. The plant output, $y(t)$, is observable.
6. There exists a unique input, $u_d(t)$, that yields $y_d(t)$.

The original solution to this problem, proposed in [Arimoto *et al.*, 1984] is depicted in figure 2.1. The steering signal for the $k$-the trial, $u_F^k(t)$, is stored in a memory buffer of length $T_p$ [s]. During motion, the tracking error, $e^k(t)$, is recorded. At the end of the trial $e^k(t)$ is used to modify the steering signal in such way that in the following trial a smaller tracking error will result. This is done by filtering $e^k(t)$ with the learning filter, $L$, and adding the result to $u_F^k(t)$, which yields $u_F^{k+1}(t)$. Note that this control system is open loop in nature.

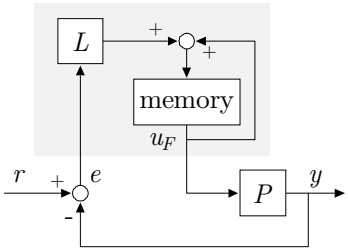



*Figure 2.1: Original ILC*

The (non-linear) plant, $P$, is given by:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}u \\ y &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{2.1}$$

in which $\boldsymbol{f}$ is globally Lipschitz continuous, but unknown. In the original ILC [Arimoto *et al.*, 1984], $L$ was implemented as a pure differentiator, which gives the following learning rule:

$$\boldsymbol{u}_F^{k+1}(t) = \boldsymbol{u}_F^k(t) + \boldsymbol{\Gamma}\dot{e}^k(t) \tag{2.2}$$

Analysis shows [Arimoto *et al.*, 1984; Heinzinger *et al.*, 1992] that learning converges in the sense of the $\lambda$-norm,

$$\left\| u_d(t) - u_F^{k+1}(t) \right\|_\lambda \leq \rho \left\| u_d(t) - u_F^k(t) \right\|_\lambda \tag{2.3}$$

with $\rho<1$, if

$$\left\| I - \boldsymbol{CB\Gamma} \right\|_{\infty} < 1 \tag{2.4}$$

The λ-norm is defined as:

$$\left\| f(t) \right\|_{\lambda} = \max_{t \in [0,T]} e^{-\lambda t \left\| f(t) \right\|_{\infty}} \tag{2.5}$$

where $\lambda \in \mathbb{R}^{+}$.

**Remark 2.1 (λ-norm)** Note that by choosing $\lambda = 2$, we obtain the following norm,

$$\left\| f(t) \right\|_{\lambda=2} = \max_{t \in [0,T_p]} e^{-2t \left\| f(t) \right\|_{\infty}} \tag{2.6}$$

which is unequal to the common $H_2$ norm, $\left\| f(t) \right\|_2$.

In literature on ILC, the λ-norm is often used to prove that learning converges. However, a learning controller that converges in the sense of the λ-norm does not necessarily have a good transient response. This is caused by the fact that values at the end of the trial ($t$ close to $T_p$) are weighted much less than those at the beginning ($t$ close to 0) [Amann and Owens, 1996]. The tracking error at the beginning of the trial decreases from trial to trial, while the tracking error at the end builds up. As stated in the previous chapter, this type of learning behaviour is unwanted.

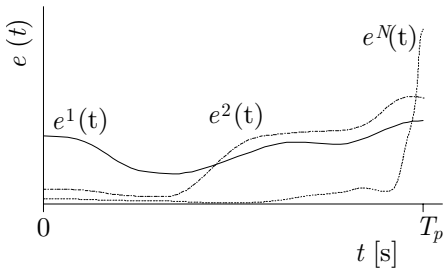



*Figure 2.2: Typical learning behaviour of the original ILC*

This problem can be overcome by considering the trial length to be infinite [Amann and Owens, 1996]. From a practical point of view it seems strange to assume the length of a task, which is to be repeatedly performed, to be infinite. However, mathematically this assumption is advantageous because:

1. It enables the use of frequency domain analysis.
2. If the trial length is long compared to the time constants of the system, the infinite trial length is a good approximation of the real situation.
3. A learning mechanism that converges on infinite trial length, must also converge on finite trial length.
4. It gives insight in the properties of the controller which may be hard to obtain by finite time analysis.

### 2.2.2 Current Cycle Feedback

The ILC of (2.2) is an open-loop controller. This means that the controller is not able to compensate for random disturbances. If these are significant, compared to the reproducible disturbances, it is not possible to obtain a high tracking performance. Another problem may occur when the plant is unstable. To overcome these difficulties [Chen *et al.*, 1996; Chien and Liu, 1996], a feedback controller is added to the ILC (figure 2.3). Since the input of the plant does not consist of the feed-forward signal only, which depends on information gathered in previous trials, but also of the feedback signal, this type of ILC is known as current cycle feedback. For sake of simplicity, in the following only LTI SISO plants will be considered.

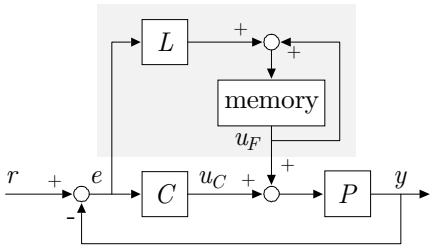



*Figure 2.3: ILC incorporating feedback controller*

In the frequency domain the learning mechanism is given by:

$$U_F^{k+1} = U_F^k + L\,E^k \tag{2.7}$$

In which, $U_F^k$ is the Fourier transform of the feed-forward signal in run $k$ and $E^k$ is the Fourier transform of the corresponding tracking error. Learning converges, i.e. the system remains stable, if the feedback loop is stable and the following condition is satisfied:

$$\forall \omega \in \mathbb{R}: \quad \left\| U_F^{k+2}(j\omega) - U_F^{k+1}(j\omega) \right\|_\infty < \left\| U_F^{k+1}(j\omega) - U_F^{k}(j\omega) \right\|_\infty \tag{2.8}$$

The left hand side of (2.8) can be written as:

$$\left\| U_F^{k+2} - U_F^{k+1} \right\|_\infty = \left\| U_F^{k+1} + LE^{k+1} - U_F^{k} - L\,E^{k} \right\|_\infty \tag{2.9}$$

Where:

$$\begin{aligned} E^k &= R - Y^k \\ &= R - \left( \frac{CP}{1+CP} R + \frac{P}{1+CP} U_F^k \right) \\ &= \frac{1}{1+CP} R - \frac{P}{1+CP} U_F^k \end{aligned} \tag{2.10}$$

Substituting (2.10) in (2.9) gives:

$$\begin{aligned} \left\| U_F^{k+2} - U_F^{k+1} \right\|_\infty &= \left\| U_F^{k+1} - U_F^{k} - \frac{LP}{1+CP} \left( U_F^{k+1} - U_F^{k} \right) \right\|_\infty \\ &= \left\| \left( 1 - \frac{LP}{1+CP} \right) \left( U_F^{k+1} - U_F^{k} \right) \right\|_\infty \\ &\leq \left\| \left( 1 - \frac{LP}{1+CP} \right) \right\|_\infty \left\| \left( U_F^{k+1} - U_F^{k} \right) \right\|_\infty \end{aligned} \tag{2.11}$$

It can thus be concluded that learning converges if:

$$\left\| 1 - \frac{LP}{1+CP} \right\|_\infty < 1 \tag{2.12}$$

In the Nyquist plot this means that $1 - \dfrac{LP}{1+CP}$ must lie in the unit circle. In order to examine in what way $L$ must be chosen, we first present a Nyquist plot of (2.12) for $L$=1. In figure 2.4 Nyquist plots of $\dfrac{P}{1+CP}$ and $1 - \dfrac{P}{1+CP}$ that are typical for motion systems are shown.

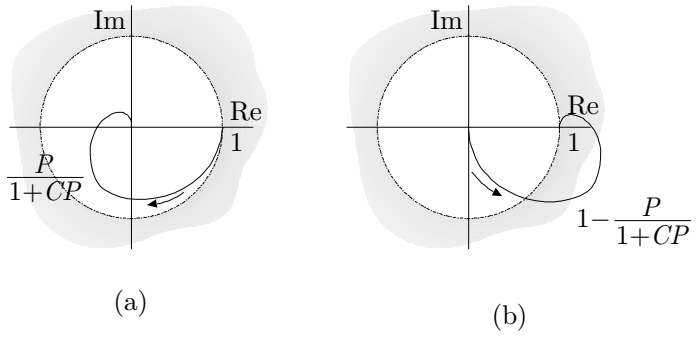



*Figure 2.4: a) Typical Nyquist plot of $P/(1+CP)$ b) Typical Nyquist plot of $1-P/(1+CP)$*

The Nyquist plot of $1-\frac{P}{1+CP}$ is outside the unit circle when $\mathrm{Re}\left(\frac{P}{1+CP}\right) < 0$, which occurs when the phase of $\frac{P}{1+CP}$ lies between $-0.5\pi$ and $-1.5\pi$. Furthermore, a too large magnitude of $\frac{P}{1+CP}$ causes violation of (2.12). The learning filter $L$ has to compensate for the phase and/or the magnitude of $\frac{P}{1+CP}$. In figure 2.5a,b the effect of such an $L$ on the Nyquist plot is given.

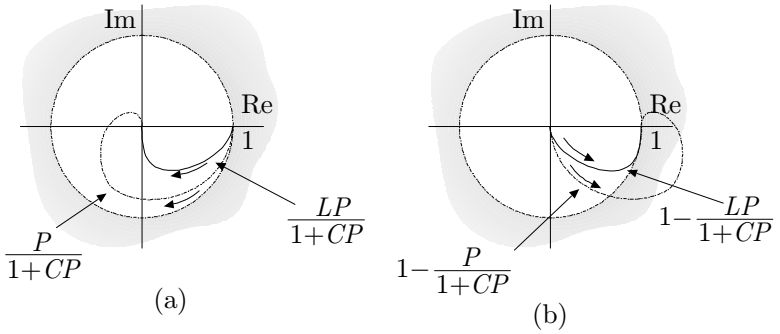



*Figure 2.5: a) Typical Nyquist plot of $LP/(1+CP)$ b) Typical Nyquist plot of 1-$LP/(1+CP)$*

To design such an $L$, detailed knowledge of the plant is required. For low frequency dynamics, a competent model of the plant often exists. However, identification and modelling of high frequency dynamics is difficult and may lead to an incompetent

model. This could result in an $L$ that compensates well for low frequencies but does not compensate appropriately for all high frequencies and therefore causes unstable behaviour.

### 2.2.3 Robustness Measures

We will discuss two methods that make ILC robust for unmodelled high frequency dynamics. In [Hara *et al.*, 1988] it is proposed to incorporate a low-pass $Q$-filter in the memory loop. The $Q$-filter is designed such that it suppresses the frequency components at which the plant model is inaccurate. The lower frequencies, at which the model is accurate, are passed. The $Q$-filter is either placed before the memory [Roover, 1996], as in figure 2.6, or in the memory feedback loop [Moore *et al.*, 1992,b].

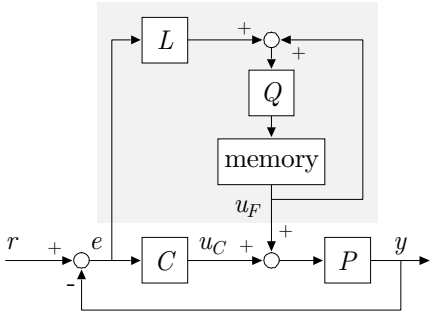



*Figure 2.6: ILC with low-pass Q-filter*

The learning mechanism becomes:

$$U_F^{k+1} = Q\left(U_F^k + L\,E^k\right) \tag{2.13}$$

Using the same approach as before, the following stability criterion can be derived:

$$\left\| Q\left(1 - \frac{LP}{1+CP}\right)\right\|_\infty < 1 \tag{2.14}$$

In figure 2.7a,b the effect of the $Q$-filter is presented graphically. Firstly, $Q$ is chosen equal to 1, which gives $1 - \dfrac{LP}{1+CP}$. Apparently, $L$ is not designed properly

since $1-\dfrac{LP}{1+CP}$ lies outside the unit disc for some frequencies and learning will thus diverge. Next, $Q$ is chosen as a low pass filter. By adding $Q$, the amplitude of $1-\dfrac{LP}{1+CP}$ at the high frequencies is decreased such that the whole Nyquist plot now lies in the unit circle.

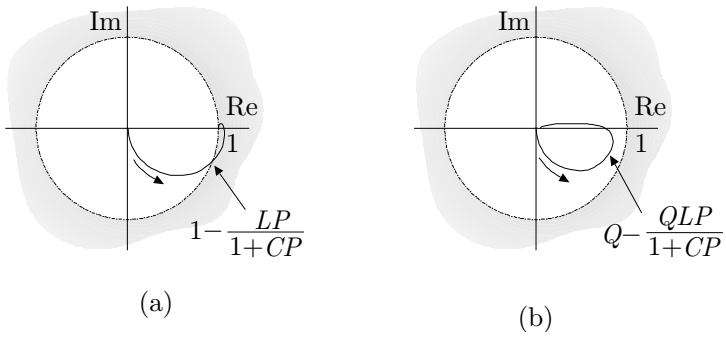



*Figure 2.7: a) Typical Nyquist plot of 1-LP/(1+CP)*
*b) Typical Nyquist plot of Q-QLP/(1+CP) (Q low pass)*

By adding the $Q$-filter, $L$ no longer needs to compensate at high frequencies exactly. This makes the ILC more robust for unmodelled high frequency dynamics. However, a disadvantage of incorporating the $Q$-filter is that a zero tracking error can no longer be obtained. The tracking error that remains can be calculated as follows. Firstly, when (2.14) is satisfied, learning converges which means that:

$$\lim_{k\to\infty} U_F^{k+1} = U_F^k \tag{2.15}$$

Substituting (2.15) in the learning rule (2.13) gives:

$$\lim_{k\to\infty} U_F^k = \frac{QL}{1-Q} E^k \tag{2.16}$$

Using (2.10) we can calculate the tracking error that remains when learning has converged:

$$\begin{aligned}
&\lim_{k\to\infty} E^k = \frac{1}{1+CP}R - \frac{P}{1+CP}\left(\frac{QL}{1-Q}\right)E^k \\
\Leftrightarrow &\lim_{k\to\infty} E^k\left(1+\frac{QPL}{(1+CP)(1-Q)}\right) = \frac{1}{1+CP}R \\
\Leftrightarrow &\lim_{k\to\infty} E^k = \frac{(1-Q)}{(1+CP)(1-Q)+QPL}R
\end{aligned} \tag{2.17}$$

The $Q$-filter is designed as a low-pass filter, which means that $Q\approx 1$ for low frequencies, while $Q\approx 0$ for high frequencies. Inspecting (2.17) we can see that for low frequencies:

$$\lim_{k\to\infty} E^k \approx 0 \tag{2.18}$$

and for high frequencies:

$$\lim_{k\to\infty} E^k = \frac{1}{1+CP}R \tag{2.19}$$

The bandwidth of the $Q$-filter should thus be chosen larger than or equal to the closed loop bandwidth we want to obtain.

An alternative approach was taken after noting that the original ILC features local (point-to-point) adaptation of the feed-forward signal. This means that at each sample instant, only the feed-forward signal belonging to that instant is adapted. However, the feed-forward signal that has to be learned is often continuous and changes relatively little between 2 samples. This suggests that the adaptation of the feed-forward signal at the current instant could well be used to adapt the feed-forward signal at neighbouring instants as well. One way to accomplish this kind of adaptation is to define the feed-forward signal as an *integral expression*, consisting of a known kernel and an unknown influence function [Horowitz *et al.*, 1991; Messner *et al.*, 1991]:

$$u_F(t) = \int_0^{T_p} K(t,\tau)\; c(t,\tau)d\tau \tag{2.20}$$

In which $K(t,\tau)$ is the known kernel and $c(t,\tau)$ is the unknown influence function. For a good approximation of the feed-forward signal, the kernel function has to be a smooth function. A typical example of a suitable kernel function is the Gaussian distribution function, which is defined by:

$$K(t,\tau) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(t-\tau)^2}{\sigma^2}} \tag{2.21}$$

For $\sigma$=1 the Gaussian distribution has the following shape:

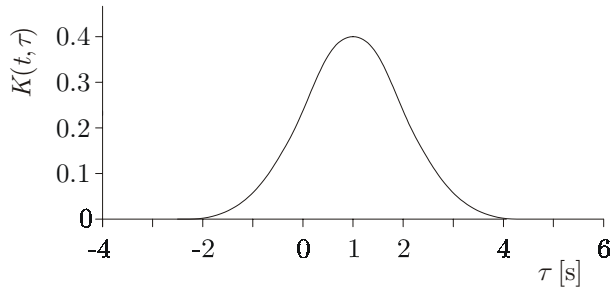



*Figure 2.8: Gaussian kernel function ($\sigma$=1,t=1);*

In this type of ILC, training is done by adapting the influence function. A learning rule that can be used for control of robot manipulators is [Horowitz *et al.*, 1989]:

$$\frac{\partial c(t,\tau)}{\partial t} = K(t,\tau)\,\dot{e}(t) \tag{2.22}$$

In which $\dot{e}(t)$ is the velocity error of the robot manipulator. Note that a Gaussian kernel is non-zero over its entire input domain. Therefore, execution of learning rule (2.22) at a particular sample instant implies that the feed-forward signal will be adapted over the full domain $[0, T_p]$. We refer to this as *global adaptation*. Due to the fact that adaptation is no longer local, learning may converge faster than in case of point to point adaptation. Experiments confirmed this.

Analyses and experiments showed that the use of kernel functions improved the robustness properties of the ILC controller. [Horowitz, 1993] indicated that this is to be expected, as the point to point adaptation of the feed-forward signal in the original ILC is the cause of unstable behaviour. The use of kernel functions improves the robustness because the 'width' $\sigma$ of the Gaussian kernel function has the character of a low pass filter; it determines the frequencies that can be present in a signal consisting of a combination of such kernel functions. Both the known kernel and the learning rule are such a combination.

### 2.2.4 Design Procedures

Several design procedures exist [Longman, 1998] of which we will very briefly discuss the following. In [Kavli, 1992], a heuristic design procedure is proposed. This procedure comprises the following steps:

1. Construct a model that describes the plant accurately up to an frequency $\omega_Q$.
2. Manually design the learning filter, $L$, such that:

$$\forall \omega \in \left[0, \omega_Q\right] : L\left(j\omega\right) \approx \left(\frac{P\left(j\omega\right)}{1 + C\left(j\omega\right) P\left(j\omega\right)}\right)^{-1} \tag{2.23}$$

   where $\omega_Q$ is frequency, up to which perfect tracking is desired. This requires a model of the plant that is accurate at least up to $\omega_Q$.

2. Choose Q as a low-pass filter with zero phase and a cut-off frequency $\omega_Q$ such that (2.14) is satisfied.

A more formal design method, based on an $\mathrm{H}_\infty$ approach, is proposed in [Amann and Owens, 1996; Roover, 1996; Roover, 1997]. The design procedure consists of the following steps:

1. Choose low-pass $Q$-filter with cut-off frequency $\omega_Q$.
2. By means of $H_\infty$ synthesis find an $L$ such that

$$L\left(jw\right) = \arg\min \left\| Q\left(jw\right) \left(1 - \frac{L\left(jw\right) P\left(jw\right)}{1 + C\left(jw\right) P\left(jw\right)}\right) \right\|_\infty \tag{2.24}$$

The advantages of this method over the heuristic design procedure are that an $L$ is obtained that minimises (2.14), and thus has the largest possible convergence rate, and that this method can be applied to multivariate ILC. However, the method does not take uncertainty in the plant model into account. In [Roover, 1996] a design method is proposed that is based on $\mu$-synthesis and that does allow plant uncertainty.

### 2.2.5 Extension of the Field of Application

At the begin of this section, 6 axioms are given that describe the type of problems ILC considers. Not all practical applications fulfil these 6 axioms. Therefore, extensive research is undertaken to modify ILC, such that it can be applied to these applications.

Axiom 4 states that for each trial, the initial conditions are the same. However, this may not always be the case. In [Lee and Bien, 1996; Lee and Bien, 1998], the robustness for a bounded but random initial error is examined of an ILC with a PD-type $L$. It was shown that learning still converges and that the final tracking error depends on the bound of the initial error.

ILC considers one specific motion that is to be repeated over and over again. However, in practice some motion systems have to perform non-repeating motions. In this case, ILC can be applied in the following way. For a number of repeating trajectories we design ILCs and train them until the feed-forward signal has converged. The trajectories and the feed-forward signal that the ILC has learned are stored in a database. When a new motion must be performed that has not been presented before, the feed-forward signal is constructed from the feed-forward signals in the database. In [Xu and Song, 1998] three different relations between the new motion and previously presented motions are considered:

- The new motion has the same time scale, but a different spatial scale [Xu, 1997].
- The new motion has the same spatial scale, but a different time scale [Xu, 1998].
- Both the time and the spatial scale differ.

In [Arif *et al.*, 1999] no direct relation between the new motion and the stored motions is assumed. Instead, their similarity is calculated. The result is a similarity index $J_s$, $0 \leq J_s \leq 1$. The feed-forward signal of the new motion is set to the feed-forward signal of the most similar stored motion multiplied by $J_s$.

## 2.3 Repetitive Control

In ILC, it was assumed that in each run the initial states are the same. When continuously performing a repetitive task, this is not the case. The states of the system at the end of the $k^{\text{th}}$ trial are equal to the initial states of the $k{+}1^{\text{th}}$ trial. Repetitive Control (RC) [Hara *et al.*, 1988; Tomizuka *et al.*, 1989; Steinbuch and Schootstra, 1996] is a type of learning control that has been created for these motions. The applications RC considers can thus be described by the axioms given in the previous section, except axiom 4. The plant may furthermore be subject to periodic disturbances with period $nT_p$, where $n{\in}\mathbb{N}$ and $T_p$ is the period of the

reference trajectory. According to the internal model principle [Francis and Wonham, 1975] perfect tracking of a reference signal or perfect attenuation of a disturbance requires a model of the reference/disturbance generating system to be included in the controller. In case of periodic trajectories or disturbances, this means that the repetitive controller has to include a periodic signal generator, such as a memory loop. The structure of the repetitive controller is presented in figure 2.9. Repetitive Control (RC) is very similar to ILC. It is to be expected that the learning controllers have comparable stability criteria and design procedures. We discuss RC to give the reader valuable insight in the learning behaviour when continuously performing a specific reference motion.

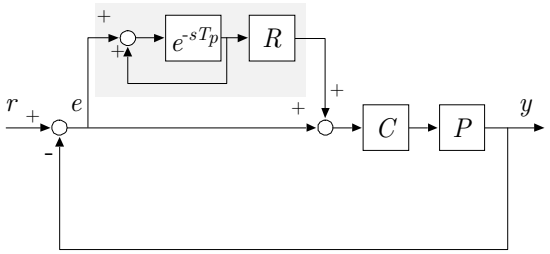



*Figure 2.9: Repetitive control scheme*

In figure 2.9 $R$ is a learning filter. In order to examine the stability of the RC, the system is represented in another way [Hara *et al.*, 1988], see figure 2.10.

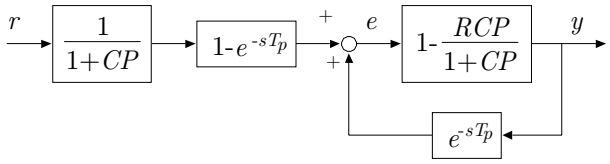



*Figure 2.10: Equivalent repetitive control scheme*

The above system is stable if:

- The system without RC is stable, which means that $\dfrac{1}{1+CP}$ is stable.
- The feedback loop is stable. According to the small gain theorem [Vidyasagar and Desoer, 1975] a sufficient condition for the loop to be stable is:

$$\left\| 1 - R\frac{CP}{1+CP} \right\|_\infty < 1 \tag{2.25}$$

Note that this stability criterion is almost the same as for ILC with an infinite trial length, given in (2.12). Therefore, it is to be expected that RC will also have difficulties with unmodelled high-frequency dynamics. The robustness measures for ILC, such as the low-pass $Q$-filter and the integral expression, have also been proposed for RC. Of course, the design procedures that are used in ILC can be applied to RC.

Next, we will study the behaviour of the RC by means of the sensitivity function, $S$. For the RC of figure 2.9, $S$ is given by:

$$S = \frac{1 - e^{-sT_p}}{1 - \left(1 - \frac{RCP}{(1+CP)}\right)e^{-sT}} \frac{1}{1+CP} \tag{2.26}$$

Next, we assume that the learning filter, $R$, is designed such that (2.25) is satisfied for all frequencies. From the previous section it follows that in this case:

$$R = \left(\frac{CP}{1+CP}\right)^{-1} \tag{2.27}$$

This yields the following $S$:

$$S = \left(1 - e^{-sT_p}\right)\frac{1}{1+CP} \tag{2.28}$$

Without RC, the sensitivity function is given by:

$$S = \frac{1}{1+CP} \tag{2.29}$$

The sensitivity function of an RC controlled system is thus equal to the sensitivity function without RC multiplied by $\left(1 - e^{-sT_p}\right)$. In figure 2.11, a Bode plot of $\left(1 - e^{-sT_p}\right)$ for $T_p$=0.1 [s] is shown.

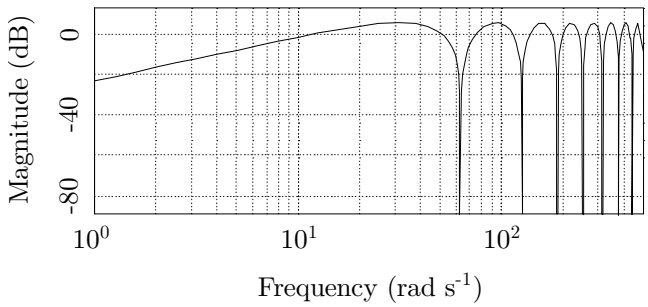



*Figure 2.11: Bode plot of* $\left(1-e^{-sT_p}\right)$

From figure 2.11 it can be concluded that the repetitive controller strongly attenuates disturbances that have an frequency $\omega{=}n2\pi T_p^{-1}$ [rad s$^{-1}$] where $n{\in}\mathbb{N}$. However, this is obtained at the expense of the suppression of disturbances at the intermediate frequencies, which are even amplified by the RC.

## 2.4 LFFC for Repetitive Motions

The main disadvantage of the BSN in the LFFC is the curse of dimensionality. This means that when the number of inputs increases, the number of weights grows exponentially, learning gets slower and the BSN looses its ability to generalise. The most effective method to prevent this is to keep the number of inputs of the BSN to a minimum. In the standard configuration, the input of the feed-forward controller consists of the reference position and derivatives thereof. When repetitive motions are considered only, as in ILC and RC, the desired feed-forward signal $u_F(r)$ becomes a function of the periodic motion time, $t \in [0, T_p]$, where, $T_p$ [s] is the period of the motion. In that case, it is beneficial to choose the periodic motion time as the only input (figure 2.12). This reduces the number of inputs and will be beneficial for the learning behaviour. In the following, this type of LFFC will be denoted as *time-indexed LFFC*.

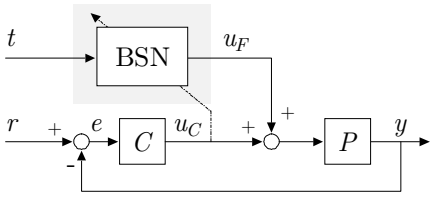



*Figure 2.12: Time-indexed LFFC*

Figure 2.13 shows the B-spline distribution in case of $N$ uniformly distributed $2^{\text{nd}}$ order B-splines defined on the input domain $[0, T_p]$ [s]. The basis functions have a sequence-based label $i \in \{1 \ldots N\}$. Basis functions 1 and $N$ have only half the width of the remaining basis functions.

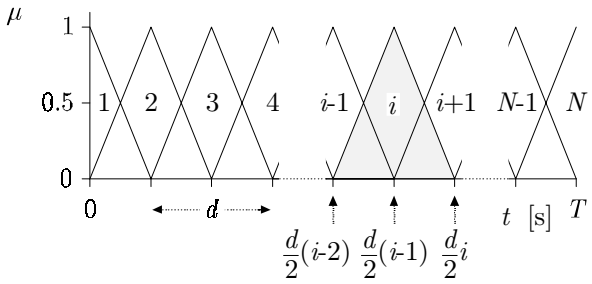



*Figure 2.13: B-spline distribution in case of time-indexed LFFC*

This type of B-spline distribution can be used when at the begin of a motion the initial states of the system are always the same, as is the case for ILC. In RC the motions are performed consecutively, which means that the states at the end of the $n^{\text{th}}$ motion are equal to the states at the begin of the $(n{+}1)^{\text{th}}$ motion. For this type motions it is desired to have a continuous feed-forward signal, which means that $u_F(0) = u_F(T_p)$. This can be obtained by making B-spline 1 and $N$ share the same weight. The B-spline distribution that results can graphically be represented as below.

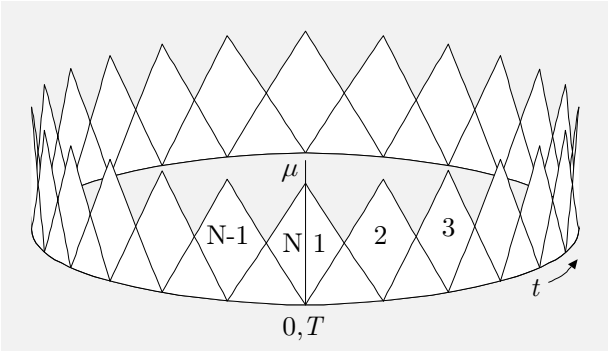



*Figure 2.14: Time-indexed B-splines for RC-type of motions*

Note that the structure of the time-indexed LFFC (figure 2.12) is the same as the structure of the ILC, when $L{=}C$ (figure 2.3) and as the structure of RC (figure 2.9), when $R{=}1$. When looking at the stability criteria of ILC (2.12) and RC (2.25), we can see that $L$ and $R$ have to compensate the phase shift and the magnitude of respectively $-\frac{P}{1+CP}$ and $-\frac{CP}{1+CP}$. If this is not realised, due to e.g. unmodelled dynamics, the ILC and RC will become unstable. Since the structure of the time-indexed LFFC is similar to the structure of ILC and RC, it is to be expected that this will also take place in LFFC. Experiments indicated that if the BSN is able to accurately approximate the frequency components for which $L$ and $R$ do *not* compensate for $-\frac{P}{1+CP}$ and $-\frac{CP}{1+CP}$ in a correct way, the LFFC becomes unstable [Velthuis *et al.*, 1996]. To prevent this, we have to choose the width of the B-splines in a proper way. Whether the BSN is able to accurately approximate a certain signal depends on the width of the B-splines that is chosen. In case 'wide' B-splines are chosen, the BSN can only store low-frequency signals (figure 2.15a). Using such a BSN to approximate a signal that has low-frequency as well as high-frequency components, will yield a good approximation of the low frequent part only (figure 2.15b). Decreasing the width of the B-splines enables the BSN to approximate the high-frequency part as well. The role of the B-splines in the BSN can thus be compared to the $Q$-filter in ILC and the kernel function in the integral expression (2.20).

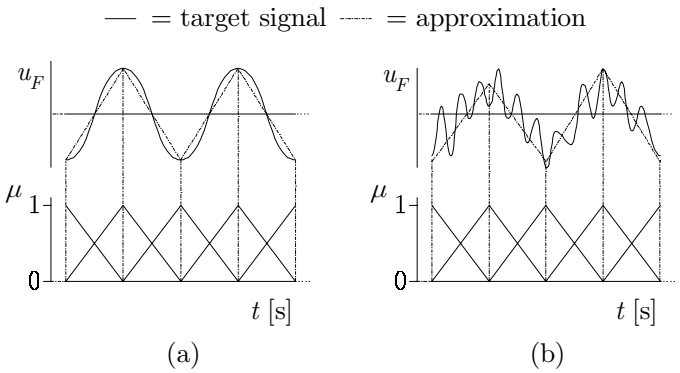



*Figure 2.15: a) Approximation of a low-frequency target signal*
*b) Approximation of a high-frequency target signal, filtering effect of a BSN*

Stabilising the time-indexed LFFC can be done by choosing the width of the B-splines such that the BSN is only able to accurately approximate signals in the low frequency range, where $L$ compensates correctly, and not signals in the high frequency range, where $L$ would cause unstable behaviour. In the next section, the relation between the width of the B-splines, the system dynamics and the stability will be studied thoroughly. Furthermore the influence of the learning rate on the stability of the system will be analysed [Schaak, 1996; Velthuis *et al.*, 2000].

## 2.5 Stability Analysis of Time-Indexed LFFC

### 2.5.1 Assumptions

In order to be able to analyse the stability of the time-indexed LFFC we assume the following:

1. The plant under control is SISO LTI.
2. The feedback controller, $C$, is linear, time-invariant and chosen such that the feedback loop is stable.
3. The discrete learning rule,

$$\Delta w_i = \gamma \frac{\sum_{k=0}^{T_p/h} \mu_i(kh)\, u_C(kh)}{\sum_{k=0}^{T_p/h} \mu_i(kh)} \tag{2.30}$$

(where $h$ is the sample time) is replaced by a continuous one

$$\Delta w_i = \gamma_c \frac{\int_0^{T_p} \mu_i(t) u_C(t)\, dt}{\int_0^{T_p} \mu_i(t) dt} \tag{2.31}$$

4. The B-spline distribution is assumed to be uniform (figure 2.13).

Assume $N$ B-splines, uniformly distributed on the input domain, $[0, T_p]$ [s], as shown in figure 2.13. The width, $d$ [s], of the support of basis functions 2 to $N$–1, is given by the following relation:

$$d = \frac{2T_p}{N-1} [\text{s}] \tag{2.32}$$

The membership of the $i$-th B-spline is now defined as:

$$\mu_i(t) = \begin{cases} \frac{2t - d(i-2)}{d} & \text{for} \quad \frac{d}{2}(i-2) \le t \le \frac{d}{2}(i-1) \\ \frac{di - 2t}{d} & \text{for} \quad \frac{d}{2}(i-1) \le t \le \frac{d}{2} i \\ 0 & \text{elsewhere} \end{cases} \tag{2.33}$$

Substituting the membership (2.33) in the learning rule (2.31) gives the following adaptation of the weights:

$$\Delta w_i = \gamma_c \frac{\int_{\frac{d}{2}(i-2)}^{\frac{d}{2}(i-1)} \frac{2t - d\,(i-2)}{d} u_C\,(t)\,dt + \int_{\frac{d}{2}(i-1)}^{\frac{d}{2}i} \frac{di - 2t}{d} u_C\,(t)\,dt}{\int_{\frac{d}{2}(i-2)}^{\frac{d}{2}(i-1)} \frac{2t - d\,(i-2)}{d}\,dt + \int_{\frac{d}{2}(i-1)}^{\frac{d}{2}i} \frac{di - 2t}{d}\,dt} \tag{2.34}$$

For the denominator of (2.34) holds:

$$\int_{\frac{d}{2}(i-2)}^{\frac{d}{2}(i-1)} \frac{2t - d\,(i-2)}{d}\,dt + \int_{\frac{d}{2}(i-1)}^{\frac{d}{2}i} \frac{di - 2t}{d}\,dt = \frac{d}{2} \tag{2.35}$$

Using (2.35), the adaptation of the weights (2.34) can be simplified to:

$$\Delta w_i = \gamma_c \int_{\frac{d}{2}(i-2)}^{\frac{d}{2}(i-1)} \frac{4t - d\,(2i-4)}{d^2} u_C\,(t)\,dt + \gamma_c \int_{\frac{d}{2}(i-1)}^{\frac{d}{2}i} \frac{2di - 4t}{d^2} u_C\,(t)\,dt \tag{2.36}$$

This implies that learning is linear in $u_C(t)$, and hence the feed-forward adaptation loop is linear. Since the feedback loop is also linear, the reference path may be taken equal to zero in the stability analysis (see figure 2.16). The desired situation is then $u_F$=0.

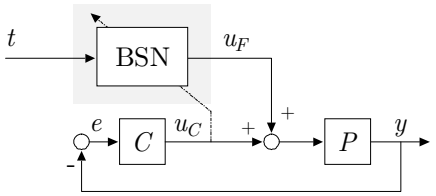



*Figure 2.16: Time-indexed LFFC when r=0.*

This system is stable if an arbitrarily chosen initial feed-forward signal will not result in an unbounded output of the plant. The (initial) feed-forward signal is determined by the (initial) values of the weights in the B-spline network. As the

feedback controlled system is stable, the output can only become unbounded when the feed-forward signal $u_F(t)$ becomes unbounded, which implies that at least one weight has become infinitely large. So, if the weights are adapted in such way that their value remains bounded, the system is stable; otherwise the system is unstable. The values of the weights remain bounded if:

1. Each weight is adapted in the right direction (towards $u_F(t)=0$), which means that:

$$\begin{aligned} \Delta w_i &\geq 0 \quad \text{for } w_i \leq 0 \\ \Delta w_i &\leq 0 \quad \text{for } w_i > 0 \end{aligned} \tag{2.37}$$

2. The weights are not adapted too strongly:

$$\begin{aligned} \Delta w_i &\leq -2w_i \quad \text{for } w_i \leq 0 \\ \Delta w_i &\geq -2w_i \quad \text{for } w_i > 0 \end{aligned} \tag{2.38}$$

Combining (2.37) and (2.38) yields:

$$\begin{aligned} 0 \leq \Delta w_i &\leq -2w_i \quad \text{for } w_i \leq 0 \\ -2w_i \leq \Delta w_i &\leq 0 \quad \text{for } w_i > 0 \end{aligned} \tag{2.39}$$

Note that (2.39) is a sufficient condition and not a necessary one. The problem is to select width $d$ and learning rate $\gamma_c$ in accordance with (2.39). In order to solve this problem, we assume that the shape of the initial feed-forward signal, $u_F(t)$ is triangular. This choice is motivated by the fact that experiments showed that when unstable behaviour occurs, the output of the BSN has a triangular shape [Velthuis *et al.*, 1996]. This input-output mapping can be realised by choosing the weights as $w_i=g$ for $i$=1,3,5… and $w_i=-g$ for $i$=2,4,6… where $g \in \mathbb{R}^+$, see figure 2.17.

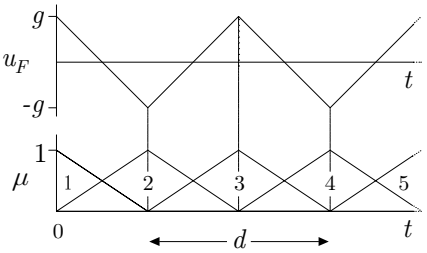



*Figure 2.17: Initial feed-forward signal*

The signal $u_F(t)$ can be written as a Fourier-series:

$$u_F(t) = \frac{8g}{\pi^2} \sum_{n=1,3,5...}^{\infty} \frac{\cos(\omega_n t)}{n^2} \tag{2.40}$$

with,

$$\omega_n = \frac{2\pi n}{d} [\mathrm{rad\,s^{-1}}] \tag{2.41}$$

In the frequency domain the relation between the output of the feed-forward controller $U_F$ and the learning signal $U_C$ is given by,

$$U_C = -T\, U_F \tag{2.42}$$

in which $T$ is the complementary sensitivity function:

$$T = \frac{CP}{1+CP} \tag{2.43}$$

$-T$ amplifies the magnitude of each frequency component of (2.40) by a factor $a_n = \left|-T(j\omega_n)\right|$ and introduces a phase shift $\varphi_n = \arg\left(-T(j\omega_n)\right)$, so that $u_C(t)$ can be written as:

$$u_C(t) = \frac{8g}{\pi^2} \sum_{n=1,3,5...}^{\infty} \frac{a_n \cos(\omega_n t + \varphi_n)}{n^2} \tag{2.44}$$

Substitution of (2.44) in (2.34) and reformulation gives:

$$\Delta w_i = \begin{cases} \dfrac{-32\gamma_c g}{\pi^4} \displaystyle\sum_{n=1,3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \text{for } i = 2,4,6... \\ \dfrac{32\gamma_c g}{\pi^4} \displaystyle\sum_{n=1,3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \text{ for } i = 1,3,5... \end{cases} \tag{2.45}$$

It can be seen that all weights that have the same initial value ($w_i$=$g$ for $i$=1,3,5… and $w_i$=$-g$ for $i$=2,4,6…) are equally adapted. Therefore learning does not change the shape of the feed-forward signal. Hence, for each iteration the feed-forward

signal can again be expressed as (2.40) and the weight adaptation as (2.45). In the following, the adaptation of weights that have a positive initial value, $w_i$=$a$, will be considered; for the other case, an analogous analysis can be made. Substituting (2.45) in the stability conditions (2.39) results in:

$$-2g \leq \frac{32\gamma_c g}{\pi^4} \sum_{n=1,3,5\ldots}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \leq 0 \tag{2.46}$$

$$\Leftrightarrow \frac{-\pi^4}{16\gamma_c} \leq \sum_{n=1,3,5\ldots}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \leq 0 \tag{2.47}$$

The left hand side inequality of (2.47):

$$\frac{-\pi^4}{16\gamma_c} \leq \sum_{n=1,3,5\ldots}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \tag{2.48}$$

contains $\gamma_c$, $a_n$ and $\varphi_n$. The values of $a_n$ and $\varphi_n$ depend on the value of $\omega_n$, which is determined by the choice of the B-spline width, $d$, see (2.41). Whether or not (2.48) is satisfied, depends thus on the choice of the learning rate and the B-spline width, $d$. The right hand side inequality of (2.47):

$$\sum_{n=1,3,5\ldots}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \leq 0 \tag{2.49}$$

only contains $\varphi_n$ and $a_n$. This means that only the choice of $d$ determines whether (2.49) is satisfied. Therefore, we propose to firstly choose $d$ such that (2.49) is satisfied. Next, using the obtained $d$ (and thus $\omega_n$) we can calculate $\gamma_c$ from (2.48). In the following we will try to find the smallest value of $d$ for which (2.49) is still satisfied.

### 2.5.2 B-Spline Width

In case an exact model of the plant, $P$, and the controller, $C$, is available, the values of $a_n$ and $\varphi_n$ can be calculated for all frequencies. This would allow the selection of the minimal $d$ for which (2.49) is satisfied by means of the following simple iterative search:

**Algorithm 2.1 (Calculation of the smallest stable value of *d*, based on a detailed model of the controlled system)**

1. Choose a uniform B-spline distribution that consists of 3 B-splines; $N$=3 in figure 2.13. Because according to (2.32) in this case $d$=$T_p$ [s], this is the minimum number of B-splines that can be chosen.
2. Determine $a_n$ and $\varphi_n$. This is done in the following way:

2.1 Choose $n$=1.

2.2 Calculate $\omega_n$ according to (2.41). Use the model of the plant and $\omega_n$ to determine $a_n$ and $\varphi_n$.

2.3 If $a_n \approx 0$ go to step 3, otherwise, increase $n$ by 1 and go to step 2.2.

3. Check if the $a_n$ and $\varphi_n$ obtained in the previous steps satisfy (2.49). If so, go to step 4, else go to step 6.
4. Increase the number of B-splines in the distribution by 1, $N$:=$N$+1.
5. Go to step 2.
6. Apparently $N$ is the smallest number of B-splines that results in unstable behaviour. Therefore, the maximum number of B-splines is $N-1$, which according to (2.32) gives:

$$d = \frac{2T_p}{N-2} \tag{2.50}$$

However, in general only the low-frequency dynamics of the plant are known and the model is inaccurate at high frequencies. Therefore, algorithm 2.1 may be unreliable. To resolve this, we take an approach that seems somewhat conservative. Firstly, we start by writing (2.49) as:

$$a_1 \cos\left(\varphi_1\right) + \sum_{n=3,5\ldots}^{\infty} a_n \frac{\cos\left(\varphi_n\right)}{n^4} \leq 0 \tag{2.51}$$

Next, we determine the smallest value of $d$ (which means the largest value of $\omega_1$) that satisfies (2.51), under the assumption that $a_n$ and $\varphi_n$ all have their worst case values. To gain insight in the worst case values, we make use of a part of a typical phase diagram of $-T$ (figure 2.18).

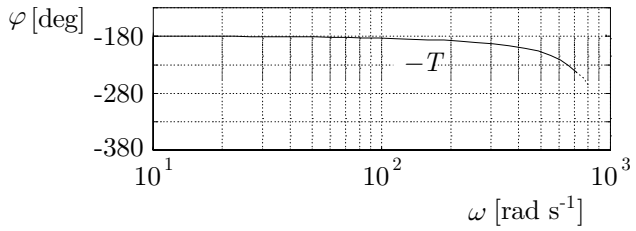



*Figure 2.18: Phase diagram of a typical $-T$*

In which $\varphi$ is the phase of $-T$. The corresponding diagram of $\cos(\varphi)$ is shown in figure 2.19.

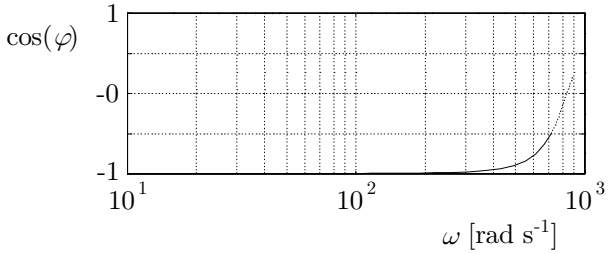



*Figure 2.19: Typical plot of $\cos(\varphi)$*

When we choose a large $d$, $\omega_1 \approx 0$, which results in $\varphi_1 = -180$ [deg] and thus $a_1 \cos\left(\varphi_1\right) \approx -a_1$. When we increase $\omega_1$ (decrease $d$), eventually we reach a value at which $\varphi_1 > -90$ [deg] or $\varphi_1 < -270$ [deg] such that $a_1 \cos\left(\varphi_1\right) > 0$, since $a_1 > 0$. Whether or not (2.51) is violated at this point depends on the value of $\sum_{n=3,5\ldots}^{\infty} a_n \frac{\cos\left(\varphi_n\right)}{n^4}$. When $\sum_{n=3,5\ldots}^{\infty} a_n \frac{\cos\left(\varphi_n\right)}{n^4} \leq 0$ we can further increase the value

of $\omega_1$ without violating (2.51). On the other hand, when $\sum_{n=3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \geq 0$ we must decrease the value of $\omega_1$. So the worst situation (from a stability point of view) is obtained when:

$$\sum_{n=3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} = \max\left( \sum_{n=3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \right) \tag{2.52}$$

Of which the upper bound is given by:

$$\begin{aligned} \max\left( \sum_{n=3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \right) &\leq \sum_{n=3,5...}^{\infty} \max(a_n) \frac{\max\left(\cos(\varphi_n)\right)}{n^4} = \\ &= \sum_{n=3,5...}^{\infty} \frac{\max(a_n)}{n^4} = \sum_{n=3,5...}^{\infty} \frac{\max\left|-T(j\omega_n)\right|}{n^4} \\ &\leq \sum_{n=3,5...}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{n^4} \end{aligned} \tag{2.53}$$

Thus (2.51) is certainly satisfied if:

$$a_1 \cos\left(\varphi_1\right) + \sum_{n=3,5...}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{n^4} \leq 0 \tag{2.54}$$

The largest value of $\omega_1$ for which (2.54) is satisfied can now be obtained by means of an iterative search in which a model of the low-frequency dynamics of the system is used:

**Algorithm 2.2 (Calculation of the smallest stable value *d*, mild assumptions on the dynamics of the controlled system)**

1. Use a low-frequency model of the system to calculate $\sum_{n=3,5...}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{n^4}$

2. Choose a uniform B-spline distribution consisting of 3 B-splines; $N$=3 in figure 2.13.
3. Calculate $\omega_1$ according to (2.41) for $n$=1.
4. Use the model to determine $a_1$ and $\varphi_1$.
5. Check if $a_1$ and $\varphi_1$ satisfy (2.54), using the result of step 1. If so, go to step 6, else go to step 8.

6 Create a uniform B-spline distribution of $N$+1 B-splines (in other words $N$:=$N$+1).

7. Go to step 3.
8. The minimum width $d$ is given by $d = \dfrac{2T_p}{N-2}$.

To overcome the need of an iterative search procedure, we can make some further assumptions about $a_1$. Firstly we represent (2.54) as:

$$\cos\left(\varphi_1\right) + \sum_{n=3,5\ldots}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{a_1 n^4} \le 0 \tag{2.55}$$

As before, the worst situation is obtained when the second term of the left-hand side of (2.55) reaches its maximum:

$$\sum_{n=3,5\ldots}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{a_1 n^4} = \max\left( \sum_{n=3,5\ldots}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{a_1 n^4} \right) = \sum_{n=3,5\ldots}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{\min\left(a_1\right) n^4} \tag{2.56}$$

Using (2.56) we can express (2.55) as:

$$\cos\left(\varphi_1\right) + \sum_{n=3,5\ldots}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{\min\left(a_1\right) n^4} \le 0 \tag{2.57}$$

We now have to determine the value of $\min\left(a_1\right)$. This is done by evaluating $\left|-T(j\omega_1)\right|$ for all possible values of $\omega_1$ that satisfy (2.55). The upper bound of the value of $\omega_1$ can be determined as follows. Using the fact that:

$$\sum_{n=3,5...}^{\infty} \frac{\left|-T(j\omega)\right|_{\infty}}{a_1 n^4} \geq 0 \tag{2.58}$$

$\varphi_1$ should satisfy (using (2.55)):

$$\cos\left(\varphi_1\right) \leq 0 \tag{2.59}$$

The upper bound of the value of $\omega_1$ is thus the frequency, $\omega$, at which $\cos(\varphi) \leq 0$. This results in:

$$\min\left(a_1\right) \geq \min_{\{\omega \in \mathbb{R} | \cos(\varphi) \leq 0\}} \left|-T(j\omega)\right| \tag{2.60}$$

In figure 2.20 an example of a typical Bode plot of $-T$ is presented, in which all $\omega$ for which $\cos(\varphi) \leq 0$ are shaded.

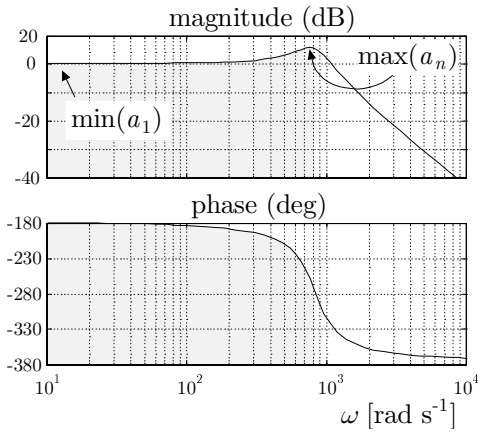



*Figure 2.20: Example of a Bode plot of $-T$*

Substituting (2.60) in (2.57) yields,

$$\begin{aligned}\cos\left(\varphi_1\right) &\leq -\sum_{n=3,5\ldots}^{\infty} \frac{\left|-T\left(j\omega\right)\right|_\infty}{\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right| n^4} \\ &\approx -0.0147\,\frac{\left|-T\left(j\omega\right)\right|_\infty}{\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right|}\end{aligned} \tag{2.61}$$

The above can be used to formulate an algorithm according to which the minimum value of $d$ can be obtained:

**Algorithm 2.3 (Calculation of the smallest stable value of *d*, strict assumptions about the dynamics of the controlled system)**

1. Determine $\left|-T\left(j\omega\right)\right|_\infty$ from a model of the closed loop system.
2. Use the Bode plot of the model to determine $\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right|$.
3. Search for the smallest value of $\omega_1$ at which $\varphi_1 = \arg\left(-T(j\omega_1)\right)$ satisfies

$$\varphi_1 = \arccos\left(-0.0147\,\frac{\left|-T\left(j\omega\right)\right|_\infty}{\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right|}\right) \tag{2.62}$$

4. The minimum value of the width of the B-splines, $d_{min}$, is given by $d_{min} = \dfrac{2\pi}{\omega_1}[\mathrm{rad\,s^{-1}}]$.

### 2.5.3 Learning Rate

To determine the maximum learning rate, we will use (2.38) for $w_i > 0$:

$$\Delta w_i \geq -2w_i \tag{2.63}$$

As a candidate feed-forward signal we choose the triangular shaped feed-forward signal of (2.40), to which a constant, $c$, is added:

$$u_F(t) = c + \frac{8g}{\pi^2} \sum_{n=1,3,5...}^{\infty} \frac{\cos(\omega_n t)}{n^2} \tag{2.64}$$

In the following we will show that the constant term causes a larger relative adaptation than the rest of the terms. The feedback signal caused by (2.64) is given by:

$$u_C(t) = a_0 c \cos(\varphi_0) + \frac{8g}{\pi^2} \sum_{n=1,3,5...}^{\infty} \frac{a_n \cos(\omega_n t + \varphi_n)}{n^2} \tag{2.65}$$

where $\varphi_0$ is the phase of $-T$ at $\omega = 0\left[\text{rad}\,\text{s}^{-1}\right]$, which is typically equal to –180 [deg].

$$u_C(t) = -a_0 c + \frac{8g}{\pi^2} \sum_{n=1,3,5...}^{\infty} \frac{a_n \cos(\omega_n t + \varphi_n)}{n^2} \tag{2.66}$$

The adaptation of $\Delta w_i$ can be split up as:

$$\Delta w_i = \Delta w_{i,c} + \Delta w_{i,rest} \tag{2.67}$$

where, $\Delta w_{i,c}$ and $\Delta w_{i,rest}$ are the part of the adaptation of the weight caused by respectively, the constant term and the triangular signal. Using (2.36) we obtain:

$$\begin{aligned} \Delta w_{i,c} &= \gamma_c \int_{\frac{d}{2}(i-2)}^{\frac{d}{2}(i-1)} \frac{4t - d(2i-4)}{d^2}\left(-a_0 c\right)dt + \gamma_c \int_{\frac{d}{2}(i-1)}^{\frac{d}{2}i} \frac{2di - 4t}{d^2}\left(-a_0 c\right)dt \\ &= -\gamma_c a_0 c \end{aligned} \tag{2.68}$$

and according to (2.45):

$$\Delta w_{i,rest} = \frac{32\gamma_c g}{\pi^4} \sum_{n=1,3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \tag{2.69}$$

When $\varphi_n = -180\,[\text{deg}]$, the largest possible negative weight adaptation is obtained:

$$\begin{aligned}\Delta w_{i,rest} = & \frac{32\gamma_c a}{\pi^4}\sum_{n=1,3,5...}^{\infty} a_n \frac{\cos(\varphi_n)}{n^4} \geq \\ & \frac{32\gamma_c a}{\pi^4}\sum_{n=1,3,5...}^{\infty} a_n \frac{-1}{n^4} = \\ & -\frac{32\gamma_c a}{\pi^4}\sum_{n=1,3,5...}^{\infty} \frac{a_n}{n^4}\end{aligned} \tag{2.70}$$

When we assume that $a_0 = a_n = \left|-T(j\omega)\right|_\infty$, the maximum adaptations are:

$$\Delta w_{i,c} = -\gamma_c \left|-T(j\omega)\right|_\infty c \tag{2.71}$$

$$\Delta w_{i,rest} = -\frac{32\gamma_c g}{\pi^4}\sum_{n=1,3,5...}^{\infty} \frac{\left|-T(j\omega)\right|_\infty}{n^4} \approx -0.0147\gamma_c \left|-T(j\omega)\right|_\infty g \tag{2.72}$$

Note that in practice $a_0$ will be close to $\left|-T(j\omega)\right|_\infty$, whereas $a_n$ will be much smaller for $n$ not close to 1. It can be seen that $\Delta w_{i,c} > \Delta w_{i,rest}$. Therefore, to determine the maximum learning rate, the initial feed-forward signal is assumed to be a constant signal, $u_F(t) = c$. Using (2.71) condition (2.63) can be written as:

$$-\gamma_c c\left|-T(j\omega)\right|_\infty \geq -2c \tag{2.73}$$

Which gives:

$$\gamma_c \leq \frac{2}{\left|-T(j\omega)\right|_\infty} \tag{2.74}$$

Herewith, a maximum value of the learning rate for a continuous learning mechanism has been obtained. However, when implemented, the time-indexed LFFC utilises the discrete learning mechanism (2.30). To calculate the maximum value of $\gamma$, a discrete approximation of the continuous learning mechanism, (2.31), is made. Assume that the sample time is small compared to the width of the B-

splines, $h << d$, then $\mu_i(t)$ and $u_C(t)$ are almost constant over one sample interval. The adaptation of the weights can then be written as:

$$\begin{aligned}\Delta w_i = \gamma_c \frac{\int_0^{T_p} \mu_i(t) u_C(t)\, dt}{\int_0^{T_p} \mu_i(t) dt} &\approx \gamma_c \frac{h \sum_{k=0}^{T_p/h} \mu_i(kh)\, u_C(kh)}{h \sum_{k=0}^{T_p/h} \mu_i(kh)} \\ &= \gamma_c \frac{\sum_{k=0}^{T_p/h} \mu_i(kh)\, u_C(kh)}{\sum_{k=0}^{T_p/h} \mu_i(kh)}\end{aligned} \tag{2.75}$$

The adaptation of the weights by the discrete approximation of the continuous learning rule, (2.75), is equal to the adaptation by the discrete learning rule, (2.30), when:

$$\gamma = \gamma_c \leq \frac{2}{\left| -T(j\omega) \right|_\infty} \tag{2.76}$$

With (2.61) and (2.76) guidelines on how to choose the width of the B-splines and the learning rate, such that the learning mechanism remains stable, based on a Bode plot of the negative complementary sensitivity function. In the following section these guidelines will be validated by means of simulations.

## 2.6 Simulations

The simulations presented in this section have two purposes. Firstly, they are used to check whether the minimal B-spline width and the maximum learning rate, as obtained by the stability analysis, will result in a stable system. Secondly, it will be examined how conservative these values are. In the stability analysis a number of worst case assumption were made that might result in conservative values of the minimum B-spline width and the maximum learning rate.

### 2.6.1 Mass Spring Mass Plant

The first system that is simulated is a mass-spring-mass (MSM) plant (figure 2.21). This plant has been chosen because it represents the dominant dynamic behaviour of many industrial motion systems [Groenhuis, 1991; Coelingh, 2000]. Here $m_1$ (=4.94 [kg]) represents mass of the motor, which drives a load mass $m_2$ (=5.39 [kg]) through a flexible transmission $c$ (1.37 $10^7$ [Nm$^{-1}$]).

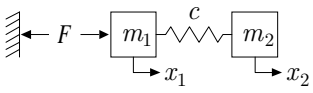



*Figure 2.21: Model of a industrial motion system*

In many situations it is difficult and/or expensive to measure the position of the load mass ($x_2$). Instead, the position of the motor ($x_1$) is used for control. In this research a PD-type feedback controller is considered of which the parameters were selected according to the rules given in [Coelingh, 2000]:

$$C = 1.68 \cdot 10^6 \left( \frac{1 \cdot 10^{-3}\, j\omega + 1}{1 \cdot 10^{-4}\, j\omega + 1} \right) \tag{2.77}$$

We use algorithm 2.3 to determine the value of $d_{min}$. Since the controlled system is LTI, assumption 1 is satisfied. The Bode plot of $-T$ is presented in figure 2.22.

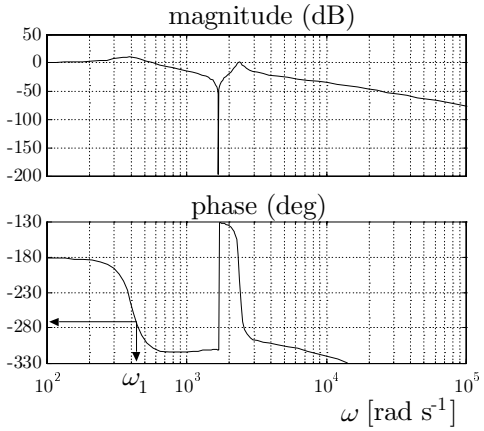



*Figure 2.22: Bode plot of $-T(j\omega)$ for the MSM plant*

From figure 2.22 it can seen that:

$$\min_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}} \left|-T(j\omega)\right| = 1\,\Big(0\text{ dB}\Big) \qquad \text{for } \omega = 0\left[\text{rad}\,\text{s}^{-1}\right] \tag{2.78}$$

$$\left|-T(j\omega)\right|_{\infty} = 3.024\,\Big(9.54\text{ dB}\Big) \tag{2.79}$$

This means that (2.61) results in:

$$\cos\Big(\varphi_1\Big) \leq -0.0445 \;\Rightarrow\; 92.55\,[\text{deg}] \leq \varphi_1 \leq 267.5\,[\text{deg}] \tag{2.80}$$

The smallest $\omega_1$ that violates (2.80) is $\omega_1$=428 [rad $s^{-1}$]. The minimum B-spline width is:

$$d_{min} = \frac{2\pi}{428} = 1.47\cdot 10^{-2}\,[\text{sec}] \tag{2.81}$$

The maximum learning rate can be calculated according to (2.76):

$$\gamma \leq \frac{2}{\left|-T\left(j\omega\right)\right|_{\infty}} = \frac{2}{3.024} = 0.661 \tag{2.82}$$

The motion that the system is to perform repeatedly is given in figure 2.23.

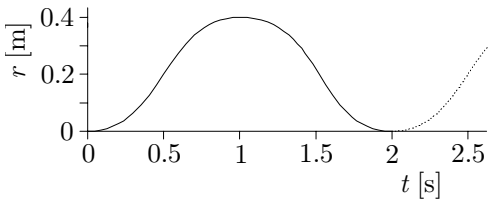



*Figure 2.23: Path for the MSM system*

**Simulation 2.1 (feedback controlled MSM)**

In the first simulation that is performed, the system is only controlled by the feedback controller. The tracking error that results is shown in figure 2.24.

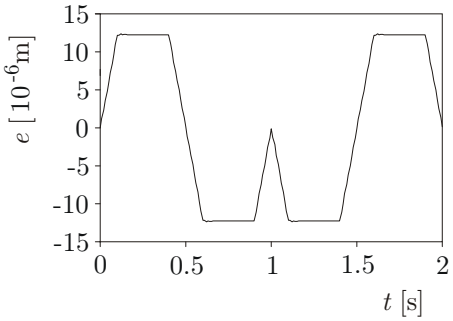



*Figure 2.24: Tracking error MSM, d conform the stability criterion*

**Simulation 2.2 (time-indexed LFFC controlled MSM, *d* conform (2.81))**

In this series of simulations the B-spline width is chosen slightly larger than $d_{min}$, i.e. $d$=1.51·10$^{-2}$ [s]. We choose $\gamma$=0.15, which should yield a stable system. The tracking error that results after, 10 [s], 40 [s] and 100.000 [s] is depicted in figure 2.25.

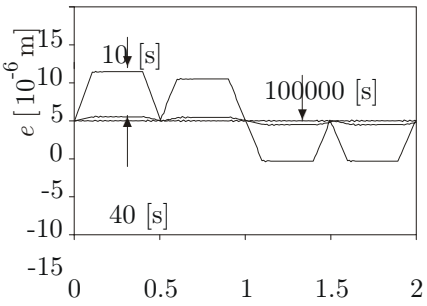



*Figure 2.25: Tracking error MSM, d conform the stability criterion*

It can be seen that after 40 [s], which corresponds to 20 motions, the LFFC was able to considerably improve upon the tracking performance of the feedback controller. When training is continued, the tracking error is decreased further. After 100000 [s] the tracking changed no more, from which we conclude that the LFFC is stable.

**Simulation 2.3 (time-indexed LFFC controlled MSM, *d* not conform (2.81))**

To test whether $d_{min}$ has a conservative value, a series of simulations is performed in which the B-spline width is chosen slightly smaller than $d_{min}$, $d$=1.46·10$^{-2}$ [s].

Again, we choose $\gamma$=0.15. Simulations show that after the tracking error has decreased in the first few runs, it increases thereafter. In figure 2.26 the tracking error after 600 seconds is presented. When the simulations are continued, the tracking error keeps on growing without bound, which means that the system has become unstable.

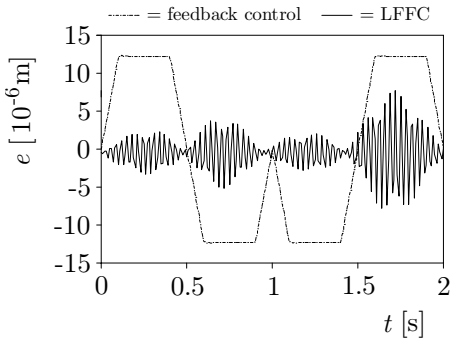



*Figure 2.26: Tracking error MSM, d just not conform stability criterion*

**Simulation 2.4 (time-indexed LFFC controlled MSM, *d* conform (2.81), maximum learning rate)**

Next, we examine whether the maximum value of the learning rate is correct and whether it is a conservative value. In the simulations, the width of the B-splines is chosen $d$=1.51·$10^{-2}$ [s]. In figure 2.27 the tracking error is shown when $\gamma$=0.661. The tracking clearly converges and learning remains stable.

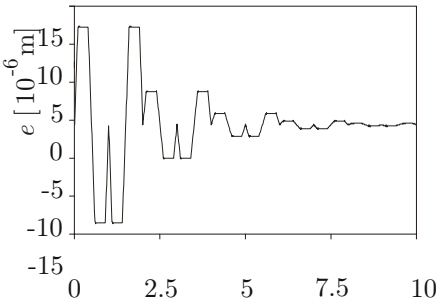



*Figure 2.27: Tracking error MSM, $\gamma$=0.661*

Next, a series of simulations is performed in which we start with a small learning rate and then increase it until the system becomes unstable. For $0<\gamma<0.98$ the feed-forward signal gradually converges to its final value. In figure 2.28 the feed-forward signal for the first 5 motions is shown ($\gamma$=0.661).

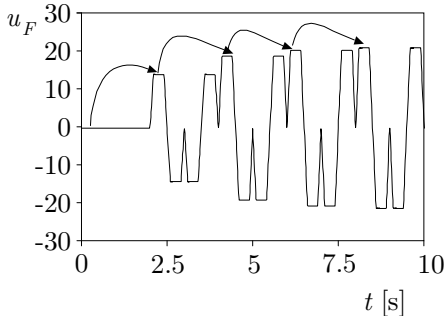



*Figure 2.28: Feed-forward signal, $\gamma$=0.661*

When $0.98<\gamma<1.96$ the feed-forward signal is adapted in an oscillatory way. In figure 2.29 the feed-forward signal is shown in a simulation in which $\gamma$=1.5. It can be seen that after the first motion, the amplitude of the feed-forward has increased, while after the second motion the amplitude has decreased. However, the amplitude of the oscillation decreases and the feed-forward signal still converges to its desired value.

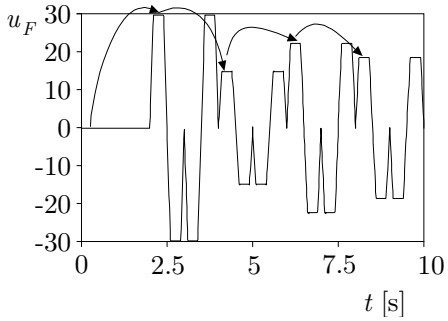



*Figure 2.29: Feed-forward signal, $\gamma$=1.5*

The simulations showed that the time-indexed LFFC became unstable when $\gamma>1.96$. This value is much larger than the value in (2.82). We may thus conclude that, for this system, the worst case assumptions in the stability analysis of the learning rate resulted in an conservative value.

### 2.6.2 LiMMS

The second system that is simulated is the Linear Motor Motion System (LiMMS) presented in chapter 1. The position of the translator is controlled by means of a

PD-type feedback controller. The parameters of the feedback controller were obtained by means of an auto-tuning mechanism, present in the LiMMS set-up.

$$C = 275280\left(\frac{0.02j\omega + 1}{0.002j\omega + 1}\right) \tag{2.83}$$

The Bode plot of $-T$ is shown in figure 2.30. It can be seen that:

$$\min_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}} \left|-T(j\omega)\right| = 0.7395\,(-2.62\text{ dB}) \quad \text{for } \omega = 220\left[\text{rad}\,\text{s}^{-1}\right] \tag{2.84}$$

$$\left|-T(j\omega)\right|_{\infty} = 1.2937\,\left(2.34\text{ dB}\right) \tag{2.85}$$

which gives:

$$\cos\left(\varphi_1\right) \leq -0.0257 \;\Rightarrow\; \text{-91.5 [deg]} \leq \varphi_1 \leq \text{-268 [deg]} \tag{2.86}$$

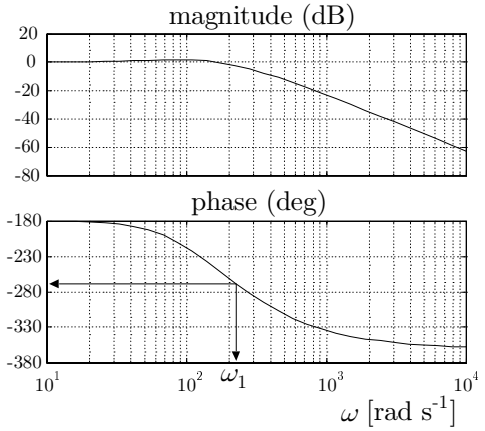



*Figure 2.30: Bode plot of the LiMMS*

Inspecting figure 2.30 shows that we should choose $\omega_1$<=220 [rad s$^{-1}$] and thus:

$$d_{min} = \frac{2\pi}{220} = 2.855 \cdot 10^{-2}[\text{s}] \tag{2.87}$$

The learning rate should now be chosen:

$$\gamma \leq \frac{2}{\left|-T\left(j\omega\right)\right|_{\infty}} = \frac{2}{1.2937} = 1.5459 \tag{2.88}$$

Now, comparable results will be pursued by simulations with a model of the LiMMS. The reference position that has to be tracked is given in figure 2.31.

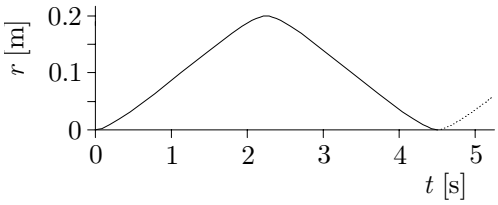



*Figure 2.31: Reference position for the LiMMS*

**Simulation 2.5 (feedback controlled LiMMS)**

The tracking error that is obtained, when the LiMMS is controlled by the feedback controller only, is given in figure 2.32. It can be seen that the feedback controller is not able to fully compensate for the cogging.

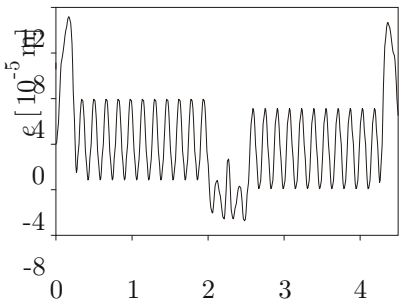



*Figure 2.32: Tracking error LiMMS, feedback control only*

**Simulation 2.6 (time-indexed LFFC controlled LiMMS, $d$ conform (2.87))**

In the first series of simulations with the time-indexed LFFC, the width of the B-splines is larger than its minimum value, $d$=2.91·10$^{-2}$ [s]. We choose $\gamma$=0.5. The tracking error after 22.5 [s] is shown in figure 2.33. It can be seen that the time-indexed LFFC has been able to decrease the tracking error drastically.

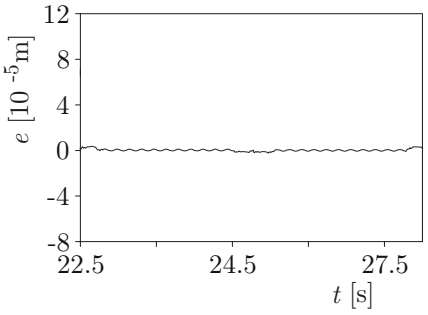



*Figure 2.33: Tracking error LiMMS, d conform the stability criterion*

When learning is continued the tracking error decreases further. Learning remains stable though. In figure 2.34 the tracking error after 100.000 seconds is shown.

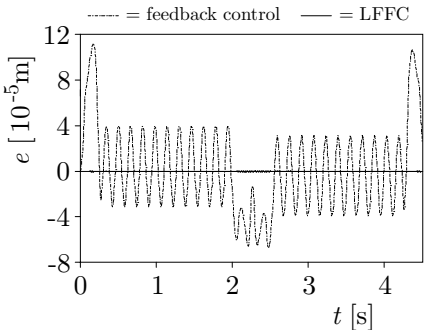



*Figure 2.34: Tracking error LiMMS, d conform the stability criterion*

The time-indexed LFFC is able to learn to compensate for the cogging force and obtains a considerably smaller tracking error than the feedback controller. Furthermore, when learning is continued the system remains stable.

**Simulation 2.7 (time-indexed LFFC controlled LiMMS, *d* not conform (2.87))**

In the following series of simulations the width of the B-splines is chosen $d$=2.78·10$^{-2}$ [s]. For this width, the system becomes unstable as can be seen in figure 2.35, where the tracking error after 2000 seconds is depicted. When learning is continued, this error keeps on growing.

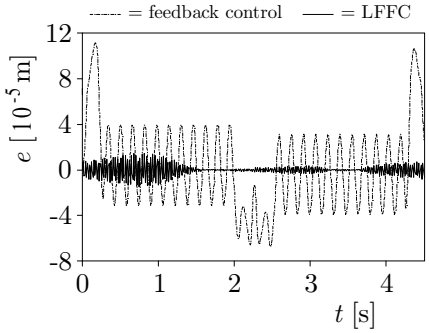



*Figure 2.35: Tracking error LiMMS, d just not conform the stability criterion*

We may conclude that even though the LiMMS is a non-linear plant, the value of $d_{min}$ is still valid and not conservative.

**Simulation 2.8 (time-indexed LFFC controlled LiMMS, *d* conform (2.87), maximum learning rate)**

Next, the maximum value of the learning rate is examined for which the system is stable. In these simulations we take $d$=2.91·10$^{-2}$ [s]. First, we choose $\gamma$=1.5459. Figure 2.36 shows that in this case the weights are adapted in an oscillatory way.

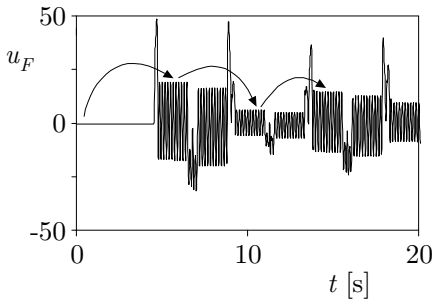



*Figure 2.36: Tracking error LiMMS, d just not conform the stability criterion*

Simulations show that for $0<\gamma<1$ the feed-forward signal converges gradually, for $1<\gamma<1.99$ in an oscillatory way and for $\gamma>1.99$ the feed-forward signal diverges. So, the maximum of the learning rate as determined in (2.88) gives a stable system and is not as conservative a value as for the mass-spring-mass system.

## 2.7 Discussion

The analysis of section 2.5 revealed that a stable time-indexed LFFC based on $2^{\text{nd}}$ order B-spline can be designed using algorithm 2.3. During the derivation a number of rather strong assumptions has been made. In this section, we will comment on these assumptions. In order to determine the minimum B-spline width, we had to assume the following:

– *The weights are adapted in the right direction*:

$$\begin{aligned} \Delta w_i &\geq 0 \quad \text{for } w_i \leq 0 \\ \Delta w_i &\leq 0 \quad \text{for } w_i > 0 \end{aligned} \tag{2.89}$$

This condition is necessary in case of the triangular shaped initial feed-forward signal with zero mean (2.40). In that case all weights are adapted equally strong; the positive weights in one direction and the negative weights in the other. By learning, only the amplitude of the feed-forward signal changes, not the shape or the mean. If in one run the negative weights are adapted in the negative direction (and the positive weights in the positive direction), this will also occur in the next run and the feed-forward signal will thus grow without bound.

When choosing a different initial feed-forward signal, (2.89) becomes a sufficient condition, not a necessary one. This can be easily seen when we add an offset to the triangular initial feed-forward signal, such as in (2.64). When the offset, $c$, equals $g-\delta$ ($\delta$ is chosen very small), the following initial feed-forward signal is obtained.

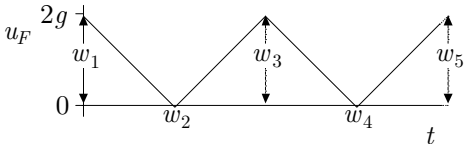



*Figure 2.37: Triangular feed-forward signal with an offset*

The weights are chosen as follows: $w_{1,3,\ldots}=2g-\delta$ and $w_{2,4,\ldots}=-\delta$,. Stability condition (2.89) says that $w_{2,4,\ldots}$ must be adapted in the positive direction. According to section 2.5, the weight adaptation consists of 2 parts, one caused by the constant term and one caused by the triangular signal. Furthermore, it was stated that the largest part of the weight adaptation is caused by the constant term. During learning, the amplitude of the constant term will thus decrease faster than the amplitude of the triangular signal.

This means that the weights $w_{2,4,\ldots}$ are first adapted in the negative direction and once the constant term is small enough, in the positive direction. So during the first phase, condition (2.89) is not satisfied, and yet the system is guaranteed to be stable.

– *The initial feed-forward signal was assumed to be triangular with zero mean.* Choosing this feed-forward signal was motivated by observing that in experiments in which the time-indexed LFFC became unstable, the feed-forward signal had a triangular shape. Furthermore, this initial feed-forward signal could be written as a simple Fourier series, of which the period of the lowest frequency component equals the B-spline width. This enabled us to make some worst-case assumptions on the high-frequency terms and then calculate what phase shift the lowest frequency component may have such that the weights still converge. When, instead, a random initial feed forward signal is chosen, the following Fourier series results:

$$u_F(t) = c_0 + \sum_{k=1}^{\infty} c_n \cos\left(\frac{2\pi k t}{T_p}\right) + \sum_{l=1}^{\infty} s_n \sin\left(\frac{2\pi l t}{T_p}\right) \tag{2.90}$$

In (2.90), the B-spline width does not correspond to the period of the lowest frequency term ($k$=1, $l$=1), but to a higher frequency component (the width of the B-spines determines which one). This means that, in order to calculate the minimum B-spline width, we have to make worst-case assumptions on both the low-frequency terms as well as the high-frequency terms. This makes the analysis so complex, that we suggest that, to examine stability for random initial feed-forward signals, another approach must be sought.

Although the choice of the initial feed-forward signal seems rather restrictive, the resulting minimum B-spline width is accurate. Simulations were performed with different sets of initial weights, $w_i \neq 0$. In spite of the fact that the initial weights do not satisfy the assumption, the time-indexed LFFC remained stable when the B-spline width was chosen slightly larger than the minimum width and became unstable when the B-spline width was chosen slightly smaller. Additionally, we performed a large number of stability simulations of the LiMMS, where the initial weights of the BSN were chosen randomly. None of the simulations showed unstable behaviour.

– *Worst case assumptions concerning the system dynamics.* In order to be able to calculate the maximum stable value of $d$ for a certain system, in (2.51)-(2.61) a number of worst case assumptions on the system dynamics was made. Regarding the results of section 2.5 we conclude that for these systems, the assumptions did not result in a conservative value of $d_{min}$. This

is understandable considering that $a_1$, $a_n$, $\varphi_n$ ($n$=2,3,...) in (2.51) are amplified by a factor $\frac{1}{n^4}$. The assumptions are particularly conservative for larger $n$, because then the factor $n^{-4}$ is dominantly small.

– *The system is linear.* We have only been able to derive a condition for the width of the support of the B-splines, in case the controlled system is LTI. The cogging force in the LiMMS makes the plant (and thus the controlled system) non-linear. In spite of this, the value of $d_{min}$ that was determined on the basis of a Bode-plot of the frequency response of the controlled system, proved to be accurate. This does not necessarily mean that for any non-linear plant we can determine the value of $d_{min}$ on the basis of a linearisation of the plant. Further research on stability of time-indexed LFFC for non-linear plants is needed.

For the calculation of the maximum learning rate we assumed that:

– *The worst case initial feed-forward signal is a constant signal.* This assumption is based on calculating the maximum weight adaptation by a triangular feed-forward signal with an offset. Since the maximum weight adaptation for the constant signal was the largest, this signal was taken as the initial feed-forward signal. As before, allowing a random initial feed-forward signal would require a different approach.

– *The amplification of the initial feed-forward signal by the system is* $\left|-T(j\omega)\right|_\infty$. Looking at the results of the simulations we may conclude that this assumption may lead to conservative maximum values of the learning rate. However, the difference between the theoretical and the practical values is not so large, that it will lead to very long training periods.

In the standard time-indexed LFFC, the minimum B-spline width, and thus the maximum achievable accuracy, is determined by the system dynamics only. To further increase the accuracy, a learning filter, $L$, can be added to the time-indexed LFFC (see figure 2.38).

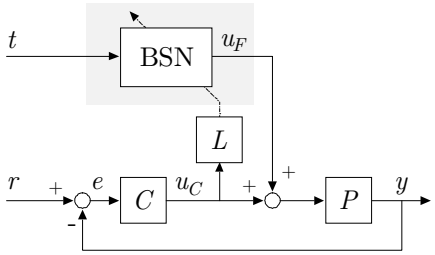



*Figure 2.38: Time-indexed LFFC with an additional learning filter*

The design of the $L$ filter can be performed according to the same procedure as the design of $L$ in ILC (section 2.2).

# 3 Non-Repetitive Motions: Input Selection and Convergence

## 3.1 Introduction

The time-indexed LFFC, the ILC and the RC store the feed-forward signal that compensates the plant dynamics and the reproducible disturbances, as a function of the periodic motion time. Consequently, these learning controllers can be used for repetitive motions only. Each time the reference motion changes, the learning controller must be retrained, which means that it is no longer able to track the old reference motion. This is necessary even when the old and the new reference motions are almost the same. In order to overcome this problem, and enable the learning controller to track several motions without retraining, it has been proposed to apply multiple feed-forward controllers instead of only one [Arif *et al.*, 1999]. Each of these feed-forward controllers is trained for one specific reference motion. LFFC offers a different solution. By choosing the reference position and derivatives thereof as inputs of the feed-forward part, instead of the periodic motion time, we are able to store the feed-forward signals for several motions in one feed-forward controller [Vries and Velthuis, 1998].

The inputs of the feed-forward part depend on the plant and the reproducible disturbances. In section 3.2 we will discuss in detail which inputs should be selected. Next, long term stability will be addressed in section 3.3. Based on the analysis of the time-indexed LFFC we can conclude that, without any additional robustness measures we cannot ensure stability. Therefore, a new robustness measure that we will refer to as *regularisation*, will be introduced in section 3.4.

## 3.2 Input Selection

Motion systems can often be modelled as a (non-)linear state space model and additive static non-linear functions, as shown in figure 3.1.

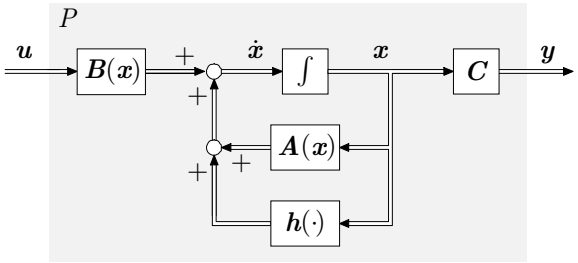



*Figure 3.1: General notation of a class of (non-linear) motion system*

This yields,

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{3.1}$$

The LFFC is intended to learn the steering signal, $\boldsymbol{u}_d$, that makes the output $\boldsymbol{y}$ equal to the desired output $\boldsymbol{y}_d$. The state trajectory for which $\boldsymbol{y}=\boldsymbol{y}_d$ is denoted as $\boldsymbol{x}_d$. We can reformulate (3.1) in the following way (if $\boldsymbol{B}^{-1}(\boldsymbol{x})$ exists):

$$\begin{aligned} \dot{\boldsymbol{x}} - \boldsymbol{A}(\boldsymbol{x})\boldsymbol{x} - \boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{B}^{-1}(\boldsymbol{x})\left(\dot{\boldsymbol{x}} - \boldsymbol{A}(\boldsymbol{x})\boldsymbol{x} - \boldsymbol{h}(\boldsymbol{x})\right) &= \boldsymbol{u} \end{aligned} \tag{3.2}$$

From (3.2) we can conclude that the desired feed-forward signal is given by:

$$\boldsymbol{u}_d = \boldsymbol{B}^{-1}\left(\boldsymbol{x}_d\right)\left(\dot{\boldsymbol{x}}_d - \boldsymbol{A}\left(\boldsymbol{x}_d\right)\boldsymbol{x}_d - \boldsymbol{h}\left(\boldsymbol{x}_d\right)\right) \tag{3.3}$$

The input-output relation of the BSN is a static mapping from $\boldsymbol{y}_d$ and if necessary derivatives / integrals of $\boldsymbol{y}_d$, $\left\{..., \int\int \boldsymbol{y}_d, \int \boldsymbol{y}_d, \boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \ddot{\boldsymbol{y}}_d, ...\right\}$ to $\boldsymbol{u}_F$. The integrals are needed when $\boldsymbol{y}_d$ does not represent the reference position, e.g. the reference velocity or acceleration. The question is now, which conditions

$\{\boldsymbol{A}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{B}(\boldsymbol{x}), \boldsymbol{C}\}$ must fulfil, so that (3.3) can be written as a static function of $\left\{\int\int \boldsymbol{y}_d, \int \boldsymbol{y}_d, \boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \ddot{\boldsymbol{y}}_d, \ldots\right\}$.

In (3.1) the state-vector, $\boldsymbol{x}$, can be chosen such that:

$$
\begin{aligned}
&\boldsymbol{x}=\begin{bmatrix}\boldsymbol{x}_1{}^T & \cdots & \boldsymbol{x}_n{}^T\end{bmatrix}^T, && \boldsymbol{x}_1,..,\boldsymbol{x}_n \in \mathbb{R}^m,\ m \in \mathbb{N}^+\\
&\boldsymbol{y} \in \mathbb{R}^m &&\\
&\boldsymbol{A}(\boldsymbol{x}) = \left[\begin{array}{c|cccc} \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{I} \\ \hline \boldsymbol{A}_1(\boldsymbol{x}) & \boldsymbol{A}_2(\boldsymbol{x}) & & \boldsymbol{A}_{n-1}(\boldsymbol{x}) & \boldsymbol{A}_n(\boldsymbol{x}) \end{array}\right], && \boldsymbol{A}_i(\boldsymbol{x}) \in \mathbb{R}^{m\times m} \qquad (3.4)\\
&\boldsymbol{B}(\boldsymbol{x})=\left[\boldsymbol{0} \quad \cdots \quad \boldsymbol{0} \mid \boldsymbol{B}_n(\boldsymbol{x})\right]^T, && \boldsymbol{B}_{\mathrm{n}}(\boldsymbol{x}) \in \mathbb{R}^{m\times m}\\
&\boldsymbol{C} = \begin{bmatrix}\boldsymbol{C}_1 & \cdots & \boldsymbol{C}_n\end{bmatrix} && \boldsymbol{C}_1,..,\boldsymbol{C}_n \in \mathbb{R}^{m\times m}
\end{aligned}
$$

The additive non-linearities, $\boldsymbol{h}(\cdot)$, can be written as a function the state:

$$\boldsymbol{h}(\cdot) = \left[\boldsymbol{0} \quad \cdots \quad \boldsymbol{0} \mid \boldsymbol{h}_n(\boldsymbol{x})\right]^T, \quad \boldsymbol{h}_n(\boldsymbol{x}) \in \mathbb{R}^{m\times m} \qquad (3.5)$$

For mechanical systems we typically find that $n$=2, where $\boldsymbol{x}_1$ consists of positions and $\boldsymbol{x}_2$ of the corresponding velocities.

**Theorem 3.1 (Plant conditions)** Consider a motion system conform (3.1). In order to control such a plant with LFFC, the following conditions must be satisfied:

1. $\boldsymbol{C}$ may contain only one element, $\boldsymbol{C}_j$, that is unequal to zero.
2. $\boldsymbol{C}_j$ must be invertible.
3. $\boldsymbol{B}_n(\boldsymbol{x})$ must be invertible.

**Proof 3.1**: First, we will derive that for the linear case, the above conditions are necessary. Next, it is shown that in the general case they are sufficient. From (3.4) it can be seen that $\boldsymbol{x}_{i+1} = \dot{\boldsymbol{x}}_i$. This gives us the opportunity to write $\boldsymbol{y}$ as a function of $\boldsymbol{x}_1$:

$$\begin{aligned} & \boldsymbol{y} = \begin{bmatrix} \boldsymbol{C}_1 & \boldsymbol{C}_2 & \cdots & \boldsymbol{C}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix}^T \\ \Leftrightarrow \quad & \boldsymbol{y} = \begin{bmatrix} \boldsymbol{C}_1 & \boldsymbol{C}_2 & \cdots & \boldsymbol{C}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_1^{(1)} & \cdots & \boldsymbol{x}_1^{(n-1)} \end{bmatrix}^T \\ \Leftrightarrow \quad & \boldsymbol{y} = \left( \boldsymbol{C}_1 + s\boldsymbol{C}_2 + \cdots + s^{n-1}\boldsymbol{C}_n \right) \boldsymbol{x}_1 \\ \Leftrightarrow \quad & \left( \boldsymbol{C}_1 + s\boldsymbol{C}_2 + \cdots + s^{n-1}\boldsymbol{C}_n \right)^{-1} \boldsymbol{y} = \boldsymbol{x}_1 \end{aligned} \tag{3.6}$$

where $\dfrac{d^n x}{dt^n}$ is denoted as $x^{(n)}$. The same is done for $\boldsymbol{x}_1$ and $\boldsymbol{u}$:

$$\begin{aligned} & \boldsymbol{x}_n^{(1)} = \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 + \cdots + \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{B}_n \boldsymbol{u} \\ \Leftrightarrow \quad & s^n \boldsymbol{x}_1 = \boldsymbol{A}_1 \boldsymbol{x}_1 + s\boldsymbol{A}_2 \boldsymbol{x}_1 + \cdots + s^{n-1} \boldsymbol{A}_n \boldsymbol{x}_1 + \boldsymbol{B}_n \boldsymbol{u} \\ \Leftrightarrow \quad & \left( s^n I - \boldsymbol{A}_1 - s\boldsymbol{A}_2 - \cdots - s^{n-1} \boldsymbol{A}_n \right) \boldsymbol{x}_1 = \boldsymbol{B}_n \boldsymbol{u} \end{aligned} \tag{3.7}$$

Substituting (3.6) in (3.7) yields:

$$\left( s^n I - \boldsymbol{A}_1 - s\boldsymbol{A}_2 - \cdots - s^{n-1} \boldsymbol{A}_n \right) \cdot \left( \boldsymbol{C}_1 + s\boldsymbol{C}_2 + \cdots + s^{n-1}\boldsymbol{C}_n \right)^{-1} \boldsymbol{y} = \boldsymbol{B}_n \boldsymbol{u} \tag{3.8}$$

The desired feed-forward signal, $\boldsymbol{u}_d$, which it to produce the desired output, $\boldsymbol{y}_d$, is given by:

$$\begin{aligned} \boldsymbol{u}_d = {} & \boldsymbol{B}_n{}^{-1} \left( s^n I - \boldsymbol{A}_1 - s\boldsymbol{A}_2 - \cdots - s^{n-1} \boldsymbol{A}_n \right) \cdot \\ & \left( \boldsymbol{C}_1 + s\boldsymbol{C}_2 + \cdots + s^{n-1}\boldsymbol{C}_n \right)^{-1} \boldsymbol{y}_d \end{aligned} \tag{3.9}$$

The LFFC is able to learn the desired feed-forward signal, when (3.9) can be written as a *static* mapping of the inputs of the BSN, $\left\{ \ldots, s^{-2}\boldsymbol{y}_d, s^{-1}\boldsymbol{y}_d, \boldsymbol{y}_d, s\boldsymbol{y}_d, s^2\boldsymbol{y}_d, \ldots \right\}$. This is the case when only one of the $\boldsymbol{C}_j$ is unequal to zero. The desired feed-forward signal that results is:

$$\boldsymbol{u}_d = \boldsymbol{B}_n{}^{-1}\left(s^{n-j+1}I - s^{-j+1}\boldsymbol{A}_1 - s^{-j+2}\boldsymbol{A}_2 - \cdots - s^{n-j}\boldsymbol{A}_n\right)\boldsymbol{C}_j^{-1}\boldsymbol{y}_d \qquad (3.10)$$

That $\boldsymbol{C}_j$ and $\boldsymbol{B}_n$ must be invertible, follows directly from (3.10). If several submatrices of $\boldsymbol{C}$ are unequal to zero, for example $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$, $\boldsymbol{u}_d$ contains the term $\left(\boldsymbol{C}_1 + s\boldsymbol{C}_2\right)^{-1}\boldsymbol{y}_d$ that cannot be written as a static function of $\left\{...,s^{-2}\boldsymbol{y}_d, s^{-1}\boldsymbol{y}_d, \boldsymbol{y}_d, s\boldsymbol{y}_d, s^2\boldsymbol{y}_d,...\right\}$.

To prove that in the non-linear case these conditions are (at least) sufficient, we will show that when they are satisfied, the LFFC is able to learn the desired feed-forward signal. Because $\boldsymbol{y} = \boldsymbol{C}_j\boldsymbol{x}_j$ and $\boldsymbol{A}$ is such that $\boldsymbol{x}_{i+1} = \dot{\boldsymbol{x}}_i$, we can transform the state space model:

$$\begin{aligned}
&\begin{bmatrix} \boldsymbol{x}_1^{(1)} \\ \vdots \\ \boldsymbol{x}_{n-1}^{(1)} \\ \boldsymbol{x}_n^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{A}_1(\boldsymbol{x}) & \boldsymbol{A}_2(\boldsymbol{x}) & \cdots & \boldsymbol{A}_n(\boldsymbol{x}) \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{n-1} \\ \boldsymbol{x}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{h}_n(\boldsymbol{x}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{B}_n(\boldsymbol{x}) \end{bmatrix}\boldsymbol{u} \\
&\boldsymbol{y} = \boldsymbol{C}_j\boldsymbol{x}_j
\end{aligned} \qquad (3.11)$$

into

$$\begin{aligned}
\boldsymbol{x}_j^{(n+1-j)} = {} & \boldsymbol{A}_1\left(...,\boldsymbol{x}_j^{(-1)},\boldsymbol{x}_j,\boldsymbol{x}_j^{(1)},...\right)\boldsymbol{x}_j^{(1-j)} + \cdots + \boldsymbol{A}_n\left(...,\boldsymbol{x}_j^{(-1)},\boldsymbol{x}_j,\boldsymbol{x}_j^{(1)},...\right)\boldsymbol{x}_j^{(n-j)} + \\
& + \boldsymbol{h}_n\left(...,\boldsymbol{x}_j^{(-1)},\boldsymbol{x}_j,\boldsymbol{x}_j^{(1)},...\right) + \boldsymbol{B}_n\left(...,\boldsymbol{x}_j^{(-1)},\boldsymbol{x}_j,\boldsymbol{x}_j^{(1)},...\right)\boldsymbol{u}
\end{aligned} \qquad (3.12)$$

where $\boldsymbol{x}_j^{(-1)} = \int \boldsymbol{x}_j$. Substituting $\boldsymbol{y} = \boldsymbol{C}_i\boldsymbol{x}_i$ in (3.12) gives

$$\begin{aligned}\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(n+1-j)} = &\boldsymbol{A}_1\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1-j)}+\cdots\\ &+\boldsymbol{A}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)\boldsymbol{y}^{(n-j)}+\cdots\\ &+\boldsymbol{h}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)+\cdots\\ &+\boldsymbol{B}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)\boldsymbol{u}\end{aligned} \tag{3.13}$$

The desired feed-forward signal is then given by:

$$\begin{aligned}&\boldsymbol{B}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)^{-1}\cdot\left(\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(n+1-j)}-\cdots\right.\\ &\quad-\boldsymbol{A}_1\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(1)},...\right)\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(1-j)}-\cdots\\ &\quad-\boldsymbol{A}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}^{(1)},...\right)\boldsymbol{y}_d^{(n-j)}-\cdots\\ &\quad\left.-\boldsymbol{h}_n\left(...,\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(-1)},\boldsymbol{C}_j^{-1}\boldsymbol{y},\boldsymbol{C}_j^{-1}\boldsymbol{y}_d^{(1)},...\right)\right)=\boldsymbol{u}_d\end{aligned} \tag{3.14}$$

Each term of $\boldsymbol{u}_d$ can be written as a static mapping of $\left\{\int\int\boldsymbol{y}_d,\int\boldsymbol{y}_d,\boldsymbol{y}_d,\dot{\boldsymbol{y}}_d,\ddot{\boldsymbol{y}}_d,...\right\}$, which means that the BSN is able to learn the desired feed-forward signal □

The inputs of the feed-forward part follow directly from the desired feed-forward signal, (3.14). Each $\left\{\int\int\boldsymbol{y}_d,\int\boldsymbol{y}_d,\boldsymbol{y}_d,\dot{\boldsymbol{y}}_d,\ddot{\boldsymbol{y}}_d,...\right\}$ in (3.14) should also be used as an input for the feed-forward part.

### 3.2.1 Example 1: LiMMS

In order to illustrate the above, we will verify whether the LiMMS can be controlled by LFFC and when this is possible, what inputs of the feed-forward controller we should select. The dynamics of the LiMMS are given by (see Chapter 1):

$$\ddot{y}=\frac{1}{m_L}\left(u-10\dot{y}-10\sin\left(1.6\cdot10^{-2}y\right)\right) \tag{3.15}$$

Where $y$ is the position of the LiMMS. Next, we try to express (3.15) in the form of (3.4). This yields the following equations.

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} &= \begin{bmatrix}0 & 1\\ 0 & \dfrac{-10}{m_L}\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{-10\sin\left(1.6\cdot 10^{-2}x\right)}{m_L}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{1}{m_L}\end{bmatrix}u \\ y &= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix}\end{aligned} \tag{3.16}$$

For the LiMMS, $m$ and $n$ in (3.4) are $m$=1 and $n$=2. Next, we check whether all conditions are satisfied:

- $\boldsymbol{C}$ *may contain only one element,* $\boldsymbol{C}_j$*, that is unequal to zero.* In (3.16), $\boldsymbol{C} = \begin{bmatrix}1 & 0\end{bmatrix}$ , which means that this condition is satisfied.
- $\boldsymbol{C}_j$ *must be invertible.* Since $\boldsymbol{C}_1 = 1$, it is invertible.
- $\boldsymbol{B}_n$ *must be invertible.* In (3.16) $\boldsymbol{B}_2 = \dfrac{1}{m_L}$, which is invertible if $m_L > 0$.

We may conclude that the output of the LiMMS can be controlled by LFFC. When $r$ is the reference position, the desired feed-forward signal is given by:

$$\begin{aligned} u_d &= m_L\left(\frac{10}{m_L}\dot{r} + \frac{10\sin\left(1.6\cdot 10^{-2}r\right)}{m_L} + \ddot{r}\right) \\ &= 10\dot{r} + 10\sin\left(1.6\cdot 10^{-2}r\right) + m_L\ddot{r} \end{aligned} \tag{3.17}$$

We should select $\{r, \dot{r}, \ddot{r}\}$ as inputs of the feed-forward controller.

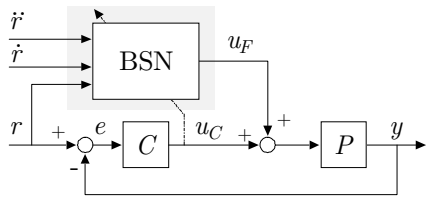



*Figure 3.2: LFFC applied to the LiMMS*

### 3.2.2 Example 2: MSM Plant

We will also examine whether the MSM-plant (figure 3.3), presented in chapter 2, can be controlled by means of standard LFFC (instead of time-indexed LFFC).

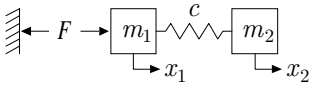



*Figure 3.3: MSM-plant*

When writing the dynamics in the shape of (3.4), the following results:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \left[\begin{array}{cc:cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hdashline \frac{-c}{m_1} & \frac{c}{m_1} & 0 & 0 \\ \frac{c}{m_2} & \frac{-c}{m_2} & 0 & 0 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} F \qquad (3.18)$$

$$y = \left[\begin{array}{cc:cc} 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{cc:cc} x_1 & x_2 & \dot{x}_1 & \dot{x}_2 \end{array}\right]^T$$

Thus, for the MSM plant $m$=2 and $n$=2. Next, we check if (3.18) fulfils all conditions:

- $\boldsymbol{C}$ *may contain only one element,* $\boldsymbol{C}_j$*, that is unequal to zero.* In (3.18) the only element of $\boldsymbol{C}$ that is unequal to zero is $\boldsymbol{C}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ . This condition is satisfied.
- $\boldsymbol{C}_j$ *must be invertible*: $\boldsymbol{C}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ is not invertible. This condition is thus not.
- $\boldsymbol{B}_n$ *must be invertible.* $\boldsymbol{B}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, which is also not invertible. This condition is not satisfied either.

We must conclude that, in spite of the fact that the MSM plant could be controlled by means of time-indexed LFFC, it cannot be controlled by the standard LFFC. Note that this fact cannot be altered by adding a secondary sensor for $x_1$. Hence, the implication of condition 3 of theorem 3.1 is significant.

## 3.3 Stability Issues

The width of the support of the B-splines determines the accuracy of the approximation. B-splines that have a narrow support will be able to approximate a function more accurately than B-splines that have a wide support. To obtain a high tracking accuracy, one might therefore intuitively choose a B-spline distribution that consists of a large number of B-splines with a narrow support. However, in the time-indexed LFFC, the width of the support of the B-splines influences the stability of the control system. Choosing B-splines that had a too narrow support resulted in unstable behaviour. In case of the standard LFFC, the width of the support of the B-splines also determines the stability of the controlled system. Unfortunately, for the standard LFFC, we have not been able to find a theoretical analysis, as was the case for time-indexed LFFC. However, we can use the results of chapter 2 to give an indication of whether the LFFC is stable or not.

The stability analysis of the time-indexed LFFC (see chapter 2) shows that stability can be ensured when:

1. The width of the B-splines defined on the periodic motion time, $d$, is larger than the minimum allowed width, $d_{min}$ [s]. The value of $d_{min}$ depends on the dynamics of the plant and can be obtained by means of a Bode plot of the closed loop system.
2. The learning rate, $\gamma$, satisfies

$$\gamma \leq \frac{2}{\left|-T\left(j\omega\right)\right|_{\infty}} \tag{3.19}$$

   where $-T\left(j\omega\right)$ is the negative complementary sensitivity function.

For specific cases, which we will discuss below, the conditions for the time-indexed LFFC can be transformed into conditions that the standard LFFC must fulfil, in order to ensure stability. This can be seen by looking at the following example in which the LiMMS is controlled by an LFFC that has $\{r, \dot{r}, \ddot{r}\}$ as inputs. Consider a reference motion in which the velocity is constant for some time (figure 3.4a). When the velocity is constant $\ddot{r} = 0\left[\mathrm{ms}^{-2}\right]$. This means that two of the three inputs of the BSN have a constant value. Multi-dimensional B-splines are formed by the tensor product of 1-dimensional B-splines (appendix A). When all inputs

but one are kept constant, the shape of the membership of the multi-dimensional B-spline, equals that of a 1-dimensional B-spline. To clarify this, an example of a 2-dimensional B-spline with inputs $\{r, \dot{r}\}$ and its cross section for constant $\dot{r}$ are shown in figure 3.4b.

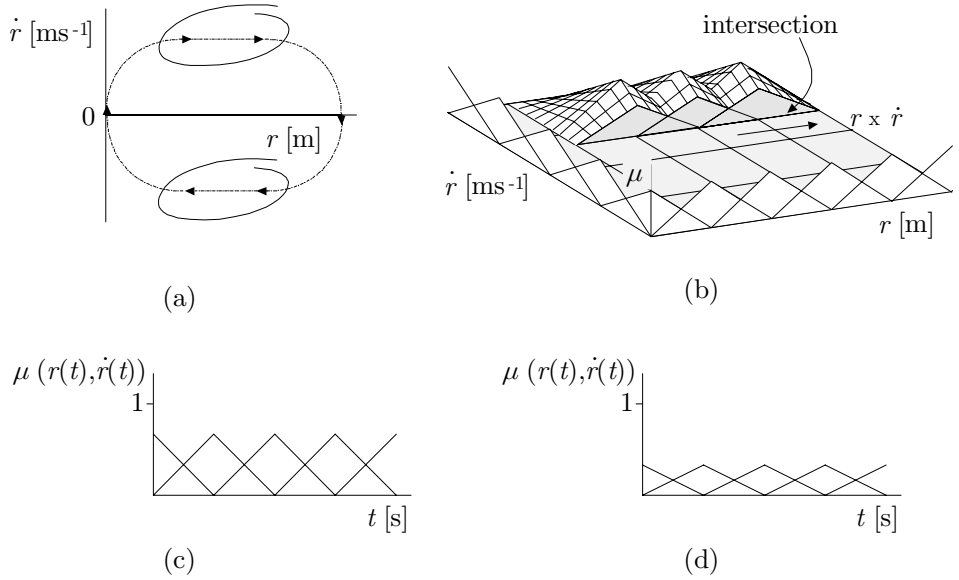
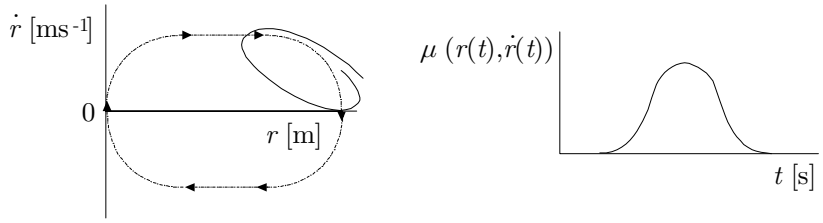



*Figure 3.4: a) Reference motion*
*b) Cross section of a 2-dimensional B-spline*
*c,d) Memberships of the B-splines in the LFFC, as seen in time*

At those sections of the reference motion where the reference velocity is constant, the memberships of the multi-dimensional B-splines have the shape shown in figure 3.4c,d. In the two dimensional case, each input is covered by 4 B-splines. Figure 3.4c shows the membership of two of these, while figure 3.4d shows the membership of the others. The sum of the memberships always equals 1. So the only difference between this case and the time-indexed LFFC is the height of the membership. When the controlled system is linear, the LFFC will thus behave in the same way as a time-indexed LFFC. For the time-indexed LFFC, we know that their support has to be at least $d_{min}$ [s] wide and that the learning rate has to satisfy (3.19), in order to ensure stability. For the multi-dimensional B-splines in the LFFC this means that in order to ensure stability,

1. The supports of the (multi-dimensional) B-splines have to cover at least $d_{min}$ [s] of the reference motion.
2. The learning rate has to satisfy (3.19).

So far, we have only considered the part of the reference motion where the velocity is constant. When looking at other parts of the reference motion, we can observe that the shape of the multi-dimensional B-splines, as seen in time, may differ from the shape shown in figure 3.4c,d. In figure 3.5b, an example of this is presented.




(a) (b)

*Figure 3.5: a) Reference motion*
*b) Membership of a 2-dimensional B-spline, as seen in time*

Obviously, the results of chapter 2 cannot be used for B-splines that have the shape that is shown figure 3.5b. We are thus able to formulate stability conditions for the B-splines for a specific case, i.e. when all inputs but one of the B-splines are constant. For all other cases we are unable to derive stability conditions in terms of the width of the B-splines, as seen in time.

Even if we were able to derive stability conditions in terms of a required width as seen in time, it would be difficult to design a B-spline distribution that satisfies these conditions for a large number of motions. When projecting the reference motion on the input domain of the B-spline network, as done in figure 3.6, it can be seen that the support of some B-splines covers only a very small part of the reference motion. When considering several motions, it is virtually impossible to design a B-spline distribution, such that for all cases the B-splines cover a sufficiently large part of the reference motion. The only option would be to set the learning rate to zero for those B-splines that do not satisfy the condition.

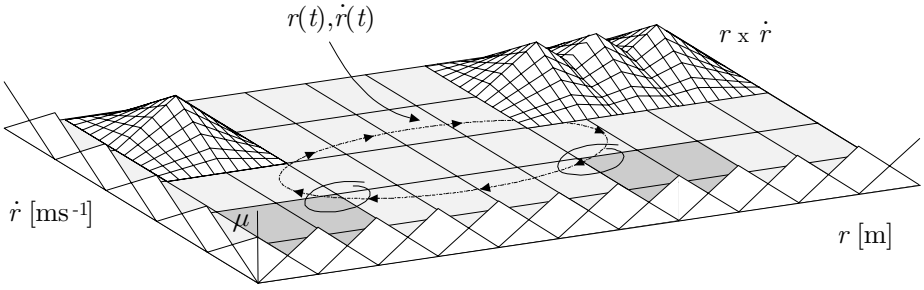



*Figure 3.6: Violation of a (not known) stability criterion*

It may even be unwanted to design a B-spline distribution that satisfies these stability conditions. This may lead to an overly conservative B-spline distribution. The standard BSN is a lattice-based network, which means that the density of B-splines in one direction is equal throughout the network. In figure 3.7, an example of a 2-dimensional BSN is given that has the reference position and velocity as inputs. The density of the B-splines in the direction of the reference position is equal for all values of the reference velocity.

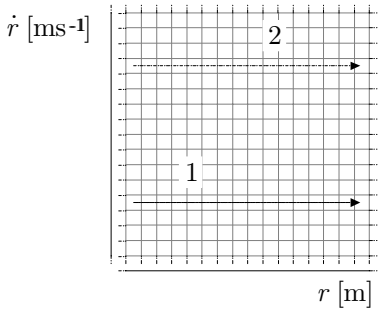



*Figure 3.7: Dense B-spline distribution*

Consider two reference motions that both have a section in which the reference velocity is constant. According to the above, the width of the B-splines, as seen in time, has to be at least $d_{min}$ [s]. When the B-splines cover $d_{min}$ [s] of motion 1 in figure 3.7, this means that for motion 2, which has a higher velocity, they cover less than $d_{min}$ [s]. For motion 2, the LFFC will become unstable. This can be prevented by choosing the width in the $r$-direction such that the B-splines cover at least $d_{min}$ [s] of motion 2 (figure 3.8). However, this results in a very low-frequency feed-forward signal for low velocities. The tracking accuracy for motion 1 will decrease, because the B-spline distribution is chosen such that the LFFC remains stable for motion 2.

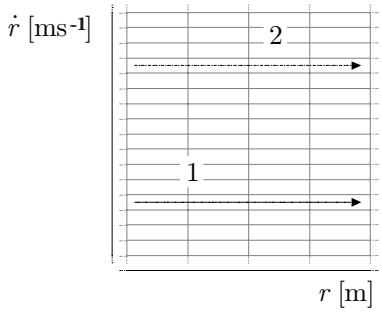



*Figure 3.8: Rough B-spline distribution*

We can conclude that only for motions in which all inputs but one of the B-splines are constant, we are able to ensure stability of the LFFC. For all other types of motions we cannot formulate stability conditions in terms of minimum support width. However, when stability conditions in terms of the width of the B-splines would exist, designing a B-spline distribution that fulfils these conditions is difficult and leads to a poor tracking accuracy. Therefore, we will pursue a different approach, presented in the next section.

## 3.4 Stability Measures

Instead of designing a B-spline distribution that ensures convergence, we can design a B-spline distribution that will yield a high tracking accuracy and then add extra stabilising measures to the BSN. We would like the weights to be adapted such that signals with a frequency content above ${d_{min}}^{-1}$ [Hz] are not stored in the BSN (see chapter 2). In literature, several methods have been presented that influence the adaptation of the weights of the BSN in such way that the input-output mapping gets certain desired properties [Bossley, 1997]. These methods are generally known as regularisation.

### 3.4.1 Regularisation

Regularisation is generally used to improve the generalising ability of a neuro-fuzzy model, in case it is trained with sparsely distributed or noisy data. For example, consider the case shown in figure 3.9, where the training data is distributed such that only the weights of B-splines 1, 2, 4 and 5 are adapted and not those of B-splines 3, 6 and 7. In spite of the fact the we usually want to approximate smooth functions, the input-output mapping is not smooth.

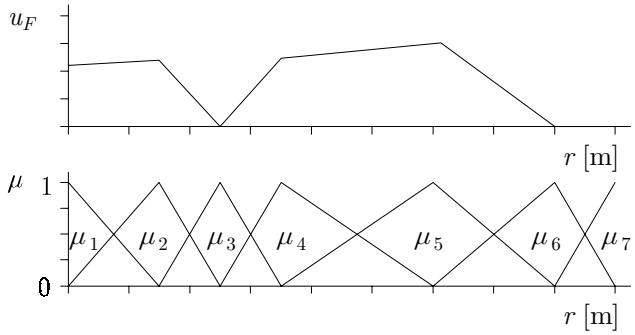



*Figure 3.9: Poor generalising ability due to sparse training data*

Until now, the cost function that underlies the training algorithm only contained the mean squared approximation error. The smoothness of the approximation can be improved by adding an extra regularisation term, $E_R$, to the cost function that learning minimises (3.20) (appendix A).

$$J = MSE\left(y_d - y\right) + \lambda E_R \tag{3.20}$$

In (3.20)$\lambda$ is the regularisation parameter that determines how much regularisation is applied. Common examples of regularisation are [Bishop, 1991]:

- *Zero order regularisation.* In this case, $E_R$ contains the absolute value of the output of the BSN. The growth of weights is penalised, to prevent large values of the weights [Sjöberg, 1995].
- *Second order regularisation.* With this type of regularisation we assume that the function we would like to approximate is smooth. To prevent large fluctuations in the input-output mapping, we let $E_R$ contain the absolute value of the second derivative of the output [Girosi *et al.*, 1995].

Second order regularisation is able to prevent the BSN from learning high-frequency signals and could thus be used to stabilise LFFC. However, finding the right value of $\lambda$ such that the BSN only learns signals up to frequency ${d_{min}}^{-1}$ [Hz] is difficult [Bossley, 1997; Haring, 1998]. Therefore, we propose an alternative form of regularisation for LFFC.

### 3.4.2 Regularisation in LFFC

An LFFC becomes unstable because signals that have a frequency content larger than $d_{min}{}^{-1}$ [Hz] are amplified in the learning loop. This means that to stabilise an LFFC, we have two options:

- *Add a regularisation mechanism, such that the BSN is unable to learn signals that have a frequency larger than $d_{min}{}^{-1}$ [Hz].* As discussed, it is difficult to choose the right value of the regularisation parameter.
- *Remove the signal components that cause instability from the learning loop.* In other words, add a learning filter that removes all signal components from $u_C$ that have a frequency larger $d_{min}{}^{-1}$ [Hz], and pass all others.

In this research, the second approach is taken. We propose to use a time-indexed LFFC as a learning filter (figure 3.10).

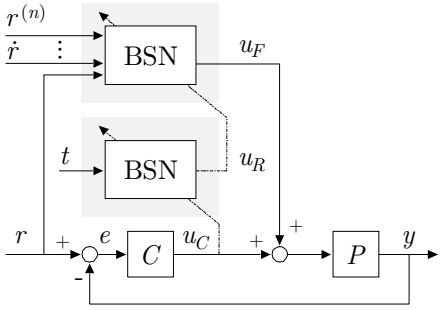



*Figure 3.10: LFFC with regularisation network*

A BSN can be used as a filter because of the following. Between 2 knots, the input-output relation of a 1-dimensional 2nd order time-indexed BSN consists of a first-order polynomial function (appendix A). The accuracy of an approximation depends on the width of the B-splines (see chapter 2). By choosing the width appropriately, the BSN is unable to approximate signals that have a frequency larger than $d_{min}{}^{-1}$ [Hz]. We propose to choose the width equal to $2\,d_{min}$ [s]. In this case, the BSN tries to approximate one period of a sine-shaped signal that has a frequency equal to $d_{min}{}^{-1}$ [Hz], using one first order polynomial function. Figure 3.11 shows the resulting approximation. We may thus conclude that, using this B-spline distribution, the BSN is unable to approximate signals that have a frequency larger than $d_{min}{}^{-1}$ [Hz] and that they are thus removed from the learning signal.

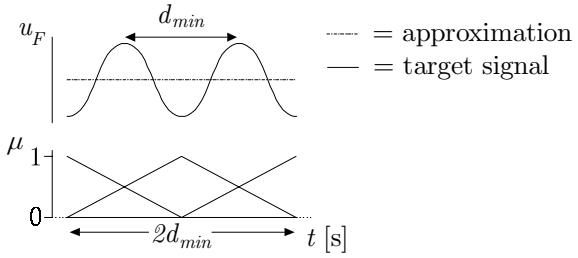



*Figure 3.11: BSN approximation of a high-frequency target signal*

Ideally, the learning filter should pass any signals that have a frequency lower than $d_{min}{}^{-1}$ [Hz]. However, for the time-indexed BSN this is not the case. Figure 3.12 shows that by making a BSN approximation of a sine-shaped signal with a frequency slightly smaller than $d_{min}{}^{-1}$ [Hz] also a part of these signal components is removed. The result may be that by adding regularisation to the LFFC, the tracking accuracy gets worse.

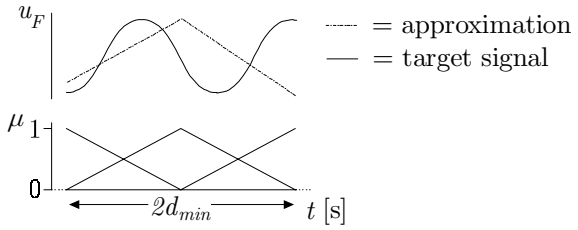



*Figure 3.12: BSN approximation of a medium-frequency target signal*

Signal components that have a much lower frequency than $d_{min}{}^{-1}$ [Hz] can be approximated accurately by the time-indexed BSN and are thus passed.

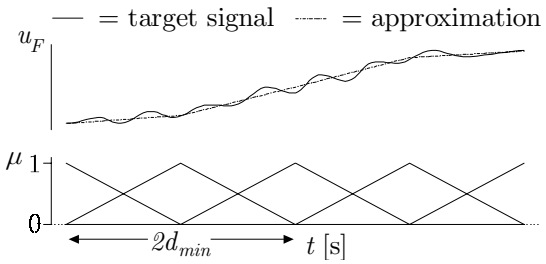



*Figure 3.13: BSN approximation of the low-frequency part of a target signal*

In order not to change the amplitude of the learning signal, the learning rate of the time-indexed BSN is chosen equal to 1.

Roughly speaking, the adaptation of the feed-forward signal of the LFFC is now performed in two different steps, see figure 3.14 and 3.15.

1. The weights of the time-indexed BSN are adapted on the basis of $u_C$. The BSN is trained off-line (in a way that we will explain in detail in the following), using a learning rate equal to 1. This yields an output $u_R$.

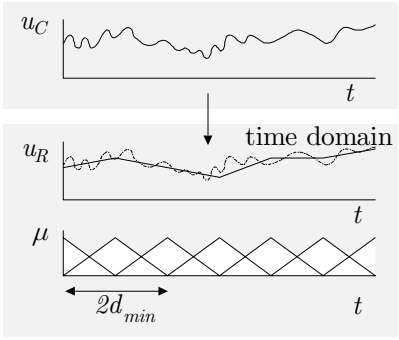



*Figure 3.14: Training the time-indexed BSN*

2. After the weights of the time-indexed BSN have been adapted, the LFFC is trained, using $u_R$ as a learning signal

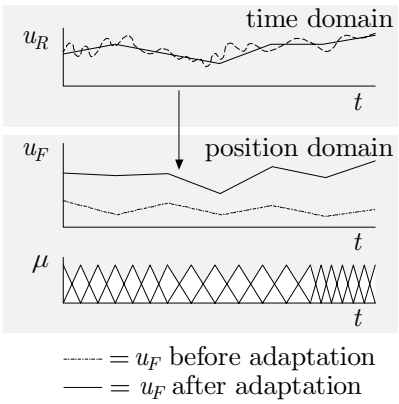



*Figure 3.15: Adaptation of the weights of the LFFC*

In the following we will discuss the regularisation mechanism in detail. This will be done on the basis of two examples. In the first example, the width of the support of the B-splines in the LFFC is smaller than the width of the support of the B-splines in the time-indexed BSN. Next, we consider the case where the width of the support of the B-splines in the LFFC is the largest.

The regularisation mechanism can be split up in the following phases:

- $t$=0 [s]. At the beginning of the experiment the weights of the time-indexed BSN are equal to 0. We assume that the weights of the LFFC have some initial value, obtained by training in a previous experiment. However, learning has not converged yet and the LFFC requires more training.
- 0 [s] $< t <$ $d_{min}$ [s]. Figure 3.16a shows the output of the feed-forward part and the feedback signal that results. The time-indexed BSN is trained off-line, which means that the adaptation of weights 1 and 2 are calculated according to

$$\Delta w_j = \gamma \frac{\sum_{i=1}^{N_s} \mu_j(t_i) e\left(t_i\right)}{\sum_{i=1}^{N_s} \mu_j(t_i)} \tag{3.21}$$

- $t$=$d_{min}$ [s]; The membership of B-spline 1 of the time-indexed BSN becomes equal to 0. Since the value of $t$ increases, this B-spline will never be visited again (the value of $\Delta w_j$ in (3.21) for $j$=1 remains constant). Therefore, we choose to update the weight of B-spline 1 at this point in time.
- $d_{min}$ [s] $< t <$ $2d_{min}$ [s]. The adaptation of weights 2 and 3 are calculated according to (3.21). No special actions are taken further (figure 3.16b).

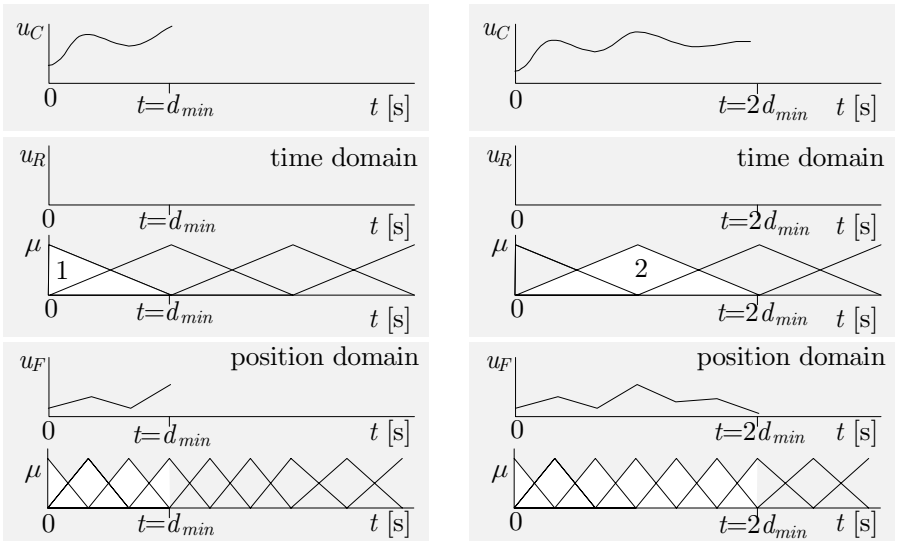



*Figure 3.16: a) Regularisation after $t$=$d_{min}$ [s]*
*b) Regularisation after $t$=$2d_{min}$ [s]*

– $t$=$2d_{min}$ [s]. The membership of B-spline 2 of the time-indexed BSN becomes equal to zero. This means that we can apply its weight adaptation. After adapting the weights of B-spline 1 and 2, we have obtained a part of the regularised learning signal, $u_R$ (see figure 3.17a). Note that $u_R$ does not contain frequency components that have a frequency larger than $d_{min}{}^{-1}$ [Hz], even though these were present in $u_C$. The newly obtained $u_R$ fully covers the support of B-splines 1, 2 and 3 of the LFFC. This means that the weights of these B-splines can be adapted on the basis of $u_R$.

$$\Delta w_k = \gamma \frac{\sum_{i=1}^{N_S} \mu_k\left(r\left(t_i\right)\right) u_R\left(t_i\right)}{\sum_{i=1}^{N_S} \mu_k\left(r\left(t_i\right)\right)}, \quad k = 1,2,3 \tag{3.22}$$

Because $u_R$ becomes available after B-splines 1, 2 and 3 of the LFFC have been visited, their memberships (or the reference trajectory) need to be stored. Figure 3.17b shows the adapted feed-forward signal that results.

– After this, the whole sequence repeats for time-domain B-splines 2 and 3, etc.

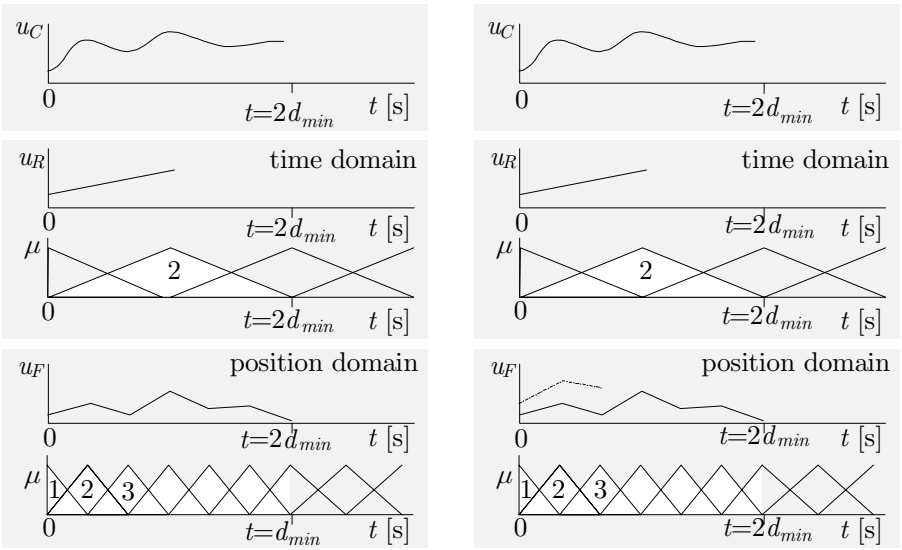



*Figure 3.17: a) Regularisation at $t$=2$d_{min}$ [s], weight adaptation of the time-indexed BSN*
*b) Regularisation at $t$=2$d_{min}$ [s], weight adaptation of the LFFC*

We will perform a similar discussion of the regularisation mechanism, for the case where the B-splines in the LFFC have the largest width.

- $t$=0 [s]. Again, we assume that the weights of the time-indexed BSN are equal to 0 and that the weights of the LFFC have some initial value.
- 0 [s] $< t <$ $d_{min}$ [s]. No special actions are taken (figure 3.18a).
- $t$=$d_{min}$ [s]. Since B-spline 1 will not be visited again, its weight adaptation is applied.
- $d_{min}$ [s] $< t <$ 2$d_{min}$ [s]. No special actions are taken (figure 3.18b).
- $t$=2$d_{min}$ [s]. The weight of B-spline 2 is adapted. This gives the first part of the regularised learning signal (figure 3.18b). Since the obtained part of $u_R$ does not fully cover the support of one of the B-splines in the LFFC, none of the weights of the LFFC can be adapted. We can only adapt a weight in the LFFC, when we have a $u_R$ that covers the entire support of the corresponding B-spline. So, we continue to train the time-indexed BSN.

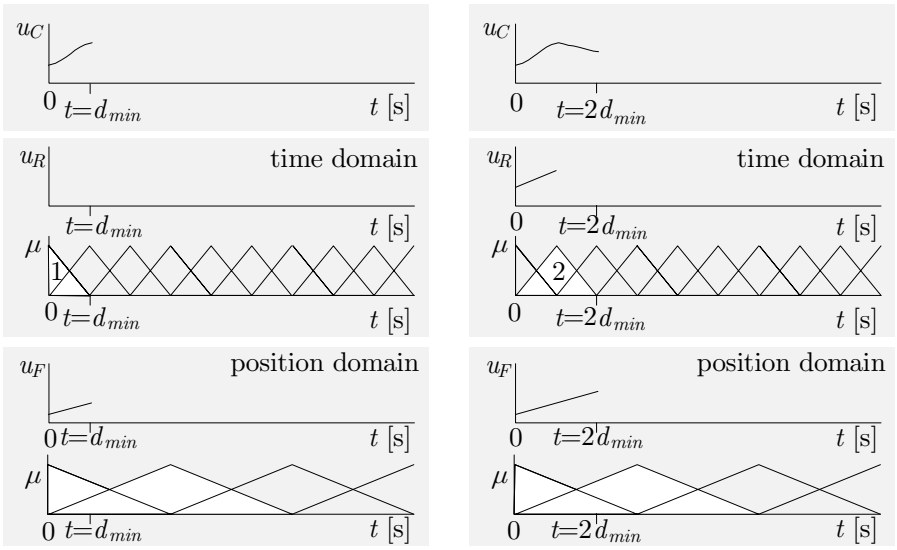



*Figure 3.18: a) Regularisation after $t{=}d_{min}$ [s]*
*b) Regularisation at $t{=}2d_{min}$ [s], weight adaptation of the time-indexed BSN*

- $t{=}4d_{min}$ [s]; At this point of time, the weights of B-spline 1, 2 and 3 of the time-indexed BSN have already been trained. By adapting the weight of B-spline 4 in the time-indexed BSN, a regularised learning signal has been obtained, that fully covers the support of B-spline 1 in the LFFC (figure 3.19a). Therefore, the weight of B-spline 1 in the LFFC is adapted.
- $t{=}7d_{min}$ [s] (at $t{=}5d_{min}$ [s] and $t{=}6d_{min}$ [s] the weights of B-splines 5 and 6 have been adapted). The weight of B-spline 7 in the time-indexed BSN is adapted (figure 3.19b). We have now obtained a regularised learning signal that covers the entire support of B-spline 2 in the LFFC. The weight of B-spline 2 is adapted on the basis of $u_R$. Figure 3.19c shows the adapted feed-forward signal. Note that in this case both $u_R$ and the membership of the B-splines in the LFFC need to be stored.

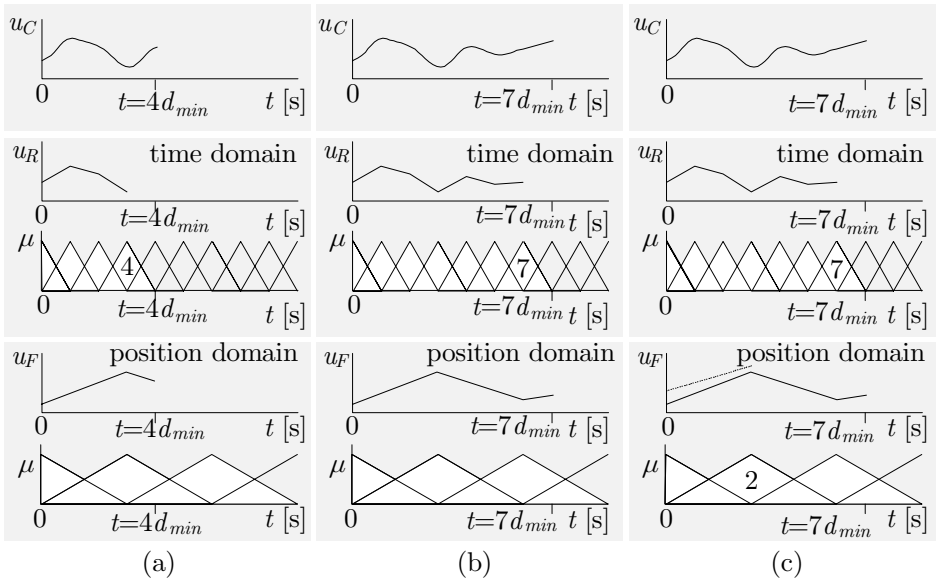



*Figure 3.19: a) Regularisation at t=4$d_{min}$ [s], weight adaptation of the time-indexed BSN*
*b) Regularisation at t=7$d_{min}$ [s], weight adaptation of the time-indexed BSN*
*c) Regularisation at t=7$d_{min}$ [s], weight adaptation of the LFFC*

In figure 3.20, an alternative representation of the regularisation mechanism is given. The supports of the B-splines of the time-indexed BSN have been projected on the reference motion. The learning signal for the LFFC consists of piecewise first order polynomial functions between the knots of the B-splines of the time-indexed BSN.

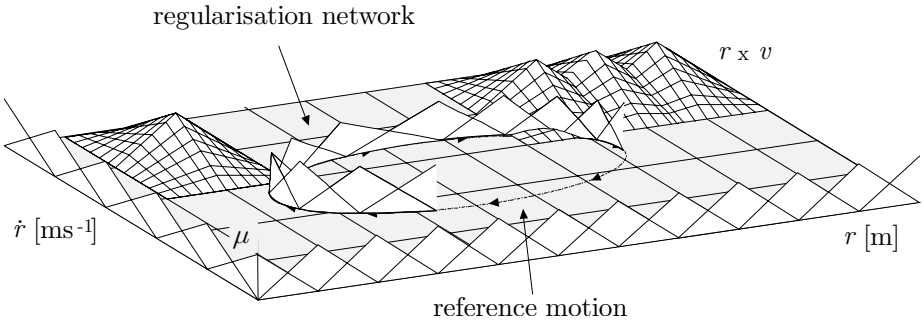



*Figure 3.20: Alternative representation of the regularisation mechanism*

In short, regularisation in LFFC can be described as:

**Algorithm 3.1 (Regularisation in LFFC)**

1. Add a time-indexed BSN to the LFFC, which filters the learning signal. The width of the support of the B-splines in the time-indexed BSN should be chosen equal to $2d_{min}$ [s].
2. Choose a random training motion.
3. Train the time-indexed BSN in an off-line way, with a $\gamma$=1. Use the output of the time-indexed BSN, $u_R(t)$, as a learning signal for the LFFC. In detail, this can be described by the following steps:

   3.1 At $t$=0 [s], initialise the weights of the time-indexed BSN to 0. Define n=1.

   3.2 While $t<n\ d_{min}$ [s], use the LFFC for control and calculate the adaptation of the weights in the time-indexed BSN. Of each B-spline in the LFFC that contributes to the feed-forward signal store $\{t, \mu(t)\}$.

   3.3 At $t=nd_{min}$ [s], the membership of one of the B-splines in the time-indexed BSN becomes equal to 0. Adapt the weight of that B-spline. Herewith, we have obtained a new part of $u_R(t)$.

   3.4 When $u_R(t)$ fully covers the support of one of the B-splines in the LFFC that have been stored, adapt the corresponding weight using $u_R(t)$ as a learning signal and $\{t, \mu(t)\}$ from the memory.

   3.5 $n$=$n$+1, go to step 3.2.

### 3.4.3 Simulations

In order to validate the above algorithm, simulations will be performed in which the LiMMS is controlled by an LFFC with a regularisation algorithm 3.1. For the values of the parameters in the simulation model and the controller, we refer to section 2.6.

The LiMMS repeatedly has to track two reference motions. The reference motion shown in figure 3.21a corresponds to motion 2 of figures 3.7 and 3.8. This is the "fast" motion that requires B-splines that have a large support, in order to ensure

stability. The reference motion that is given in figure 3.21b corresponds to motion 1. This motion allows a more dense B-spline distribution.

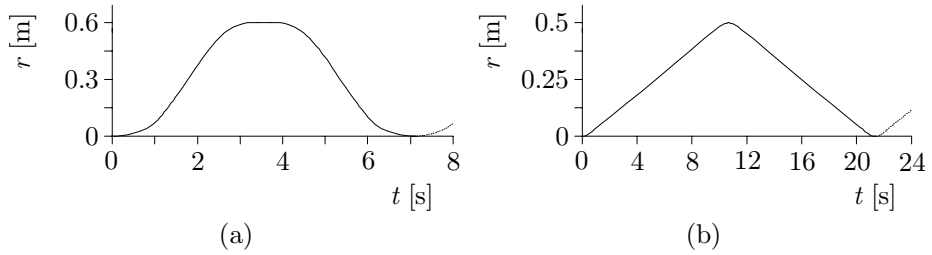



*Figure 3.21: a) "Fast" reference motion*
*b) "Slow" reference motion*

**Simulation 3.1 (feedback control only, "fast" reference motion)**

In the first simulation, the LiMMS is controlled by the feedback controller only. In figure 3.22 the tracking error is presented. In the tracking error two components can be observed. The small sine-shaped ripple in the tracking error is caused by cogging. The larger part has the shape of the reference acceleration. The feedback controller is unable to realise the desired acceleration.

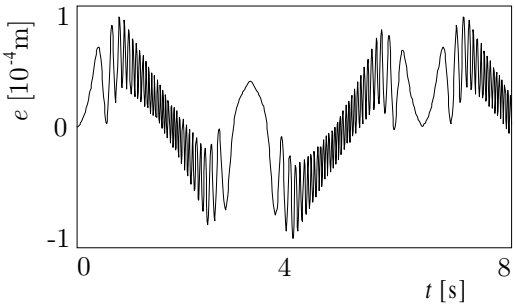



*Figure 3.22: Tracking error without LFFC, "fast" motion*

**Simulation 3.2 (feedback control only, "slow" reference motion)**

The tracking error of the "slow" motion, is presented in figure 3.23. The major part of the tracking error is caused by cogging.

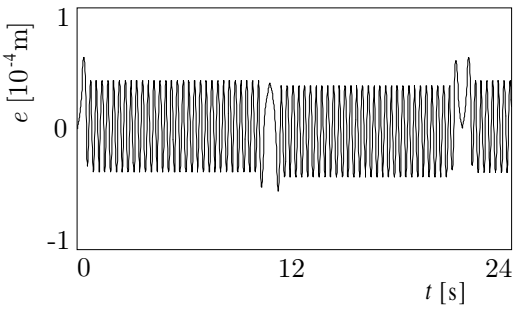



*Figure 3.23: Tracking error without LFFC, "slow" motion*

Next, an LFFC will be applied. The structure of the LFFC has been presented in figure 3.2. In chapter 2, it has been derived that the minimum width of the support of the B-splines, as seen in time, is (chapter 2):

$$d_{\min} = \frac{2\pi}{220}[s] = 0.0288 \text{ [s]} \tag{3.23}$$

First, we apply LFFC without regularisation and try to select a B-spline distribution that will avoid instability. In order to keep the number of B-splines that contribute to the output for less than $d_{min}$ to a minimum, we select the B-spline distribution as follows:

- At the highest reference velocity, $\dot{r}_{max}$, it should take at least $d_{\min}$[s] to cross a B-spline in the direction of the reference position.
- At the highest reference acceleration, $\ddot{r}_{max}$, it should take at least $d_{\min}$[s] to cross a B-spline in the direction of the reference velocity.
- At the highest reference jerk, $\dddot{r}_{max}$, it should take at least $d_{\min}$[s] to cross a B-spline in the direction of the reference acceleration.

This way of selecting the B-spline distribution guarantees that the supports of the B-splines are wide enough when only one of the inputs of the BSN varies, while the other inputs are constant. Such a distribution cannot prevent situations as described in figure 3.6, where all inputs vary. It may thus occur that some B-splines do not cover $d_{min}$ [s] of the reference motion.

For the slow motion the above yields:

$$\dot{r}_{max} = 0.05\left[\mathrm{ms}^{-1}\right],\ \ddot{r}_{max} = 0.15\left[\mathrm{ms}^{-2}\right],\ \dddot{r}_{max} = 0.5\left[\mathrm{ms}^{-3}\right] \tag{3.24}$$

and for the fast motion:

$$\dot{r}_{max} = 0.31\left[\mathrm{ms}^{-1}\right],\ \ddot{r}_{max} = 0.39\left[\mathrm{ms}^{-2}\right]\ \dddot{r}_{max} = 0.5\left[\mathrm{ms}^{-3}\right] \tag{3.25}$$

This yields the following minimum width of the support on each of the input domains:

$$\begin{aligned} d_{r,min} &= 0.31\left[\mathrm{ms}^{-1}\right]\cdot 2.88\cdot 10^{-2}\,[\mathrm{s}] = 8.928\cdot 10^{-3}\,[\mathrm{m}] \\ d_{\dot{r},min} &= 0.39\left[\mathrm{ms}^{-2}\right]\cdot 2.88\cdot 10^{-2}\,[\mathrm{s}] = 1.123\cdot 10^{-2}\left[\mathrm{ms}^{-1}\right] \\ d_{\ddot{r},min} &= 0.5\left[\mathrm{ms}^{-3}\right]\cdot 2.88\cdot 10^{-2}\,[\mathrm{s}] = 1.44\cdot 10^{-2}\left[\mathrm{ms}^{-2}\right] \end{aligned} \tag{3.26}$$

**Simulation 3.3 (LFFC, conservative distribution, "fast" reference motion)**

First, we choose a conservative B-spline distribution, in which the widths are slightly larger than the minimum widths: $d_r = 9.3{\cdot}10^{-3}$ [m], $d_v = 1.4{\cdot}10^{-1}$ [ms$^{-1}$], $d_a = 1.6{\cdot}10^{-2}$ [ms$^{-2}$]. The learning rate is chosen 0.9. In figure 3.24 the tracking error after 2000 [s] is shown. It can clearly be seen that the LFFC has removed the part of the tracking error that had the shape of the reference acceleration. The sine-shaped part, caused by cogging, is not removed completely.

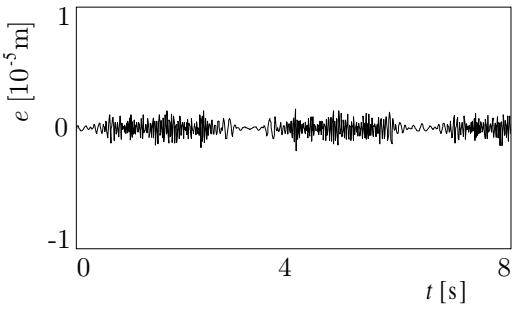



*Figure 3.24: Tracking error (conservative distribution, " fast" reference motion)*

**Simulation 3.4 (LFFC, conservative distribution, "slow" reference motion)**

Using the same LFFC for the "slow" motion, gives the tracking error depicted in figure 3.25. Due to the large supports of the B-splines, the LFFC has not been able to compensate for cogging.

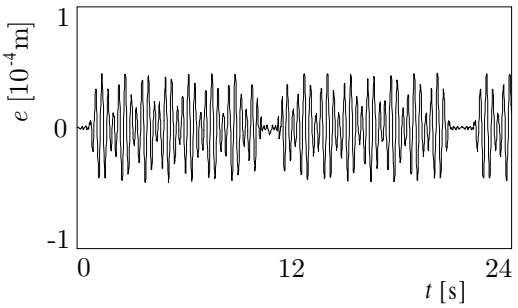



*Figure 3.25: Tracking error (conservative distribution, " slow" reference motion)*

The LFFC is stable for both types of motions. However, due to the large width of the B-splines on the reference position, the LFFC is not able to compensate the cogging forces. This can be seen in figure 3.26, where the desired compensation of the cogging, $u_d$, and the actual compensation, $u_F$, are shown.

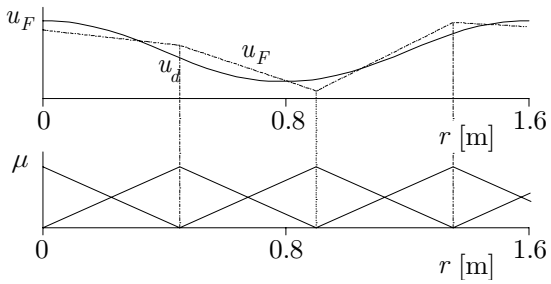



*Figure 3.26: Desired and actual compensation of cogging*

**Simulation 3.5 (LFFC, dense distribution, "slow" reference motion)**

To obtain better tracking for slow motions, the width of the B-splines on the reference position is decreased to $d_r = 2.8 \cdot 10^{-3}$ [m]. Using this distribution, a much smaller tracking error results after performing a slow motion for 2000 [s] (figure 3.27).

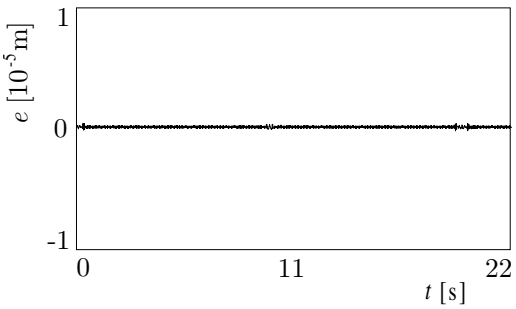



*Figure 3.27: Tracking error (dense distribution, " slow" reference motion)*

**Simulation 3.6 (LFFC, dense distribution, "fast" reference motion)**

When this B-spline distribution is used to track fast motions, the LFFC becomes unstable. The tracking error depicted in figure 3.28 grows as learning continues.

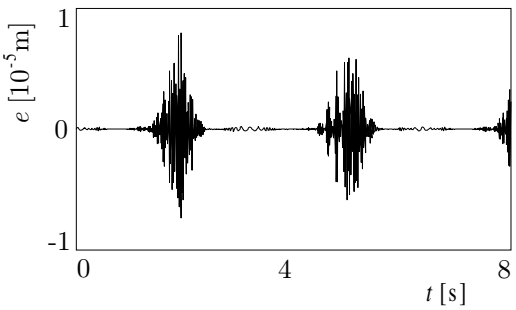



*Figure 3.28: Tracking error (dense distribution, " fast" reference motion)*

**Simulation 3.7 (LFFC, dense distribution, regularisation, "fast" reference motion)**

To overcome this problem we add the regularisation mechanism to the LFFC of simulation 3.6. The width of the B-splines in the time-indexed BSN is chosen $2 \cdot 0.0288[s]=0.059$ [s]. After performing the fast motion with the regularised LFFC for 2000 [s], no sign of instability can be seen (figure 3.29). We may thus conclude that the regularisation mechanism stabilised the LFFC.

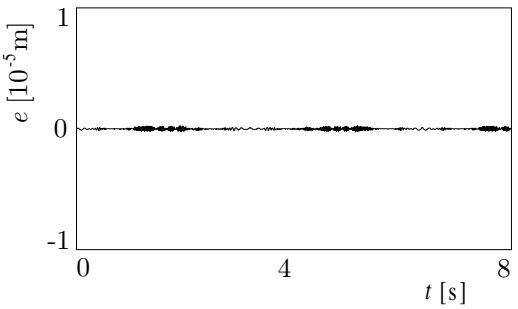



*Figure 3.29: LFFC with regularisation, "fast" motion*

**Simulation 3.8 (LFFC, dense distribution, regularisation, "slow" reference motion)**

Next, we examine whether the regularisation mechanism may cause a loss of performance. Therefore, simulation 3.5 is performed again with a regularised LFFC. In figure 3.30 the tracking error is presented after performing the slow motion for 2000 sec. This tracking error is comparable to the tracking error that was obtained without the regularisation mechanism (figure 3.27). It may thus be concluded that the regularisation mechanism does not cause a significant loss of performance for slow motions.

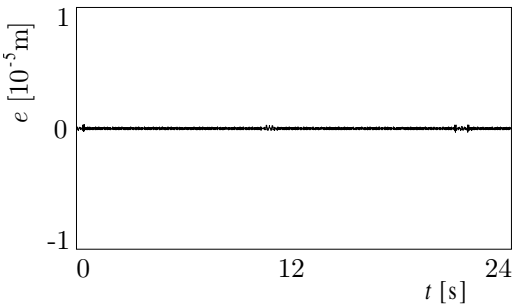



*Figure 3.30: LFFC with regularisation, "slow" motion*

## 3.5 Discussion

We have not been able to derive general stability conditions for LFFC, like we have done for the time-indexed LFFC in chapter 2. Only in specific situations, when all inputs but one of the B-splines are constant, the outcome of chapter 2 can be used to indicate whether a certain B-spline distribution will yield unstable

behaviour or not. However, even if we were able to formulate stability conditions in terms of the B-spline support width, the B-spline distribution that fulfils these conditions would be very conservative, resulting in a large tracking error. Therefore, instead of stabilising the LFFC by means of a sufficiently wide B-spline support, we proposed to add extra stabilising measures to the BSN.

This can, for example, be done by means of regularisation. Regularisation usually means changing the learning mechanism of the BSN by adding an extra term to the cost function that must be minimised. This extra term penalises undesired input-output behaviour (e.g. large output values). In case of LFFC we would penalise learning of signals that have a frequency larger than ${d_{min}}^{-1}$ [Hz]. This is done by penalising the output curvature. However, it is difficult to determine how much the output curvature should be penalised, such that only signals that have a frequency lower than ${d_{min}}^{-1}$ [Hz] are learned. Therefore, an alternative form of regularisation was proposed.

Regularisation is performed by filtering the learning signal of the LFFC by a time-indexed BSN. This approach is based on the fact that the smoothness of an approximation is determined by the width of the support of the B-splines. By choosing the width of the support of the B-splines in the time-indexed BSN sufficiently wide, all frequency components that have a frequency larger than ${d_{min}}^{-1}$ [Hz], and therefore cause unstable behaviour, are removed from the learning signal. By removing these frequency contents from the learning loop, the LFFC remains stable. To ensure that frequency components that have a frequency larger than ${d_{min}}^{-1}$ [Hz] cannot be approximated, the width of the support of the B-splines in the time-indexed BSN is chosen equal to $2\,d_{min}$ [s].

Simulations showed that learning diverged when the width of the support of B-splines in the LFFC became too small. Adding the regularisation mechanism stabilised the LFFC. However, by choosing the width of the B-splines in the time-indexed LFFC equal to $2\,d_{min}$ [s], also signals that have a frequency that is slightly lower than ${d_{min}}^{-1}$ [Hz] are (partly) removed from the learning signal. This may cause loss of tracking performance, even though this was not observable in the simulations that were performed.

The regularisation mechanism proposed here is a rather heuristic stabilising measure. While a proper choice of the width of the B-splines in the time-indexed BSN can stabilise LFFC, little is known about the filtering properties at frequencies that are slightly lower than ${d_{min}}^{-1}$ [Hz]. It is unclear whether the selected width of $2\,d_{min}$ [s] is a proper choice or if it is a conservative one. This makes the results preliminary. Further research on this topic is desired.

# 4 Non-Repetitive Motions: Parsimonious LFFC

## 4.1 Introduction

In the standard LFFC, the reference position and derivatives/integrals thereof, are used as inputs of the feed-forward part. In general, this will result in a BSN that has multiple inputs. When the number of inputs of a BSN is large, the problems associated with the *curse of dimensionality* occur that make the LFFC less suited for practical application. In section 4.2 we will show that, in case the inputs of the feed-forward part are selected as proposed in chapter 3, we quickly end up with a BSN that has a large number of inputs.

Similar problems are encountered in the field of neuro-fuzzy modelling [Brown and Harris, 1994]. Here, a neuro-fuzzy network, such as a BSN, is used to model the plant dynamics. The BSN is trained on the basis of observed input-output data. When the inputs are chosen in a straightforward way (similar to the way the inputs of the LFFC are chosen in chapter 3), one encounters problems associated with the curse of dimensionality. To keep these problems to a minimum, it has been proposed to choose the inputs of the BSN and the B-spline distribution in an alternative way. This design approach requires more detailed knowledge of the plant dynamics. This knowledge may either be present in the form of a qualitative a-priori model or can be obtained in an automated way by means of experiments.

In section 4.3, we examine whether the same approach can also be used to overcome the problems associated with the curse of dimensionality in LFFC [Idema, 1996; Vries *et al.*, 1998].

## 4.2 Curse of Dimensionality in LFFC

In chapter 3, it was proposed to select the inputs of the BSN by expressing the plant in the form of (3.4). The desired feed-forward signal was then given by (3.14). Each term $\left\{\boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, ...\right\}$ existing in (3.14) should be selected as an input of the BSN. This method quickly results in a large number of inputs. We will show this, by selecting the inputs of the BSN for a class of plants for which this is especially the case, i.e. rigid robot manipulators.

### 4.2.1 Illustrative Example: Rigid Robot Manipulator

The dynamics of a rigid revolute-joint robot manipulator are well known [Craig, 1989]:

$$\boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + \boldsymbol{D}\dot{\boldsymbol{\theta}} + \boldsymbol{S}\,\mathrm{sgn}\left(\dot{\boldsymbol{\theta}}\right) + \boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{u} \tag{4.1}$$

where $\boldsymbol{\theta}$ is the vector of joint angles, $\boldsymbol{M}(\boldsymbol{\theta})$ is the inertia-matrix, $\boldsymbol{C}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)$ models Coriolis and centrifugal effects, $\boldsymbol{D}$ is a diagonal matrix containing viscous friction coefficients, $\boldsymbol{S}$ is a diagonal matrix containing Coulomb friction coefficients, $\boldsymbol{G}(\boldsymbol{\theta})$ describes gravitational load and $\boldsymbol{u}$ is the vector of joint motor torques. In case of a 2-DOF robot manipulator, shown in figure 4.1, the dynamics are given by (4.2):

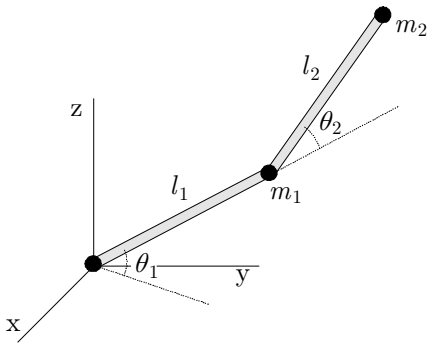



*Figure 4.1: 2-DOF robot manipulator*

$$\begin{bmatrix} \frac{J_1}{r_1^2} + m_2\left(l_2^2 + 2l_1l_2\cos\left(\theta_2\right) + l_1^2\right) + l_1^2 m_1 & l_2^2 m_2 + 2l_1l_2m_2\cos\left(\theta_2\right) \\ l_2^2 m_2 + 2l_1l_2m_2\cos\left(\theta_2\right) & \frac{J_2}{r_2^2} + l_2^2 m_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} +$$
$$\begin{bmatrix} -2l_1l_2m_2\sin\left(\theta_2\right)\dot{\theta}_2 & -l_1l_2m_2\sin\left(\theta_2\right)\dot{\theta}_2 \\ l_1l_2m_2\sin\left(\theta_2\right)\dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \tag{4.2}$$
$$\begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} \operatorname{sgn}\left(\dot{\theta}_1\right) \\ \operatorname{sgn}\left(\dot{\theta}_2\right) \end{bmatrix} + \begin{bmatrix} gm_2l_2\cos\left(\theta_1+\theta_2\right) + gl_1\cos\left(\theta_1\right)\left(m_1+m_2\right) \\ gl_2m_2\cos\left(\theta_1+\theta_2\right) \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

where

$m_{1,2}$: mass of link 1,2
$l_{1,2}$: length of link 1,2
$J_{1,2}$: inertia of motor 1,2
$r_{1,2}$: gear ratio of motor 1,2
$d_{1,2}$: viscous friction in joint 1,2
$s_{1,2}$: Coulomb friction joint 1,2

Writing (4.1) in the shape of (3.4) results in :

$$\begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & -\boldsymbol{M}(\boldsymbol{\theta})^{-1}\boldsymbol{C}\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) - \boldsymbol{M}(\boldsymbol{\theta})^{-1}\boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} +$$
$$\begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{M}(\boldsymbol{\theta})^{-1}\boldsymbol{G}(\boldsymbol{\theta}) - \boldsymbol{M}(\boldsymbol{\theta})^{-1}\boldsymbol{S}\operatorname{sgn}\left(\dot{\boldsymbol{\theta}}\right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}(\boldsymbol{\theta})^{-1} \end{bmatrix} \boldsymbol{u} \tag{4.3}$$
$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix}$$

Where $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T$, $\boldsymbol{\theta} \in \mathbb{R}^2$ and $\boldsymbol{M}(\boldsymbol{\theta})$ is a positive definite matrix [Lewis *et al.*, 1993].

Next, we check whether the conditions derived in chapter 3 are satisfied.

– $\boldsymbol{C}$ *may contain only one element,* $\boldsymbol{C}_j$*, that is unequal to zero.* In (4.3) it can be seen that $\boldsymbol{C}$ has the correct form, namely:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \tag{4.4}$$

– $\boldsymbol{C}_j$ *must be invertible.* Since, $\boldsymbol{C}_1 = \boldsymbol{I}$, it is invertible. This condition is satisfied.

– $\boldsymbol{B}_n(\boldsymbol{\theta})$ *must be invertible.* From (4.3) it follows that:

$$\boldsymbol{B}_n(\boldsymbol{\theta}) = \boldsymbol{M}(\boldsymbol{\theta})^{-1} \tag{4.5}$$

$\boldsymbol{B}_n(\boldsymbol{\theta})$ is thus invertible. This condition is satisfied as well.

We may conclude that the joint angles of a rigid robot manipulator can be controlled by means of LFFC. The desired feed-forward signal is:

$$\boldsymbol{u}_d = \boldsymbol{C}\left(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d\right)\dot{\boldsymbol{\theta}}_d + \boldsymbol{D}\dot{\boldsymbol{\theta}}_d + \boldsymbol{G}\left(\boldsymbol{\theta}_d\right) + \boldsymbol{S}\,\mathrm{sgn}\left(\dot{\boldsymbol{\theta}}_d\right) + \boldsymbol{M}\left(\boldsymbol{\theta}_d\right)\ddot{\boldsymbol{\theta}}_d \tag{4.6}$$

From (4.6), it follows that in a straightforward implementation, the inputs of the BSN must consist of $\left\{\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d\right\}$ (figure 4.2), which means 6 inputs for a 2 DOF manipulator.

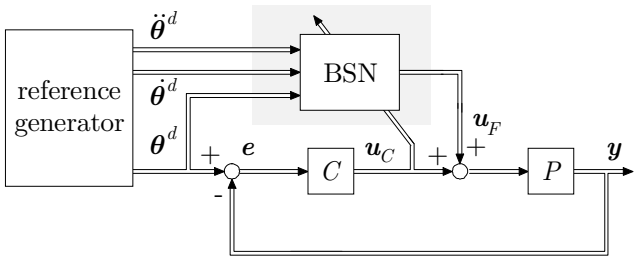



*Figure 4.2: LFFC of a rigid robot*

### 4.2.2 Curse of Dimensionality

From the previous example, one can see that even for a relatively simple system, such as the 2 DOF robot manipulator, the BSN has a large number of inputs, in this case 6. When considerable non-linearities are present, each input needs to be covered with a significant number of basis functions. In this situation, one encounters problems that are associated with the *curse of dimensionality* [Bellman, 1961; Brown and Harris, 1994; Bossley *et al.*, 1996]. As briefly discussed in chapter 1, these problems are:

- *A large number of network weights.* The number of weights in a BSN depends on the dimension of the input in an exponential way. If, in case of the 2-DOF robot manipulator, the number of B-splines on each of the inputs of the BSN is, for example, 15, the total number of network weights is:

$$N_T = 15^6 = 11.390.625 \tag{4.7}$$

  Since the LFFC is usually implemented in an embedded computer, the amount of computer memory is limited. Storing the number of weights of (4.7), as real numbers (requiring 4 bytes of memory each) requires approximately 45 MB of computer memory. This makes it difficult to implement the LFFC on smaller embedded computers. It will be clear that for a 6 DOF manipulator this problem is dramatic.

- *Large training sets.* To properly train the BSN, a number of training samples should be present for each B-spline. A large number of B-splines means that a large number of training samples is needed. Slow learning will be the result.

- *Poor generalising ability.* The generalising ability is the ability to produce a sensible output for motions that are similar to previously trained motions. When the dimension of the input of a BSN is large, a poor generalising ability results. This can easily be seen by the following example. Consider a system that is subject to two independent disturbances, one depending on the velocity and one on the position. The feed-forward signal that compensates these disturbances can be stored in a 2-dimensional BSN that has the reference velocity and position as inputs (figure 4.3). Next, two training motions are performed that contain each possible reference position and reference velocity. In figure 4.3, the B-splines of which the weights are adapted during these training motions are marked grey. It can be seen that the LFFC is not able to generate a feed-forward signal for a 3$^{\text{rd}}$ motion even

if it is similar to the training motions. Thus, in spite of the fact that all possible velocities and positions have been presented during training, and thus each possible disturbance has occurred, the LFFC is not able to compensate the disturbances for a new motion.

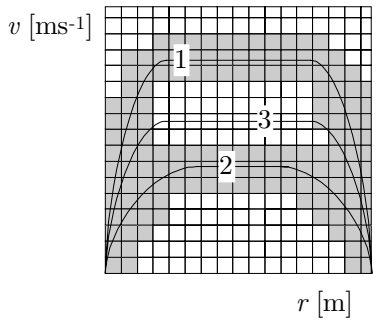



*Figure 4.3: Generalising ability*

We may conclude that for practical applicability it is important to keep the problems associated with the curse of dimensionality to a minimum. In the following section, we present methods that achieve this.

## 4.3 Parsimonious LFFC

### 4.3.1 Relations with Neuro-Fuzzy Modelling

In neuro-fuzzy modelling [Wang, 1994; Jang and Sun, 1995] the problems associated with the curse of dimensionality are kept to a minimum by employing parsimonious modelling techniques. According to the principle of parsimony [Bossley, 1997]: "*the best models are obtained using the simplest possible, acceptable structures that contain the smallest number of parameters*". In general, parsimony can be obtained by either of the following strategies:

– *Minimise the number of B-splines on each input domain.* A large number of B-splines causes a large number of weights, large training sets and a poor generalising ability. It logically follows that for a certain BSN, these problems can be minimised by minimising the number of B-splines on each input domain.

– *Split up the high-dimensional BSN in several lower-dimensional BSNs.* The number of B-splines depends on the dimension of the BSN in an exponential way. Thus reducing the dimension of the BSN is a much more effective method to overcome the effect of the curse of than minimising the number of B-splines on each input domain. In which way one can split up the high-dimensional BSN can be seen by writing the target function in the ANalysis Of VAriance (ANOVA) representation. The ANalysis Of VAriance (ANOVA) representation of an $n$-dimensional function $f(\cdot)$ is given by:

$$\begin{aligned} y &= f(x_1, x_2, .., x_n) \\ &= f_0 + \sum_i f_i(x_i) + \sum_{i,j} f_{i,j}(x_i, x_j) + \cdots + f_{1,2,..,n}(x_1, x_2, .., x_n) \end{aligned} \tag{4.8}$$

where $f_i(\cdot)$, $f_{i,j}(\cdot)$, *etc.* represent univariate, bivariate, *etc. additive* components of $f(\cdot)$. Instead of using one BSN that learns $f(x_1, x_2, .., x_n)$, we can create a parsimonious network structure in which for each term in (4.8) a specific BSN exists. When the maximum input dimension of the BSNs in the parsimonious network structure is smaller than $n$, a reduction of the number of B-splines results. As an example, we take the following $f(x_1, x_2, x_3)$:

$$y = f(x_1, x_2, x_3) = f_1(x_1) + f_{1,3}(x_1, x_3) + f_{2,3}(x_2, x_3) \tag{4.9}$$

This function can either be approximated by 1 BSN or by a parsimonious network structure consisting of 3 BSNs, see figure 4.4.

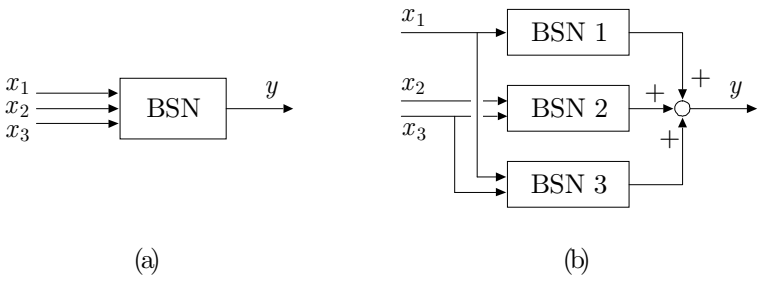



*Figure 4.4: a) Single BSN*
*b) ANOVA-based BSN structure*

The number of weights in the 3-dimensional BSN is (figure 4.4a):

$$N_{tot} = N_1 N_2 N_3 \tag{4.10}$$

where $N_i$ is the number of 1-dimensional B-spline functions defined on the input domain $i$. Assuming that in the parsimonious network structure the number of B-splines on each input domain is the same as in the 3-dimensional BSN (figure 4.4a), the total number of weights results in:

$$N_{tot} = N_1 + N_2 N_3 + N_1 N_3 \tag{4.11}$$

Especially for large $N_i$ this means a drastic reduction of the number of B-splines.

### 4.3.2 ANOVA based LFFC

It is straightforward that the main idea of parsimony in neuro-fuzzy modelling can also be applied to LFFC [Idema, 1996]. The problems associated with the curse of dimensionality can be kept to a minimum, when the LFFC consisting of one BSN is replaced by an LFFC that consist of several BSNs that each have a low input dimension. Figure 4.5a,b presents an example.

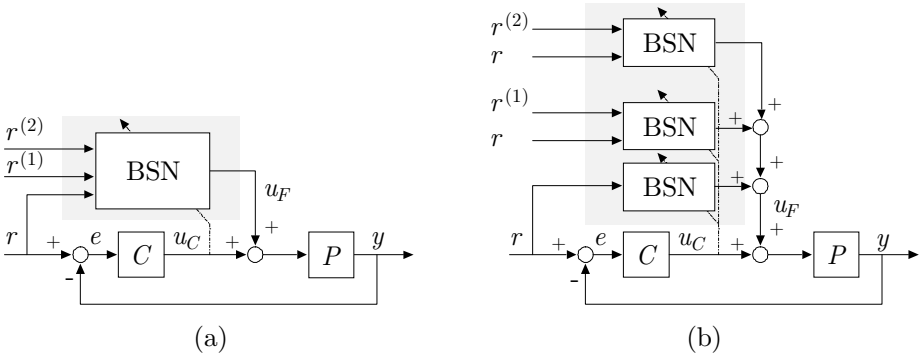



*Figure 4.5: a) Single LFFC*
*b) Parsimonious LFFC structure*

The parsimonious network structure follows directly from the desired feed-forward signal. One BSN should be created for each combination of $\{\boldsymbol{y}_d, \boldsymbol{y}_d, \boldsymbol{y}_d, \ldots\}$ in (3.14). The minimum number of B-splines depends on the shape of the term that each BSN is intended to learn.

The parsimonious network structure can be obtained in the three ways:

- *Manually*. In this case the designer has a-priori knowledge of the desired feed-forward signal, (chapter 3), such that a parsimonious network structure can be created.
- *Automated*. When insufficient a-priori knowledge exists, the parsimonious network structure can be created by means of empirical modelling techniques, using data gathered in experiments.
- *Combination*. In this case the designer first chooses a parsimonious network structure based on a-priori knowledge. Subsequently, empirical modelling techniques will be used to optimise the parsimonious network structure.

First, we will discuss the manual way.

### 4.3.3 Parsimony by Means of Prior Knowledge

In practice, the a-priori plant knowledge may include one or more of the following items:

- *Structure of* $\boldsymbol{A}(\boldsymbol{x}), \boldsymbol{B}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x})$. Often, the designer has knowledge of the dynamic behaviour of the plant. It is then possible to express the dynamics in the form presented in chapter 3, and herewith calculate the desired feed-forward signal, $\boldsymbol{u}_d$. The terms in $\boldsymbol{u}_d$ can be divided such that each of them contains one specific combination of $\left\{\boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \ddot{\boldsymbol{y}}_d, \ldots\right\}$. For each of these terms, one BSN can be created. E.g. in case of the robot manipulator, we know that the system behaves as a rigid robot manipulator. Furthermore, we know that it is subject to viscous and Coulomb friction. From these it follows that the dynamics are given by (4.1) and that $\boldsymbol{u}_d$ is given by (4.6)
- *Relevance of terms*. Besides the occurrence of certain combinations of $\left\{\boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \ddot{\boldsymbol{y}}_d, \ldots\right\}$ in the desired feed-forward signal, the designer often has knowledge about the magnitude and the shape of the terms. The

parsimonious network structure can be simplified by deleting all BSNs that learn terms that hardly contribute to the feed-forward signal. For the 2 DOF robot manipulator, $\boldsymbol{C}$ may be disregarded when the velocities are low and the load is relative small. Furthermore, because the gear ratio, $r_{1,2}$, is small $\boldsymbol{M}$ can often be considered diagonal and constant.

$$\boldsymbol{M} \approx \begin{bmatrix} \frac{J_1}{r_1^2} & 0 \\ 0 & \frac{J_2}{r_2^2} \end{bmatrix} \tag{4.12}$$

Substituting (4.12) in (4.6) and disregarding $\boldsymbol{C}$ yields the following desired feed-forward signal:

$$\begin{aligned} \boldsymbol{u}_d &= \begin{bmatrix} u_{d,1} \\ u_{d,2} \end{bmatrix} \\ &\approx \begin{bmatrix} \frac{J_1}{r_1^2}\ddot{\theta}_{d,1} + d_1\dot{\theta}_{d,1} + s_1 \operatorname{sgn}\left(\dot{\theta}_{d,1}\right) + gm_2l_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) + gl_1\cos\left(\theta_{d,1}\right)\left(m_1+m_2\right) \\ \frac{J_2}{r_2^2}\ddot{\theta}_{d,2} + d_2\dot{\theta}_{d,2} + s_2 \operatorname{sgn}\left(\dot{\theta}_{d,2}\right) + gl_2m_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) \end{bmatrix} \end{aligned} \tag{4.13}$$

The parsimonious network structure that follows from (4.13) is presented in figure 4.6. It consists of the following BNSs:

BSN 1 has 2 inputs, $\left\{\theta_{d,1}, \theta_{d,2}\right\}$ and two outputs, $\left\{u_{d,1}, u_{d,2}\right\}$. The target function it has to learn is:

$$\text{BSN 1: } \boldsymbol{u}_d = \begin{bmatrix} u_{d,1} \\ u_{d,2} \end{bmatrix} = \begin{bmatrix} gm_2l_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) + gl_1\cos\left(\theta_{d,1}\right)\left(m_1+m_2\right) \\ gl_2m_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) \end{bmatrix} \tag{4.14}$$

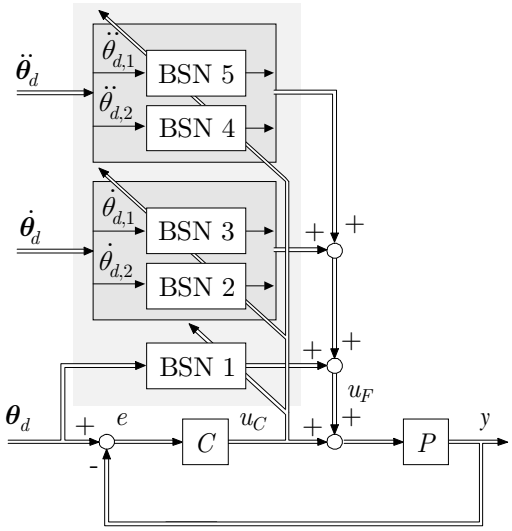



*Figure 4.6: Parsimonious learning feed-forward controller for a rigid robot*

BSN 2 and 3 each learn to compensate the friction phenomenons of one joint:

$$\text{BSN2: } u_{d,2} = d_2\dot{\theta}_{d,2} + s_2 \operatorname{sgn}\left(\dot{\theta}_{d,2}\right) \tag{4.15}$$

$$\text{BSN3: } u_{d,1} = d_1\dot{\theta}_{d,1} + s_1 \operatorname{sgn}\left(\dot{\theta}_{d,1}\right) \tag{4.16}$$

Finally, BSN 4 and 5 compensate for the inertia of the robot manipulator.

$$\text{BSN4: } u_{d,2} = \frac{J_2}{r_2^2}\ddot{\theta}_{d,2} \tag{4.17}$$

$$\text{BSN5: } u_{d,1} = \frac{J_1}{r_1^2}\ddot{\theta}_{d,1} \tag{4.18}$$

– *Shape of the terms.* In order to approximate a function using a minimum number of B-splines, the shape of the target function must be known. In case of the robot manipulator, we know that the target functions of BSN 4 and 5 are smooth, which means that the B-spline distributions of these BSNs may consist of B-splines that have a large support. It has no use approximating a smooth function with a large number of B-splines that have a narrow support.

Sometimes, the qualitative knowledge of the system is insufficient to create a parsimonious LFFC. In this situation, iterative application of empirical modelling techniques may offer a solution.

### 4.3.4 Parsimony by Iterative Empirical Modelling

Empirical modelling techniques are able to construct a model of a plant, on the basis of observed input-output data, without utilising any prior knowledge of the dynamics of the plant [Bossley, 1997]. For LFFC, we would like the empirical modelling techniques to create a parsimonious model of the inverse plant dynamics. However, we do not have appropriate training data of the function we want to approximate, i.e. the relation between the reference motion, consisting of $\{\boldsymbol{r}, \dot{\boldsymbol{r}}, ...\}$ and the desired feed-forward signal. Instead, we have the output of the feedback controller, $\boldsymbol{u}_C$, that indicates how the feed-forward signal, $\boldsymbol{u}_F$, should be adapted such that it resembles $\boldsymbol{u}_d$ better. This means that the parsimonious network structure could be obtained in the following iterative way:

**Algorithm 4.1 (Parsimony by means of iterative empirical modelling)**

1. *Choose an initial network structure and B-spline distribution.* Any prior knowledge of the plant that is available can be used to create an initial network structure and B-spline distribution (i.e., the manual method). If this is not the case, the empirical modelling technique in step 5 will select it.
2. *Select a training motion and train the LFFC until convergence.* Convergence is noticeable from the fact that the weight adaptation is zero. Note that after learning has converged $u_C$ is not equal to zero.
3. *To collect the data needed in step 5,* $\left\{\boldsymbol{r}, \dot{\boldsymbol{r}}, ..., \boldsymbol{u}_F, \boldsymbol{u}_C\right\}$, *the training motion is performed once using the fully trained LFFC.*
4. *If the MSE tracking error of the motion performed in step 3 has decreased as compared with previous iterations go to 5 else go to 7.*
5. *Build a new parsimonious network structure based on the collected data.*
6. *Go to step 2.*
7. *Stop.*

Building a parsimonious network structure on the basis of the collected data (step 5) is done in the following way. Because learning has converged, $\boldsymbol{u}_F$ is the best possible feed-forward signal, given the present parsimonious network structure. Since $\boldsymbol{u}_C$ indicates the error in $\boldsymbol{u}_F$, we can conclude that $\boldsymbol{u}_F + \boldsymbol{u}_C$ resembles the desired feed-forward signal, $\boldsymbol{u}_d$, better than the present one, $\boldsymbol{u}_F$, and will thus yield a smaller tracking error. Therefore, we now employ empirical modelling techniques to create a parsimonious network structure that is able to approximate $\boldsymbol{u}_F + \boldsymbol{u}_C$. This is done by setting $\{\boldsymbol{r}, \dot{\boldsymbol{r}}, ...\}$ as the inputs and $\boldsymbol{u}_F + \boldsymbol{u}_C$ as the target function for the empirical modeller. Note that the new parsimonious network structure is only able to accurately approximate $\boldsymbol{u}_F + \boldsymbol{u}_C$, which is in general unequal to $\boldsymbol{u}_d$. After training the LFFC with the new parsimonious network structure, a $\boldsymbol{u}_C$ will remain that indicates how the feed-forward signal (and therefore the parsimonious network structure) can be further improved. The empirical modelling technique should thus be applied in an iterative way. In the following, we will briefly discuss two examples of implementing step 5 using an empirical modelling technique.

### 4.3.5 Empirical Modelling: ASMOD

The Adaptive Spline Modelling of Observational Data (ASMOD) algorithm [Kavli, 1992; Kavli, 1993; Kavli and Weyer, 1995] creates a parsimonious BSN model by means of an iterative procedure that utilises refinement and pruning (simplification) methods. The outline of the iterative procedure is given below.

**Algorithm 4.2 (ASMOD modelling)**

5.1 *Generate candidate models.* In this step, a set of candidate models is generated by refining and pruning the existing parsimonious BSN model. The existing parsimonious BSN model can either be the initial model, chosen in step 1, or a model that has been obtained by previous application of the ASMOD algorithm. Possible steps for refining and pruning will be discussed in more detail below.

5.2 *Train and evaluate all candidates.* The most common way to evaluate the candidate models is by means of validation. Firstly, the collected data,

$\left\{\boldsymbol{r}, \dot{\boldsymbol{r}}, ..., \boldsymbol{u}_F + \boldsymbol{u}_C\right\}$, is split in a training set and an evaluation set. Next, the candidate models are trained with the training set. Their performance is evaluated by presenting the input values of the evaluation set, $\{\boldsymbol{r}, \dot{\boldsymbol{r}}, ...\}$, to the candidate models and comparing their outputs to the desired output, $\boldsymbol{u}_F + \boldsymbol{u}_C$. The best performing model is the model that has the lowest MSE approximation error.

5.3 *If the MSE approximation error for the evaluation set of the best performing model is lower than that of the current model, then select this model as the current model and go to 5.1 else go to 5.4.*

5.4 *Stop.*

The refinement and pruning methods that ASMOD may apply in step 5.1 are:

– *Adding / deleting a 1-dimensional BSN.* Each iteration step, one 1-dimensional BSN can either be added to the parsimonious network structure, or be removed. In figure 4.7 an example is presented, in which a 1-dimensional BSN with input $x_1$ is added / removed.

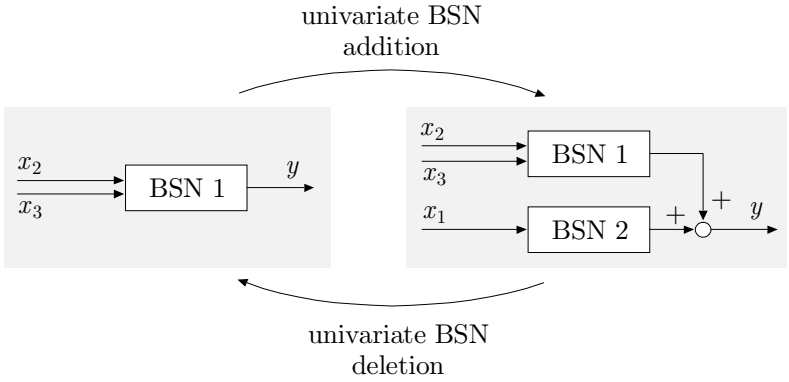



*Figure 4.7: Addition / deletion of a 1-dimensional BSN*

– *Splitting / combining BSNs.* To enable the model to approximate a larger class of functions, ASMOD can combine BSNs. In figure 4.8, two BSNs that have respectively $x_1$ and $x_4$ as inputs, are combined into one BSN that has both $x_1$ and $x_4$ as inputs. While the separate BSNs can approximate functions of the class $f\left(x_1\right) + g\left(x_4\right)$, the single BSN can approximate functions of the

class $f\left(x_1, x_4\right)$, which is a larger class. For reasons of parsimony, ASMOD can also split up a single BSN into two separate BSNs.

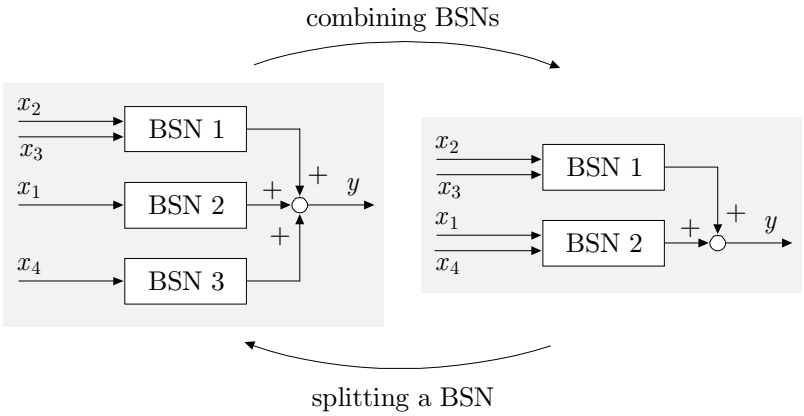



*Figure 4.8: Splitting / combining BSNs*

– *Knot insertion / removal.* The accuracy of an approximation depends on the size of the support and the location of the B-splines. ASMOD searches for the optimal B-spline distribution by adding a new B-spline halfway between two existing B-splines or by deleting B-splines from a distribution. In figure 4.9 an extra B-spline is added on the domain of input variable $x_1$.

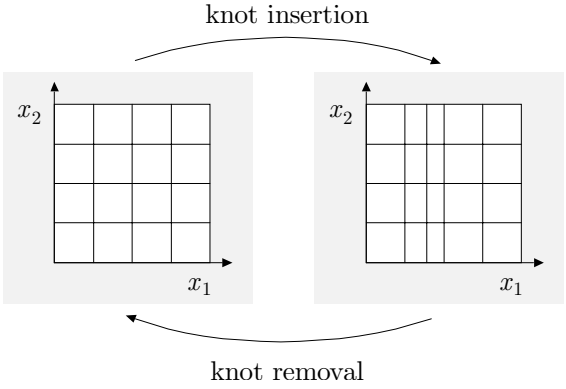



*Figure 4.9: Knot insertion / removal*

The ASMOD algorithm can apply several methods for refinement and pruning. For example, each B-spline in the model can be selected for combining. This large amount of possibilities makes the algorithm computationally expensive.

### 4.3.6 Empirical Modelling: Fuzzy Product Space Clustering

Fuzzy product space clustering, also called *fuzzy clustering*, is an empirical modelling technique that can be used to construct the B-spline distribution of the LFFC [Jang *et al.*, 1997; Babuška, 1997,b]. In short, fuzzy clustering [Bezdek and Pal, 1992] [Backer, 1995] generates fuzzy sets [Zadeh, 1973; Lee, 1990] (appendix A) that describe observed data. The fuzzy sets are generated in such way, that similar data points belong to the same fuzzy set. For example, consider the 2-dimensional data set shown in figure 4.10.

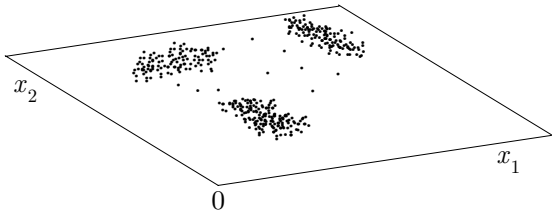



*Figure 4.10: Data for clustering*

Applying fuzzy clustering would yield the following fuzzy sets, $F_1$, $F_2$ and $F_3$, in which the data is categorised (figure 4.11).

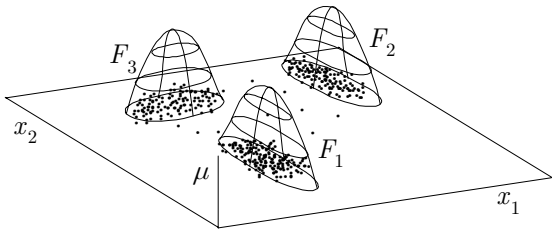



*Figure 4.11: Data and fuzzy sets*

Many fuzzy clustering algorithms exist. The fuzzy clustering algorithm that we use to obtain the clusters is the Gustafson-Kessel (GK) algorithm [Gustafson and Kessel, 1979]. This algorithm is chosen because it can create fuzzy sets of different shapes and orientations.

The data for fuzzy clustering consists of $\left\{r, \dot{r}, ..., u_F + u_C\right\}$. We will now explain how a B-spline distribution can be created by generating fuzzy sets that categorise this data. For reasons of clarity, we consider an LFFC that only has the reference position, $r$, as an input. The data is shown in figure 4.12.

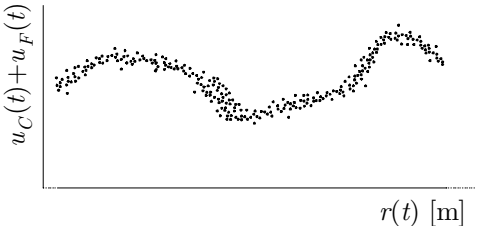



*Figure 4.12: Data for clustering algorithm in case of LFFC*

Next, the GK-algorithm is applied to construct fuzzy sets that describe the data. The fuzzy sets that have been obtained are represented by their level curves (figure 4.13).

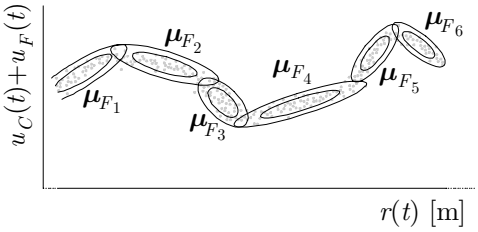



*Figure 4.13: Data and fuzzy sets in case of LFFC*

We have now obtained fuzzy sets that describe similar data points in the data set $\left\{r, u_F + u_C\right\}$. In order to give insight in how B-splines can be formed from the fuzzy sets, we first build a Takagi-Sugeno (TS) fuzzy model [Takagi and Sugeno, 1985](appendix A) of the data. Subsequently, from the TS fuzzy model, a B-spline distribution is generated [Babuška, 1997,b]. A TS fuzzy model consists of a set of rules that have the following form:

$$\text{Rule } i\text{:} \quad \begin{array}{l} \text{IF } \left(\boldsymbol{r} \text{ is } A_{1,i}\right) \text{AND} \cdots \text{AND } \left(\boldsymbol{r}^{(n)} \text{ is } A_{n,i}\right) \\ \text{THEN } \boldsymbol{u}_F = a_{1,i}\boldsymbol{r} + \cdots + a_{n,i}\boldsymbol{r}^{(n)} + b_i \end{array} \tag{4.19}$$

Where, $A_{1,i} \ldots A_{n,i}$ are the fuzzy sets defined on the inputs of the model, also known as fuzzy antecedent sets. Both the fuzzy antecedent sets and the coefficients $a_{1,i}$,…, $a_{n,i}$ and $b_{1,i}$ can be derived from the fuzzy sets obtained by fuzzy clustering. The fuzzy antecedent sets $A_i$ can be obtained by projecting the fuzzy sets, $F_1$, …, $F_6$, on the input axis, $r$ (figure 4.14).

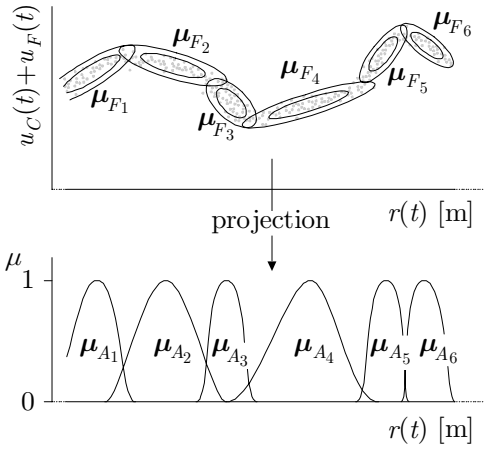



*Figure 4.14: Projection of the fuzzy sets*

The parameters $a_{1,i}$, ..., $a_{n,i}$ and $b_{1,i}$ can be derived from the position and the orientation of the clusters [Babuška, 1997,b]. In figure 4.15, the data of figure 4.12 is approximated by a TS fuzzy model, using the fuzzy antecedent sets obtained in figure 4.14. It can be seen that the approximation consists of locally-linear approximations with smooth transitions.

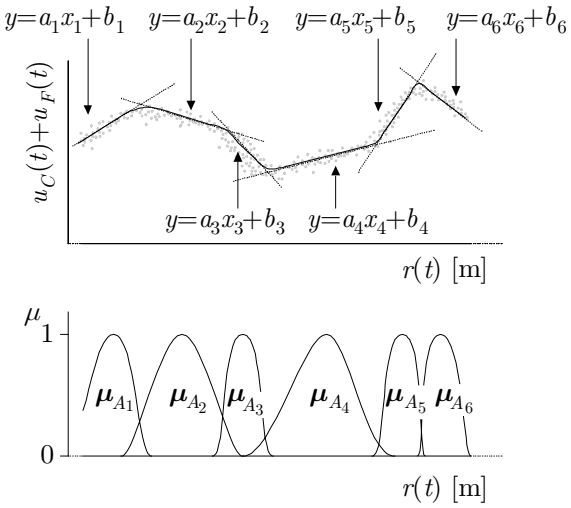



*Figure 4.15: TS-model approximation of the data*

Next, we use the fuzzy antecedent sets of the TS fuzzy model to design a B-spline distribution. To explain how this can be done, in figure 4.16 a BSN approximation and a TS fuzzy model approximation of the data of figure 4.12 are shown.

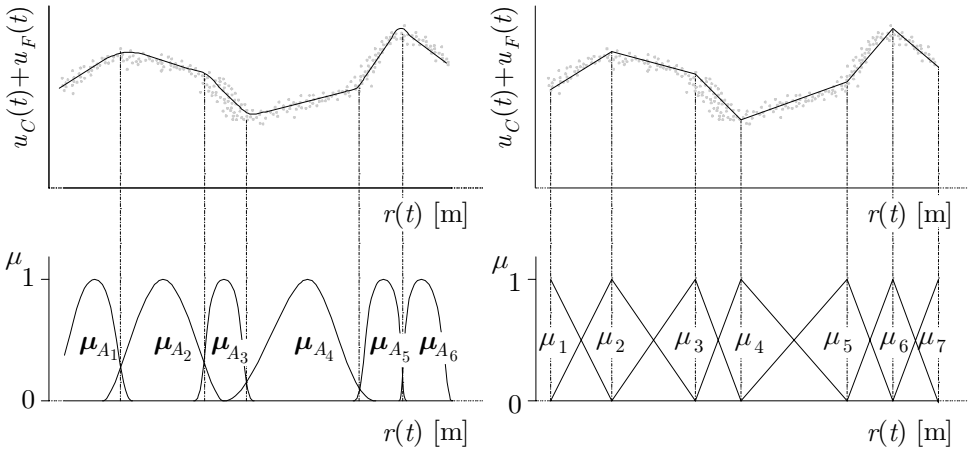



*Figure 4.16: Equivalence between a TS-fuzzy model and a BSN*

It can be seen that both approximations consist of a number of locally-linear approximations. In case of the TS fuzzy model, the transitions between locally-linear approximations is smooth. Switching from one locally-linear approximation to another occurs at those points where the fuzzy antecedent sets have the same membership. In case of the BSN, the transition between locally-linear approximations is sharp. The transition occurs when a B-spline reaches its maximum membership. Note that this point corresponds to the point where one B-spline ends and the other begins. B-splines can thus be derived from the fuzzy antecedent sets, by locating the B-spline knots at to those points where the fuzzy antecedent sets have equal memberships (figure 4.17).

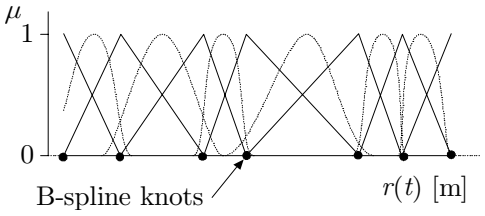



*Figure 4.17: B-spline knots and fuzzy sets*

The previous can be summarised in the following algorithm.

**Algorithm 4.3 (B-spline distribution by means of fuzzy clustering)**

5.1 *Employ GK fuzzy clustering [Gustafson and Kessel, 1979; Babuška, 1997,b] using the data set* $\left\{r,\dot{r},...,u_F+u_C\right\}$.

5.2 *Construct a TS fuzzy model that describes the relation between* $\{r, \dot{r}, ...\}$ *and* $u_F + u_C$, *on the basis of the fuzzy sets obtained in the previous step.* The fuzzy antecedent sets of the TS fuzzy model are created by projecting the fuzzy sets on the input domain $\{r, \dot{r}, ...\}$. The coefficients of the locally-linear functions that describe the output follow from the position and the orientation of the fuzzy sets.

5.3 *Transform the fuzzy antecedent sets of the TS fuzzy model into B-splines.* The modelling capabilities of a TS fuzzy model and a BSN are equivalent when the B-spline knots are situated that the intersections of the fuzzy antecedent sets.

## 4.4 Training the Parsimonious LFFC

Unlike training of a standard LFFC, training a parsimonious LFFC is not a trivial task. This is caused by the fact that only one common training signal is available for all BSNs in the parsimonious LFFC. For the parsimonious LFFC for the 2 DOF robot manipulator, represented in figure 4.18, this means that one learning signal exists for 5 BSNs. This learning signal indicates how the total input-output relation (from $\left\{\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d\right\}$ to $\boldsymbol{u}_F$) has to be adapted and not how the separate network input-output relations (from $\boldsymbol{\theta}_d$ to $\boldsymbol{u}_{F_1}$, $\dot{\boldsymbol{\theta}}_d$ to $\boldsymbol{u}_{F_2}$ and $\ddot{\boldsymbol{\theta}}_d$ to $\boldsymbol{u}_{F_3}$) should change. It may thus occur that, even though the total feed-forward signal is correctly adapted, the weights of the separate BSNs are adapted in the wrong way. The only way the designer can influence the training of the BSNs is by selecting the reference motions and the learning rates. In the following, we will examine in what way the training motions and the learning rates should be chosen such that the BSNs are correctly trained on the basis of one learning signal.

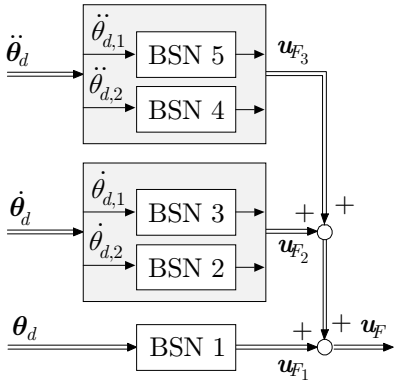



*Figure 4.18: Parsimonious LFFC for a robot manipulator*

Consider, for example, training BSN 1 in figure 4.18, which has to compensate for gravitational influences and has the reference joint angles as inputs. The learning signal $\boldsymbol{u}_C$, indicates how the feed forward signal, $\boldsymbol{u}_F$, must be adapted, such that it resembles the desired feed-forward signal of the entire LFFC (4.20):

$$\begin{bmatrix} \frac{J_1}{r_1^2}\ddot{\theta}_{d,1} + d_1\dot{\theta}_{d,1} + s_1 \operatorname{sgn}\left(\dot{\theta}_{d,1}\right) + gm_2l_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) + gl_1\cos\left(\theta_{d,1}\right)\left(m_1+m_2\right) \\ \frac{J_2}{r_2^2}\ddot{\theta}_{d,2} + d_2\dot{\theta}_{d,2} + s_2 \operatorname{sgn}\left(\dot{\theta}_{d,2}\right) + gl_2m_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) \end{bmatrix} \tag{4.20}$$

while the target function of BSN 1 is given by:

$$\begin{bmatrix} gm_2l_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) + gl_1\cos\left(\theta_{d,1}\right)\left(m_1+m_2\right) \\ gl_2m_2\cos\left(\theta_{d,1}+\theta_{d,2}\right) \end{bmatrix} \tag{4.21}$$

When only a few training motions would be performed, there is no way of guaranteeing that BSN 1 would not learn part of the desired feed-forward signals of the other BSNs. In other words, BSN 1 would learn to compensate the friction forces and the inertia of the manipulator for these particular motions. However, the inputs of BSN 1 consist of $\boldsymbol{\theta}_d$, while the friction forces depend on $\dot{\boldsymbol{\theta}}_d$. This means

that the LFFC is not able to compensate the friction forces for other motions than the training motions. In short, the LFFC has a poor generalising ability.

To correctly train BSN 1 using a particular set of motions, the terms of (4.20) that are not in (4.21), should cancel out. This can be formalised as follows. Consider a set of $N$ training motions, $\boldsymbol{\theta}_d^i$, $i = \{1,...,N\}$. To train BSN 1 at the input values $\boldsymbol{\theta}_d = \begin{bmatrix} \theta_{d,1} & \theta_{d,2} \end{bmatrix}^T$ the following should be satisfied:

$$\begin{aligned} \sum_{i=1}^{N} \left( \frac{J_1}{r_1^2} \ddot{\theta}_{d,1}^i \left( t_{i,1} \right) + d_1 \dot{\theta}_{d,1}^i \left( t_{i,1} \right) + s_1 \operatorname{sgn} \left( \dot{\theta}_{d,1}^i \left( t_{i,1} \right) \right) \right) = 0 \\ \sum_{i=1}^{N} \left( \frac{J_2}{r_2^2} \ddot{\theta}_{d,2}^i \left( t_{i,2} \right) + d_2 \dot{\theta}_{d,2}^i \left( t_{i,2} \right) + s_2 \operatorname{sgn} \left( \dot{\theta}_{d,2}^i \left( t_{i,2} \right) \right) \right) = 0 \end{aligned} \tag{4.22}$$

where, $t_{i,1}$ is the point in time where $\theta_{d,1}^i = \theta_{d,1}$ and $t_{i,2}$ the points in time where $\theta_{d,2}^i = \theta_{d,2}$:

$$\begin{aligned} t_{i,1} &= \arg \left( \theta_{d,1}^i (t) = \theta_{d,1} \right) \\ t_{i,2} &= \arg \left( \theta_{d,2}^i (t) = \theta_{d,2} \right) \end{aligned} \tag{4.23}$$

We will refer to a set of training motions that satisfies the above as a sufficiently rich set of training motions. One way to obtain a sufficiently rich set of training motions is to perform a large amount of different reference motions. For the above example, the same reference angles will be realised at a range of different reference angular velocities and reference angular accelerations. However, a large number of training motions results in an extensive training period, which is undesired. Furthermore, some of these reference motions may be hard to realise due to hardware limitations. Think, for example, of large reference velocities near hardware end-stops.

We propose to generate a sufficiently rich set of training motions in another way. A training motion should be chosen such that the desired output of one specific BSN is temporarily much larger than the desired outputs of the other BSNs. Since the desired outputs of the other BSNs can be disregarded, this training motion is sufficiently rich with respect to one BSN. In this situation, the desired feed-forward signal of the entire LFFC is equal to the desired feed-forward signal of one of the BSNs in the LFFC. The learning signal indicates how the weights of a separate

BSN must be adapted. Instead of training all BSNs simultaneously, only the weights of the BSN of which the desired output is dominant are adapted. The weights of the other BSNs are kept constant. To train all BSNs, we need to select a series of training motions which each make the desired output of one specific BSN dominant. Note that when choosing a training motion, we only need to consider the untrained BSNs in the LFFC. We assume that the trained BSNs have learned their desired input-output mapping, such that the learning signal only contains information on how to adapt the weights of the untrained BSNs.

**Proposition 4.1 (Strategic training procedure)** *In a parsimonious LFFC, one BSN should be trained at a time. The training reference motions should be chosen such, that the desired output of one of the untrained BSNs is temporarily dominant. Only the weights of this BSN are adapted during this period, the weights of the other BSNs are kept constant. This yields the following training procedure:*

1. *Choose a training motion, such that the target signal of one of the untrained BSNs in the parsimonious LFFC is dominant.*
2. *Train the corresponding BSN until its weights have converged. During training, the previously trained BSNs are used for control.*
3. *If there are any untrained BSNs left, go to 1, else, go to 4.*
4. *Stop.*

For the robot manipulator, the strategic training procedure will look as follows [Steenkuijl, 1999]:

**Algorithm 4.5 (Strategic learning procedure for a robot manipulator)**

1. The first BSN that is trained is the BSN that has to compensate the gravitation force, see figure 4.6. By choosing a low-velocity low-acceleration reference motion, the effects of the inertia of the manipulator and the friction in the joints can be made small. Since the desired feed-forward signal (4.21) contains the term $\left(\theta_{d,1}+\theta_{d,2}\right)$, the reference motion should contain various combinations of $\left\{\theta_{d,1},\theta_{d,2}\right\}$. After training the weights of this BSN are fixed.
2. Next, the BSN is trained that compensates for the inertia of the manipulator. The reference motions should be chosen such that the friction forces are kept low. This means that the reference motion will consists of a

rapid sequence of short accelerations and decelerations. After training, the weights of this BSN are also fixed.

3. Finally, we train the BSN that compensates for the friction forces. Since the other BSNs have already been trained, no specific reference motion is necessary. However, the reference motion should span all velocities that will occur during normal operation.

## 4.5 Discussion

When using the straightforward way to determine the inputs of the LFFC, as presented in chapter 3, a multidimensional BSN will often result. This confronts us with the problems associated with the curse of dimensionality. As a solution, we proposed to split up the single high-dimensional BSN into a number of low-dimensional BSNs, each compensating for one specific disturbance. The result is known as a parsimonious network structure. Splitting up the single BSN can be performed in two ways:

- Firstly, we make use of prior qualitative knowledge of the plant dynamics and the reproducible disturbances to split up the BSN.
- Secondly, when prior knowledge is unavailable or insufficient, we can employ empirical modelling techniques to obtain the parsimonious network structure. Two different techniques have been discussed, ASMOD and fuzzy clustering. Both techniques are computationally intensive. At the time of this research, generating a parsimonious network structure from a highly non-linear, multi-dimensional data set took excessive computer time. This makes these solutions inapplicable in practice, until the computer power has increased drastically. To show that, in principal, these techniques can create a parsimonious network structure, in the next chapter, fuzzy clustering is used to optimise the B-spline distribution in a time-indexed LFFC.

Training the BSNs in a parsimonious LFFC is not a trivial task. When all BSNs are trained simultaneously, a large number of training reference motions is usually needed. Otherwise, BSNs may learn part of desired feed-forward signals of other BSNs. To train a parsimonious LFFC with a small number of training reference motions, we proposed a strategic learning procedure. Instead of training all BSNs

simultaneously, each BSN is trained individually. The training reference motion is chosen such that the desired feed-forward signal of this BSN is much larger than the desired feed-forward signals of the other untrained BSNs.

Sometimes it may be difficult or even impossible to design a set of training motions suitable for strategic training. Consider, for example, the case where the Coriolis and centrifugal effects in the robot manipulator cannot be disregarded. To compensate for these effects, two BSNs should be added to the parsimonious LFFC that have respectively $\left\{\dot{\theta}_1, \theta_2, \dot{\theta}_2\right\}$ and $\left\{\dot{\theta}_1, \theta_2\right\}$ as inputs. The desired feed-forward signals are:

$$\text{BSN 6: } u_d = -2l_1 l_2 m_2 \sin\left(\theta_2\right)\dot{\theta}_2\dot{\theta}_1 - l_1 l_2 m_2 \sin\left(\theta_2\right)\dot{\theta}_2\dot{\theta}_2 \tag{4.24}$$

$$\text{BSN 7: } u_d = l_1 l_2 m_2 \sin\left(\theta_2\right)\dot{\theta}_1\dot{\theta}_1 \tag{4.25}$$

The desired feed-forward signal of BSN 6 makes proper training of BSN 2, which has to compensate for the friction in the second joint, difficult. The desired feed-forward signal of BSN 2 is $u_{d,2} = d_2\dot{\theta}_{d,2} + s_2 \operatorname{sgn}\left(\dot{\theta}_{d,2}\right)$. To properly train BSN 2, a wide range of velocities should be realised. However, in this case the term $-l_1 l_2 m_2 \sin\left(\theta_2\right)\dot{\theta}_2\dot{\theta}_2$ cannot be kept small. The solution in this kind of situations is to combine the two separate BSNs in one BSN. In the above case, this means that the compensation of the friction, $u_{d,2} = d_2\dot{\theta}_{d,2} + s_2 \operatorname{sgn}\left(\dot{\theta}_{d,2}\right)$, is stored in a BSN that has $\left\{\dot{\theta}_1, \theta_2, \dot{\theta}_2\right\}$ as inputs. Consequently, the network structure is less parsimonious and more training samples are needed to compensate for the friction, but at least the BSN can be trained properly.

The strategic training procedure remains rather heuristic. More research is needed on the following topics:

- *Existence of a strategic training procedure given a parsimonious LFFC.* As discussed above, it may not always be possible to select a set of training motions that make only one specific desired feed-forward signal dominant. It

would be desirable to have a systematic procedure that indicates whether a set of training motions exists that is sufficiently rich.

- *Systematic procedure that gives sufficiently rich training motions.* The way in which the training motions are selected is ad-hoc. This may result in designing a set of strategic training motions by means of trial and error. To prevent this, a systematic procedure is needed, that derives a set of sufficiently rich training motions, on the basis of qualitative knowledge of the plant dynamics and the disturbances.
- *Other types of training motions that are sufficiently rich.* We proposed to design a set of sufficiently rich training motions, by making the desired feed-forward signal of one of the untrained BSNs dominant. Other training motions may exist, that are also sufficiently rich. One could think, for example, of adding noise to the reference motion.

# 5 Design & Applications

## 5.1 Introduction

In the previous chapters, several aspects of LFFC have been dealt with. The obtained knowledge and insight will now be used to formulate a practical design procedure. In short, the following design choices have to be made:

1. *Feedback controller.* The feedback controller compensates for random disturbances and generates a learning signal for the feed-forward part. In chapter 2, it has been shown that the minimum width of the B-splines, and thus the maximum achievable accuracy, depends on the frequency response of the closed loop system. Since the frequency response of the closed loop system depends on the feedback controller, it indirectly determines the maximum achievable tracking performance. When the minimum width of the B-splines is too large to achieve an acceptable tracking error, redesigning the feedback controller offers a solution. However, this requires that the feedback controller has to be designed such that the bandwidth of the closed-loop system increases (see chapter 2) and it means that the stability robustness for variations in the plant dynamics decreases. We propose to solve this problem in an alternative way. The feedback controller is designed such that it features robust stability. If the resulting minimum width of the B-splines does not yield the desired tracking performance, a learning filter is added to the LFFC. If the learning filter is designed as proposed in chapter 3, the minimum allowable width of the B-splines decreases.
2. *Inputs of feed-forward part.* The inputs of the feed-forward part depend on the type of motion that has to be performed. In case of repetitive motions, the periodic motion time is preferred as input (see chapter 2). When

performing random motions, the inputs should consist of the reference position and eventually derivatives/integrals thereof. In chapter 3 it has been shown how the inputs can be selected on the basis of a state-space description of the plant.

3. *Structure of the feed-forward part.* In general, the result of the previous design choice is that the feed-forward part should have multiple inputs. When implementing the feed-forward part as one multi-dimensional BSN, we are confronted with the problems associated with the curse of dimensionality. This problem can be overcome by replacing the multi-dimensional BSN by a so-called parsimonious network structure, consisting of several BSNs that have a lower dimension. Chapter 4 shows that this can be done either on the basis of prior knowledge of the plant dynamics and the disturbances, or in an automated way by using empirical modelling techniques.

4. *B-spline distribution.* Chapter 2 clarified that a too small width of the B-splines causes learning to diverge. For the time-indexed LFFC, the minimum width of the B-splines, for which learning converges, can be determined on the basis of the frequency response of the closed loop system. In chapter 3, it was argued that in case of an LFFC, the width of the B-splines, as seen in time, should also be sufficiently wide, in order to guarantee that learning converges. In case of a multi-dimensional BSN it may be difficult to design a B-spline distribution that satisfies this. It was shown that regularisation may overcome this problem.

5. *Learning rate.* The learning rate determines how strong the weights of the BSN are adapted. In chapter 2, the maximum value of the learning rate, for which learning converges, has been determined by means of a frequency response of the closed loop system. The learning rate should be chosen small (close to 0), when the plant is subject to considerable noise. Otherwise, a large learning rate can be chosen.

6. *Training motions.* Training a time-indexed LFFC and an LFFC that comprises only one BSN can be performed in a straightforward way. Special care should be taken when training a parsimonious LFFC. Chapter 4 shows that simultaneously training all BSNs, generally results in BSNs that learn feed-forward signals other than the desired ones. To overcome this, the BSNs are trained one at a time. The reference motions should be chosen such that the desired feed-forward signal of one of the untrained BSNs becomes dominant. Only the corresponding BSN is trained, the weights of the other BSNs are kept constant.

In the following, design procedures for both a time-indexed LFFC and an LFFC will be dealt with in more detail. This will be done for two applications. The first application is the LiMMS. The second application is a part of a flight simulator.

## 5.2 LiMMS

### 5.2.1 Set-up

The LiMMS has been described in chapter 1. An approximate model, used in the simulations, is show in figure 5.1.

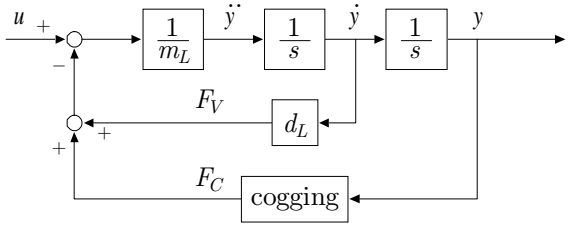



*Figure 5.1: Simulation model of the LiMMS*

The cogging was modelled as:

$$F_C(x) = 10\sin\left(1.6\,10^{-2}x\right) \tag{5.1}$$

Experiments showed that (5.1) is only a rough approximation of the actual cogging characteristic. This is caused by the fact that relatively low-cost magnets have been used with considerable magnetic tolerances and that the magnets are not placed with the highest possible accuracy. The cogging characteristic that results is a sine-shaped function of which both the period and the amplitude depend on the position of the translator LiMMS. Furthermore, the cogging characteristic showed to be dependent on the motion direction. In the simulation model, only viscous friction is considered. It is assumed that in the real set-up the friction characteristics can be described by the Stribeck curve. We conclude that the LFFC has to compensate for:

- The inertia of the LiMMS.
- Non-linear friction.
- The cogging forces.

### 5.2.2 Design Procedure for a Time-Indexed LFFC

In some applications the LiMMS has to perform repetitive motions. Therefore, we firstly consider a time-indexed LFFC. The design procedure for a time-indexed LFFC is given below.

**Step 1: Design the feedback controller**

In these experiments, the feedback controller was obtained by means of an auto-tuning mechanism present in the LiMMS set-up. The feedback controller is of the PD-type, which is put in series with a low-pass filter:

$$C(s) = (5538s + 275280)\frac{(400\pi)^2}{s^2 + 100\pi s + (400\pi)^2} \tag{5.2}$$

**Step 2: Determine the minimum width of the support of the B-splines and the maximum learning rate**

In order to determine the minimum width of the B-splines for which learning converges, a Bode plot of the negative complementary sensitivity function is required. This Bode plot of figure 5.2 has been obtained by means of experimental frequency analysis.

The peak in the amplitude at approximately 20 Hz is caused by a measurement error and should be disregarded. To obtain the minimum B-spline width, we need to determine (see chapter 2):

$$\varphi_1 = \arccos\left(-0.0147\frac{\left|-T(j\omega)\right|_\infty}{\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right|}\right) \tag{5.3}$$

Close inspection of figure 5.2 gives:

$$\left|-T(j\omega)\right|_\infty = 1.5\left(3.5\text{ dB}\right) \tag{5.4}$$

$$\min_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}}\left|-T(j\omega)\right| = 0.84\left(-1.5\text{ dB}\right) \tag{5.5}$$

Using (5.4) and (5.5), (5.3) yields:

$$\varphi_1 = \arccos(-0.0263) = -1.49\pi \tag{5.6}$$

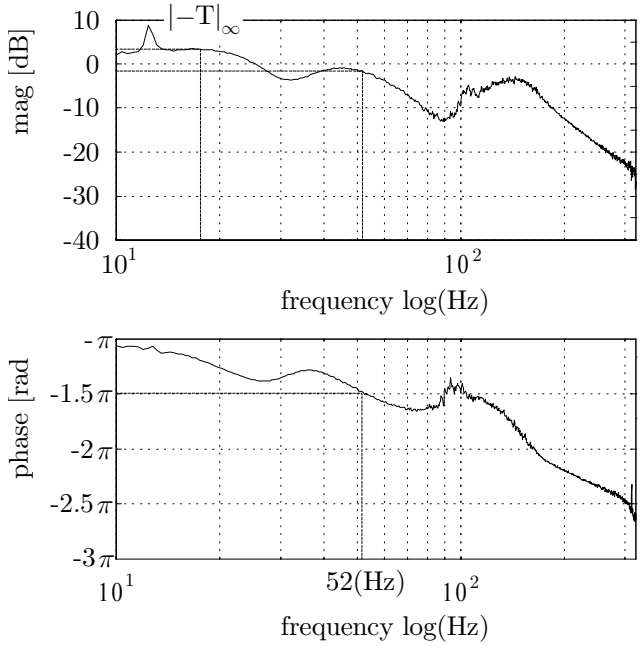



*Figure 5.2: Bode plot of $-T$*

The frequency at which the phase shift is equal to –1.49$\pi$ is:

$$\omega = 52 \cdot 2\pi \left[\mathrm{rad\,s^{-1}}\right] = 104\pi \left[\mathrm{rad\,s^{-1}}\right] \tag{5.7}$$

This gives the following minimum width of the support of the B-splines (see chapter 2):

$$d_{\min} = \frac{2\pi}{104\pi}\,[\mathrm{s}] = 0.0192\,[\mathrm{s}] \tag{5.8}$$

Next, the maximum learning rate is determined. In order for learning to converge, the learning rate should satisfy:

$$\gamma \leq \frac{2}{\left|-T(j\omega)\right|_\infty} \tag{5.9}$$

Using (5.4) this gives:

$$\gamma \leq \frac{2}{\left|-T(j\omega)\right|_\infty} = \frac{2}{1.5} = 1.33 \tag{5.10}$$

**Step 4: Choose the B-spline distribution**

The B-spline distribution should be chosen such that the width of the support of the B-splines is larger than $d_{min}$.

**Step 5: Select a learning rate**

As discussed before, $\gamma$ should be chosen small (close to 0), when the plant is subject to considerable noise. If this is not the case, a larger value of $\gamma$ is allowed. When $0 \leq \gamma \leq \frac{1}{\left|-T(j\omega)\right|_\infty}$, the tracking error will decrease gradually (see chapter 2), while $\frac{1}{\left|-T(j\omega)\right|_\infty} \leq \gamma \leq \frac{2}{\left|-T(j\omega)\right|_\infty}$ causes the tracking error to decrease in an oscillatory way. Since we consider the latter learning behaviour undesired, we recommend $\gamma \approx \frac{1}{\left|-T(j\omega)\right|_\infty} = 0.67$.

**Step 6: Train the time-indexed LFFC**

Training a time-indexed is straightforward.

### 5.2.3 Validation Experiments of the Time-Indexed LFFC

In the following, two series of experiments will be performed. The goal of the first series of experiments is to validate the minimum width of the support of the B-splines and the maximum learning rate. Next, the ability of fuzzy clustering to determine a B-spline distribution will be examined.

In order to validate the minimum width of the support of the B-spline and the maximum learning rate, these values will be determined by means of experiments. The reference position that the LiMMS has to track is given in figure 5.3.

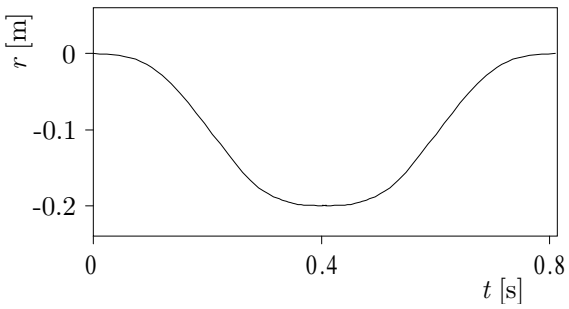



*Figure 5.3: Reference position*

**Experiment 5.1 (Width and learning rate conform the rules)**

In the first experiment, the support of the B-splines is chosen slightly larger than the minimum support, namely $d$=0.0197 [s]. In order to be sure that eventual occurrence of unstable behaviour is caused by a too small width of the support of the B-splines and not by a too large learning rate, the value of the learning rate is chosen significantly smaller than the maximum value, namely $\gamma$ =0.6. These values should yield a stable and well-performing LFFC. The tracking error before learning is given in figure 5.4.

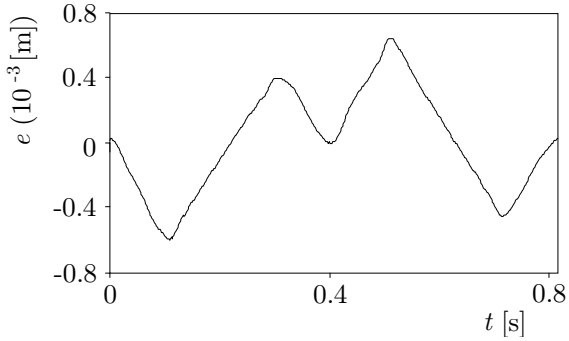



*Figure 5.4: Tracking error without learning*

Within 10 repetitions the time-indexed LFFC has converged. In order to examine whether the LFFC remains stable, learning is continued. In figure 5.5, the tracking error after 400 runs is presented. By learning the tracking error has decreased drastically. The controller remains stable and the learning mechanism has converged, which means that the tracking error does not decrease any further.

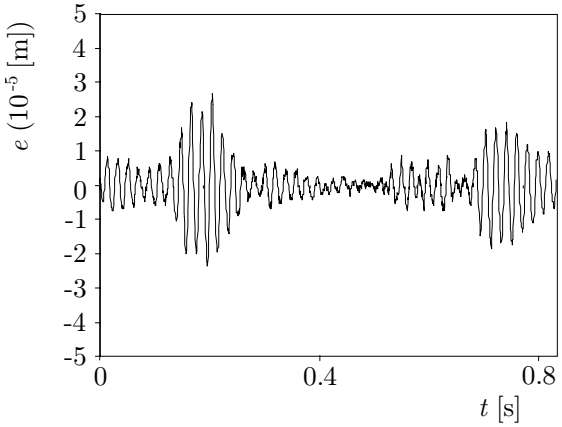



*Figure 5.5: Experiments with* $d > d_{\min}$ *, and* $\gamma_d < \gamma_{d,max}$

**Experiment 5.2 (Width not conform the rules)**

In the second experiment the support of the B-splines is chosen somewhat smaller than $d_{min}$, $d$=0.0183 [s]. Again, the learning rate is chosen $\gamma$=0.6. When these values are used the LFFC should become unstable. In figure 5.6 the tracking error after 200 runs is given. After the error had decreased in the first number of runs, it started to increase as learning continued.

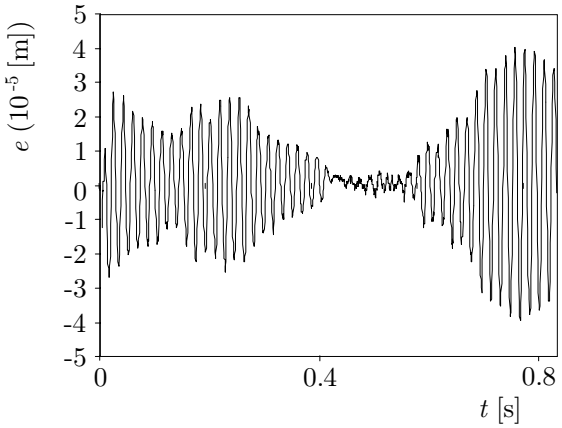



*Figure 5.6: Experiments with* $d < d_{\min}$ *and* $\gamma_d < \gamma_{d,max}$

Using this result and the result of experiment 5.1, it can be concluded that the value of $d_{min}$ is not conservative. When the support of the B-splines is chosen only slightly smaller than $d_{min}$ an unstable LFFC results.

**Experiment 5.3 (Determination of the maximum learning rate)**

Next, the maximum value of $\gamma$, as obtained in (5.10), is validated. This is done by searching for the maximum experimental value of $\gamma$ in an iterative way. The width of the support of the B-splines is chosen $d$=0.035 [s]. The result of these experiments is that the maximum value of $\gamma$ is equal to 2. This can be seen in the following two experiments. First, the learning rate is chosen $\gamma$ =1.98. In figure 5.7 the tracking error in the first, the 15$^{\text{th}}$ and the 50$^{\text{th}}$ run is given. The tracking error slowly converges and remains small as learning continues.

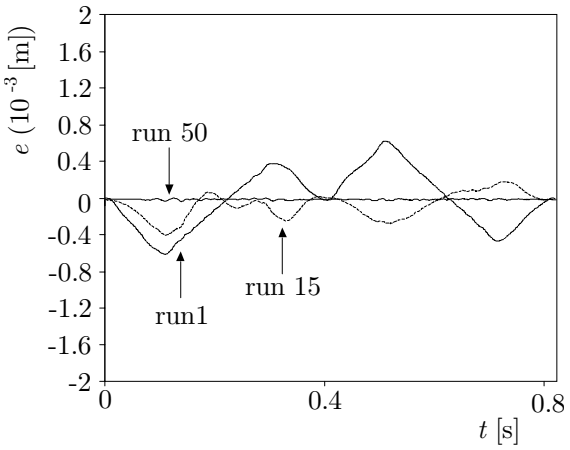



*Figure 5.7: Learning rate experiments, d> $d_{\text{min}}$ and $\gamma$ =1.98*

Next, the learning rate is chosen $\gamma$=2.01. Figure 5.8 shows the tracking error in the first and 6$^{\text{th}}$ run. It can be seen that the tracking error has increased.

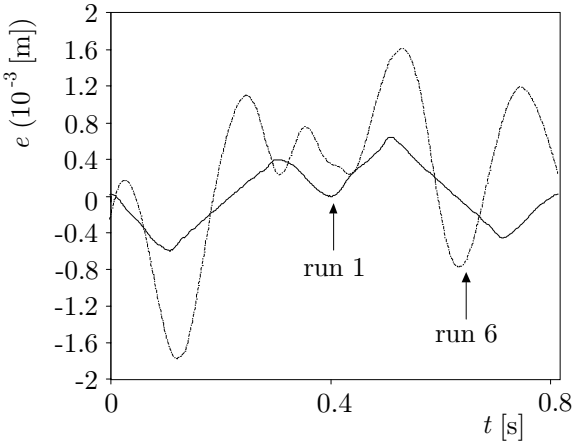



*Figure 5.8: Learning rate experiments, d> $d_{min}$ and $\gamma$ =2.01*

As learning continues, it will increase even further. The maximum value of $\gamma$ as determined in (5.10) is thus a somewhat conservative value.

Next, fuzzy clustering will be used to construct a B-spline distribution for a time-indexed LFFC. In chapter 4, it was proposed to use fuzzy clustering to obtain a B-spline distribution for an LFFC. However, such an LFFC involves large multi-dimensional data sets. The computational complexity of fuzzy clustering would cause excessive computation time. Therefore, we choose to demonstrate fuzzy-clustering with a time-indexed LFFC. The reference position that the LiMMS has to track repeatedly is shown in figure 5.9.

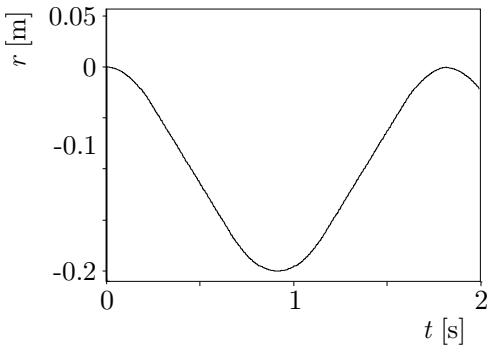



*Figure 5.9: Reference position for fuzzy clustering experiment*

**Experiment 5.4 (First step of the clustering algorithm)**

The basic idea of applying empirical modelling techniques in LFFC is the following. After learning has converged, a tracking error (and $u_C$) remains. Apparently, using the current B-spline distribution, the BSN is unable to approximate the desired feed-forward signal better. In other words, it is unable to approximate $u_F+u_C$. To overcome this, empirical modelling techniques are used to find a B-spline distribution that is better able to approximate $u_F+u_C$. Since at the start of the algorithm $u_F=0$, the initial B-spline distribution is thus based on $u_C$. The initial B-spline distribution can be either constructed by fuzzy clustering or by hand (after which fuzzy clustering optimises this B-spline distribution). The output of the feedback controller and the resulting tracking error are given in figure 5.10 and 5.11.

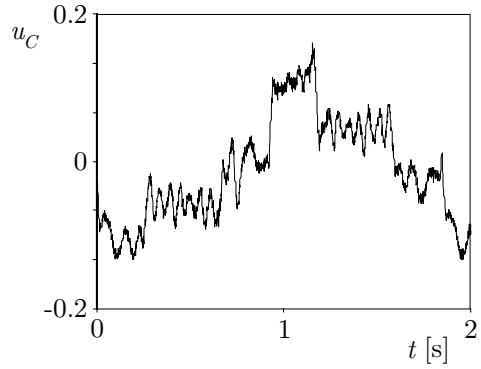



*Figure 5.10: Output of the feedback controller*

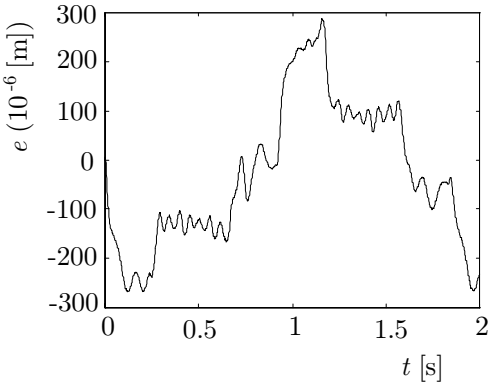



*Figure 5.11: Tracking error of the feedback controller*

A B-spline distribution is chosen that has a small number of B-splines. In this way, the positions of the B-splines strongly determine the ability of the LFFC to accurately compensate for disturbances such as cogging. In this research, we choose to select the initial B-spline distribution by hand, based on the shape of $u_C$ (figure 5.10). The BSN approximates $u_C$ by piecewise first-order polynomial functions. Approximately 75 piecewise first-order polynomial functions are required to make a rough approximation of $u_C$. Therefore, a total number of 75 B-splines is chosen. The learning rate is chosen small, $\gamma$=0.4. The tracking error that remains after learning can be seen in figure 5.12. (Note the difference in scale compared to figure 5.11).

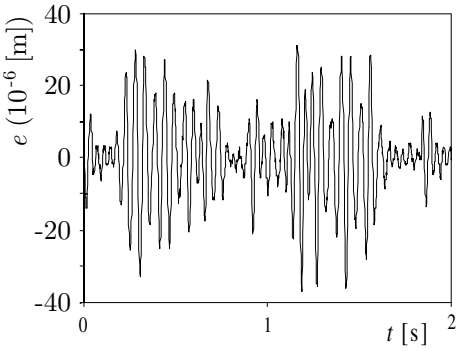



*Figure 5.12: Tracking error using initial B-spline distribution*

The reference path had to be presented 8 times before learning had converged and no further improvement could be obtained. The tracking error has been reduced by a factor 10 compared to the tracking error of the feedback controller.

**Experiment 5.5 (Result of the clustering algorithm)**

To obtain an even higher performance, fuzzy clustering is used to optimise the B-spline distribution. After the fuzzy clustering algorithm (as described in chapter 4) has been performed twice, no further improvement of the tracking error could be obtained. In figure 5.13, the tracking error that resulted after optimisation is shown.

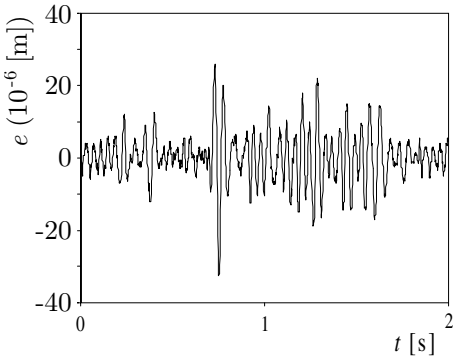



*Figure 5.13: Tracking error using improved distribution*

It can be seen that fuzzy clustering has been able to obtain a B-spline distribution that gives a smaller tracking error than could be obtained by the initial B-spline distribution.

### 5.2.4 Design of a Parsimonious LFFC

**Design Step 1: Design the feedback controller**

The same feedback controller is used as in case of the time-indexed LFFC:

$$C(s) = (5538s + 275280)\frac{(400\pi)^2}{s^2 + 100\pi s + (400\pi)^2} \tag{5.11}$$

**Step 2: Determine the inputs of the feed-forward part**

The required inputs of the feed-forward part can be obtained by expressing the plant dynamics in the form presented in chapter 3. For the LiMMS, this yields the following state-space description (see chapter 3):

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} &= \begin{bmatrix}0 & 1\\ 0 & -\dfrac{d_L}{m_L}\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ -\dfrac{F_C(x,\dot{x})}{m_L} - \dfrac{F_f(\dot{x})}{m_L}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{1}{m_L}\end{bmatrix}u\\ y &= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix}\end{aligned} \tag{5.12}$$

Where $F_f(\dot{x})$ represents the Coulomb friction and the stiction (note that this term is not present in the model used in chapter 3), $F_C(x)$ represents the cogging force and $d_L$ is the viscous friction. In section 3.2 it was derived that the LiMMS (5.12) belongs to the class of plants that can be controlled by means of LFFC. From (5.12) we can derive the following desired feed forward signal:

$$\begin{aligned}u_d &= m_L\left(\frac{d_L}{m_L}\dot{r} + \frac{F_C(r,\dot{r})}{m_L} + \frac{F_f(\dot{r})}{m_L} + \ddot{r}\right)\\ &= d_L\dot{r} + F_C(r,\dot{r}) + F_f(\dot{r}) + m_L\ddot{r}\end{aligned} \tag{5.13}$$

The feed-forward part should thus have the following inputs $\{r, \dot{r}, \ddot{r}\}$.

**Step 3: Choose the structure of the feed-forward part**

In (5.13) it can be clearly seen that the $u_d$ consists of 3 independent parts.

- *Friction compensation*: $d_L\dot{r} + F_f(\dot{r})$ , which requires a BSN that has $\dot{r}$ as input.
- *Cogging compensation*: $F_{cog}(r,\dot{r})$ . In the first part of this section, it was argued that the cogging characteristic differs for positive and negative velocities. We could compensate cogging using a BSN that has $r$ and $\dot{r}$ as inputs. Since the cogging characteristic only depends on the motion direction, it is also possible to compensate cogging by 2 BSNs, one for each motion direction. From a parsimonious point of view, the latter option is attractive.
- *Compensation of the inertia*: $m_L\ddot{r}$ , requiring a BSN that has $\ddot{r}$ as input.

By creating one BSN for each term, the following parsimonious LFFC results.

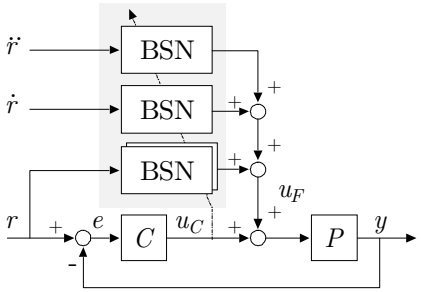



*Figure 5.14: Parsimonious LFFC for the LiMMS*

**Step 4: Choose the B-Spline distribution**

Once a network structure has been created, the B-Spline distribution should be selected. The B-splines that are defined on the reference motion, need to be sufficiently wide when seen in time (see chapter 3). Therefore, the analysis that is performed in step 2 of the design procedure for a time-indexed LFFC should also be carried out for the LFFC. From the previous design procedure, we know that the B-splines need to be at least 0.0192 [s] wide. With this in mind, we design a B-spline distribution on the basis of prior knowledge of the plant dynamics and the disturbances:

– *Cogging BSNs.* The period of the cogging is known to be 1.6 [cm]. In order to be able to accurately approximate the desired feed-forward signal, which can roughly be described by $10\sin\left(1.6\,10^{-2}r\right)$, we choose to define 1000 B-splines on the input domain, [−0.61 [m], 0.11 [m]]. This means that 21 B-splines are defined on 1 cogging period. The width of the B-splines is 0.00144 [m]. In figure 5.15 a part of the B-spline distribution and the desired feed-forward signal (based on the cogging in the simulation model) are shown. The width of the B-splines is sufficiently small to accurately approximate the desired feed-forward signal.

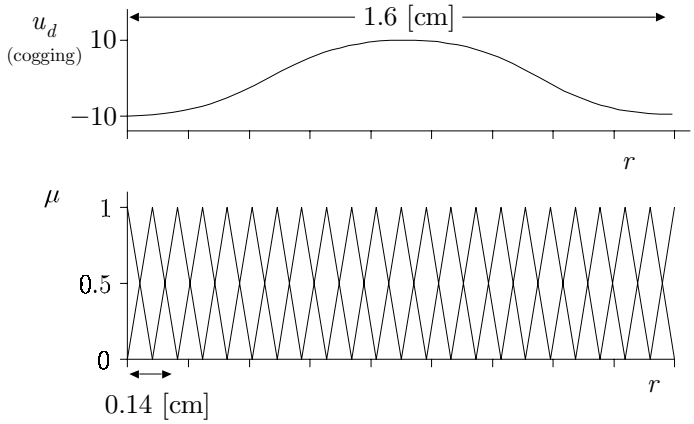



*Figure 5.15: B-spline distribution of the friction BSN*

In order to guarantee that, seen in time, the B-splines are at least 0.0192 [s] wide, this BSN must be trained at a reference velocity that satisfies the following condition:

$$\dot{r}(t) < \frac{0.00144\ [\mathrm{m}]}{0.0192\ [\mathrm{s}]} = 0.075\ [\mathrm{ms^{-1}}] \tag{5.14}$$

– *Friction BSN.* Because the friction characteristic is non-linear, we choose to define 11 B-splines on the input domain, $\left[-1\,[\mathrm{ms}^{-1}],1\,[\mathrm{ms}^{-1}]\right]$. Note that this B-spline distribution is not dense enough to compensate for stiction.

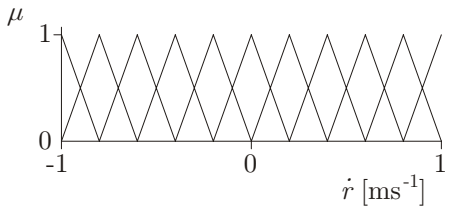



*Figure 5.16: B-spline distribution of the friction BSN*

The width of the B-splines is $0.2\,[\mathrm{ms}^{-1}]$. Therefore, this BSN must be trained at a reference acceleration that satisfies:

$$\ddot{r}(t) < \frac{0.2\,[\mathrm{ms}^{-1}]}{0.0192\,[\mathrm{s}]} = 10.4\,[\mathrm{ms}^{-2}] \tag{5.15}$$

– *Inertia BSN.* This BSN has to learn a simple linear function, namely $m_L\ddot{r}$. This feed-forward signal can be generated using only 3 B-splines defined on $\left[-5\,[\mathrm{ms}^{-2}], 5\,[\mathrm{ms}^{-2}]\right]$.

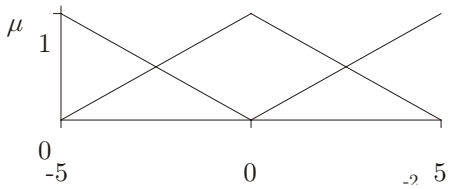



*Figure 5.17: B-spline distribution of the inertia BSN*

**Step 5: Choose the learning rate**

In the experiments the learning rate is chosen relatively small, i.e. 0.1.

**Step 6: Train the LFFC**

Because the LFFC consists of multiple BSNs, a series of training experiments needs to be chosen in which each BSN is trained separately. The reference motions must be chosen such that the desired feed-forward signal of one of the untrained BSNs (and thus one physical phenomenon) becomes dominant. In case of the LiMMS, the first BSN that is trained is the cogging BSN. By choosing a reference motion that has a low velocity and a low acceleration, the effects of viscous friction and the inertia can be made small. For this reference motion, the dominant physical effects

are cogging, Coulomb friction and stiction. Since two cogging BSNs exist, one for positive and one for negative velocities, these are able to learn to compensate the Coulomb friction as well.

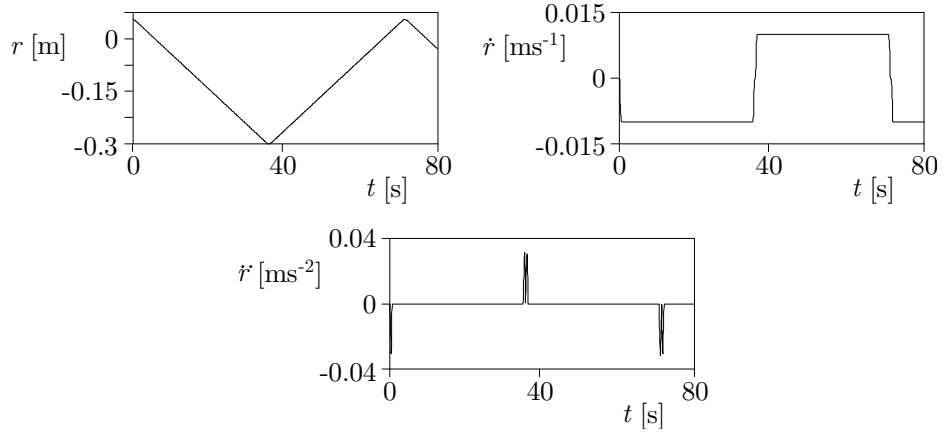



*Figure 5.18: Training motion for the cogging BSNs*

After applying the above training experiments, the following input-output mappings of the separate BSNs results (figure 5.19) It can be clearly seen that these contain an sine-shaped component, to compensate for cogging, and an offset, to compensate for Coulomb friction. From the fact that the sine-shaped parts of the feed-forward signals differ, we may conclude that the cogging characteristic indeed depends on the motion direction.

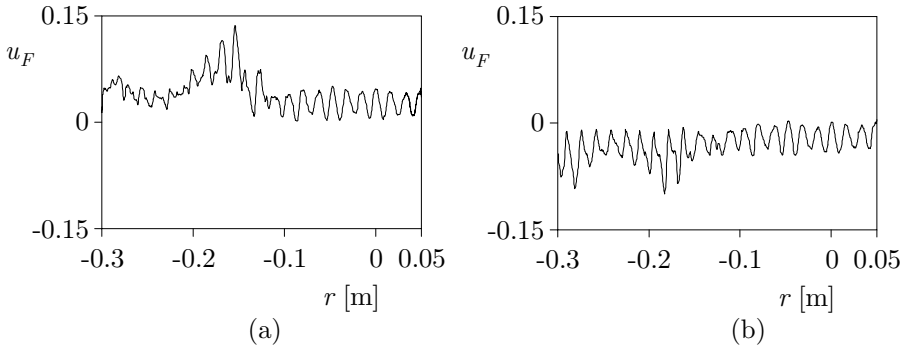



*Figure 5.19: a) IO-mapping of the cogging BSN for positive velocities*
*b) IO-mapping of the cogging BSN for negative velocities*

Next, the acceleration BSN it trained. Since cogging and Coulomb friction are compensated by the cogging BSNs, the reference motion should be chosen such that the effects of the viscous friction are small. This can be done by selecting a

reference motion which consist of a rapid sequence of positive and negative accelerations (see figure 5.20).

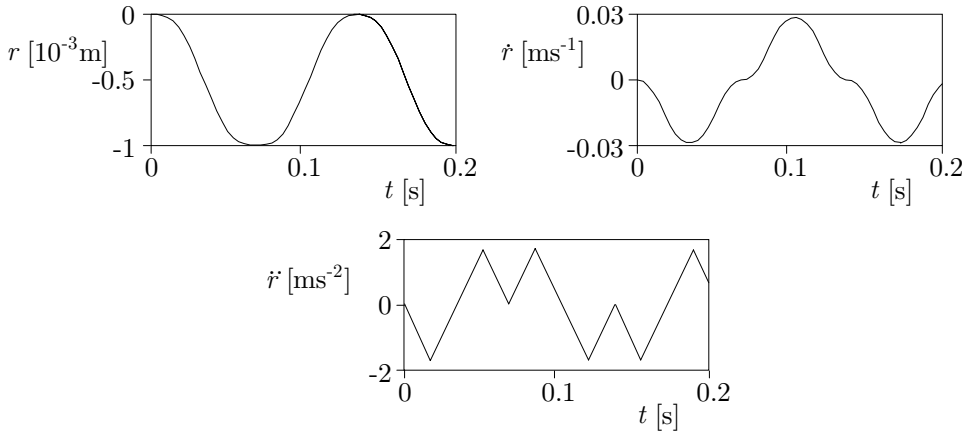



*Figure 5.20: Training motion for the inertia BSN*

Figure 5.21, the IO-relation of the BSN that results is shown. It equals the input-output relation that was to be expected, i.e. a linear function passing through the origin.

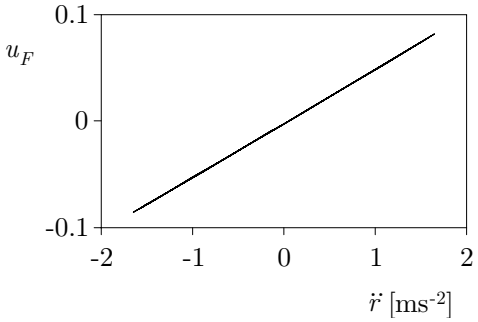



*Figure 5.21: IO-mapping of the inertia BSN*

Finally, the friction BSN is trained. Since all other phenomenons have been compensated for any type of reference motion can be applied. In order to accurately compensate friction, we choose to train the friction BSN in two stages. The reference motion in the first experiment contains a wide range of velocities, see figure 5.22.

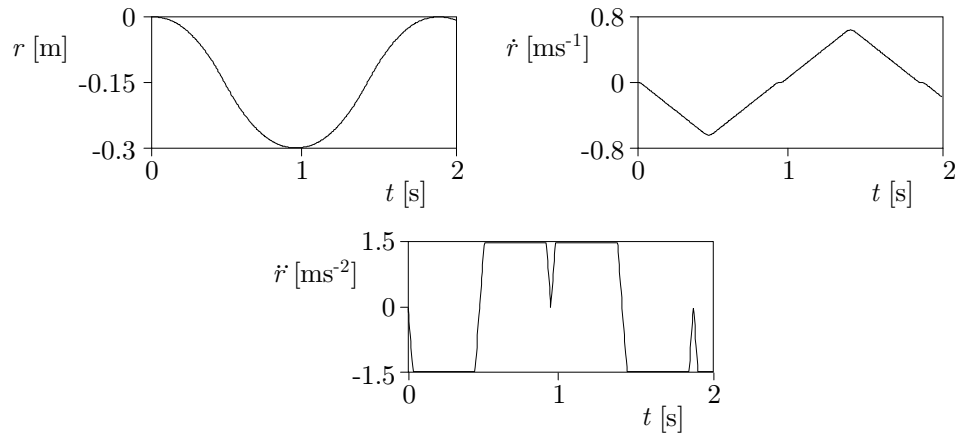



*Figure 5.22: First training motion for the velocity BSN*

In the above reference motion, the reference velocity only has a small velocity for a short time. The LFFC may not have learned to accurately compensate friction at low velocities. Therefore, a second reference motion is applied in which the reference velocity is small.

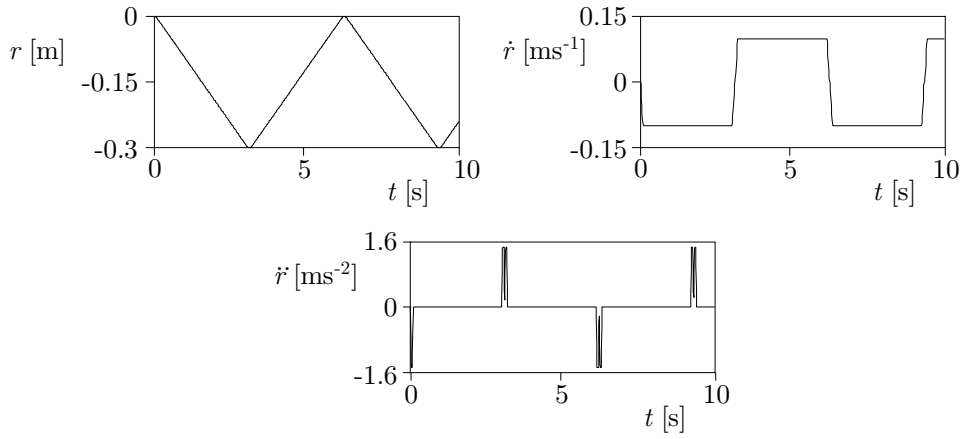



*Figure 5.23: Training of the velocity BSN (2)*

When looking at the input-output mapping of the friction BSN (figure 5.24a), it can be seen that it did not learn to accurately compensate friction at high velocities. In the intermediate velocities the viscous friction appears to be linear. Figure 5.24b shows the input-output mapping of the inertia BSN.

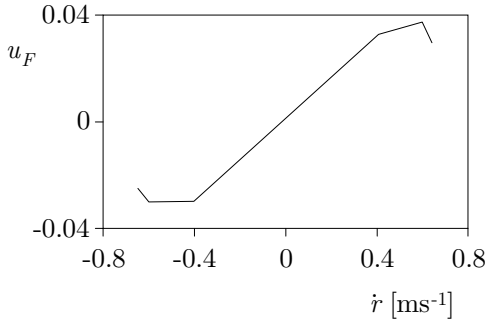



*Figure 5.24: IO-mapping of the friction BSN*

### 5.2.5 Evaluation Experiments

In this section, the performance of the parsimonious LFFC is evaluated. This is done by presenting 3 reference motions, for which the parsimonious LFFC has not been trained. We will compare the tracking performance of a parsimonious LFFC with the tracking performance of the feedback controller and a time-indexed LFFC.

**Experiment 5.6 (Evaluation motion I)**

The first evaluation motion is a low-velocity motion (figure 5.25).

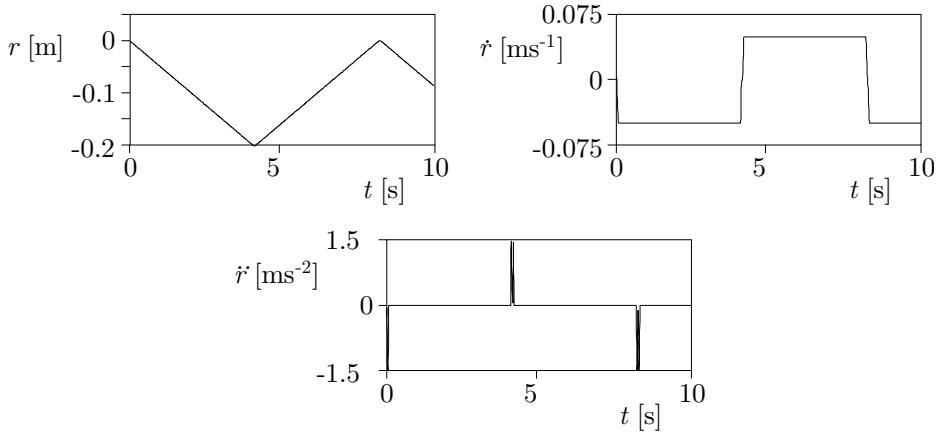



*Figure 5.25: Evaluation motion I: slow*

This evaluation motion has been performed by a time-indexed LFFC in which the width of B-splines equals 0.04 [s]. In figure 5.26 the tracking performance of the controllers is presented (note the different scales).

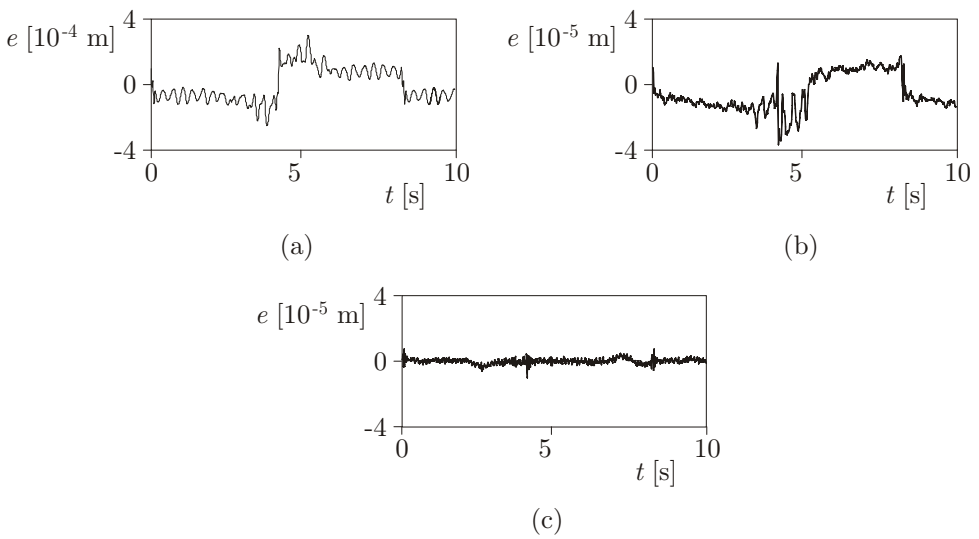



*Figure 5.26: a) Tracking error of the feedback only*
*b) Tracking error of the parsimonious LFFC*
*c) Tracking error of the time-indexed LFFC (d=0.04[s])*

The parsimonious LFFC obtains a tracking error that is a factor 10 smaller than the tracking error of the feedback controller. Figure 5.26b shows an offset in the tracking error that has the same shape as the reference velocity. This is probably caused by friction. Apparently, the friction BSN is unable to completely compensate for friction. Furthermore, a sine shaped component appears in the tracking error. It may be either so that the cogging BSN has not been trained properly or that another phenomenon is present, probably errors in the commutation, that causes a similar tracking error.

The time-indexed LFFC shows a much smaller tracking error than the parsimonious LFFC. Because the feed-forward signal of the time-indexed LFFC has converged, we assume that the phenomenons that are compensated for are reproducible. We conclude that either the BSNs of the parsimonious LFFC have not been trained accurately enough or that a phenomenon exists that cannot be compensated by this set of BSNs. Because the tracking error that remained after training the cogging BSN (a figure of this is not presented in this thesis) was much smaller than the amplitude of the sine-shaped part of the tracking error, the latter conclusion is most likely.

**Experiment 5.7 (Evaluation motion II)**

The reference position, velocity and acceleration of the second reference motion are given in figure 5.27.

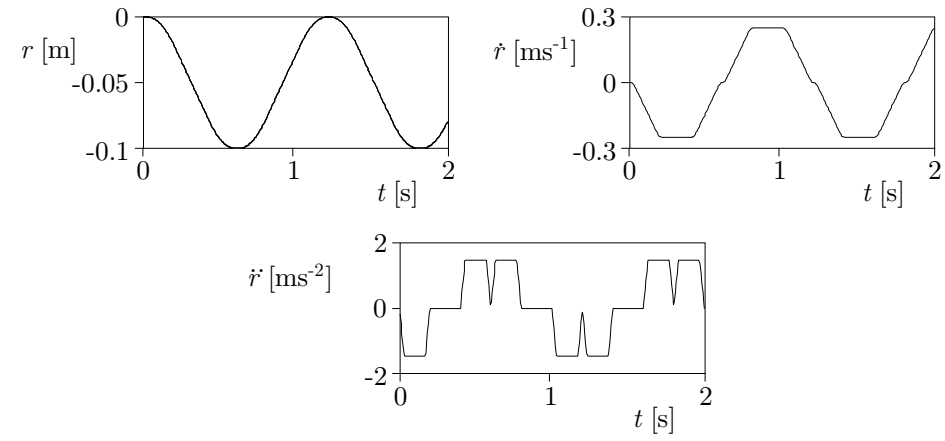



*Figure 5.27: Evaluation motion II: intermediate*

The resulting tracking errors are shown below.

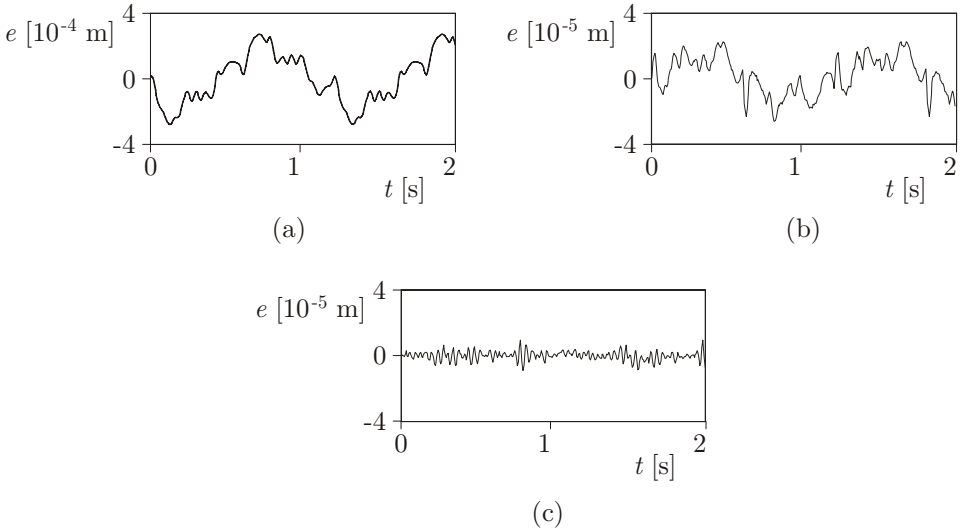



*Figure 5.28: a) Tracking error of the feedback only*
*b) Tracking error of the parsimonious LFFC*
*c) Tracking error of the time-indexed LFFC (d=0.04[s])*

Again, the tracking error of the parsimonious LFFC is a factor 10 smaller than the tracking error of the feedback controller. The conclusion that a phenomenon other than cogging causes the ripple in the tracking error is confirmed by figure 5.28b. The average amplitude of the ripple in the tracking error has increased, i.e. the

disturbance depends on the velocity of the LiMMS. The smallest tracking error was obtained by the time-indexed LFFC.

**Experiment 5.8 (Evaluation motion III)**

The last evaluation motion is presented in figure 5.29.

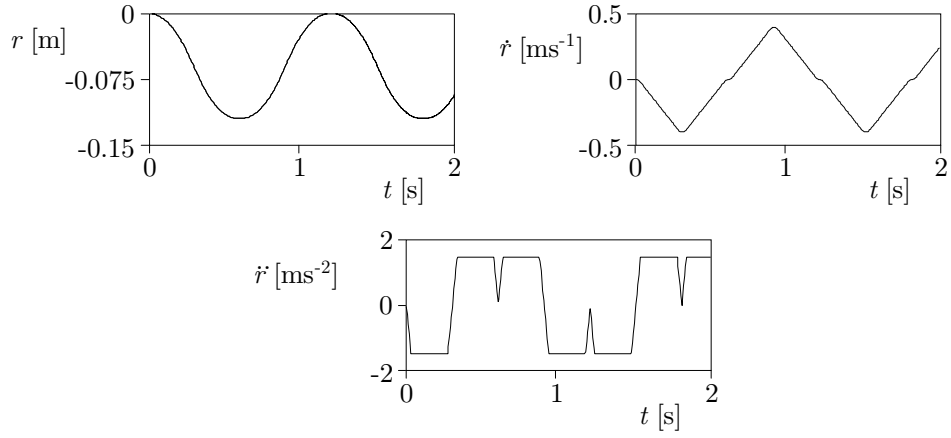



*Figure 5.29: Evaluation motion III: fast*

Figure 5.30 presents the resulting tracking errors.

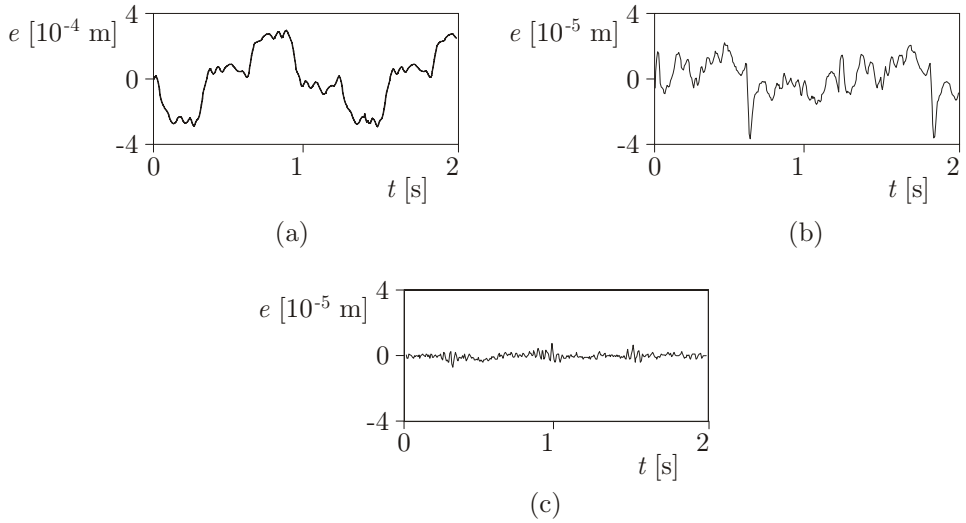



*Figure 5.30: a) Tracking error of the feedback only*
*b) Tracking error of the parsimonious LFFC*
*c) Tracking error of the time-indexed LFFC (d=0.03[s])*

Figure 5.30b shows a relatively large tracking error at the points in time where the reference velocity equals 0. We conclude that the parsimonious LFFC is unable to compensate for stiction, which is to be expected on the basis of the input-output relation of the velocity BSN (5.24). As before, the tracking of the time-indexed LFFC is the smallest.

### 5.2.6 Discussion

Experiments using a time-indexed LFFC were performed to validate the stability conditions derived in chapter 2. The results of the experiments are similar to the results of simulations, i.e. the value of the $d_{min}$ is accurate while the maximum learning rate has a rather conservative value. In spite of the conservative value of the maximum learning rate, learning converged in a limited number of runs (typically 10 to 15).

A parsimonious LFFC has been designed for the LiMMS. To train the parsimonious LFFC a number of training experiments were selected. Experiments that are not discussed in this thesis show that this needs to be done carefully. Selecting an unfit training motion causes the BSN to approximate the wrong feed-forward signal. All BSNs that are trained subsequently also approximate an erroneous feed-forward signal. After training, the input-output relation of the BSN should be inspected to ensure that it has not approximated (parts of) a desired feed-forward signal of another BSN in the parsimonious LFFC. The training motions proposed in this chapter yielded a parsimonious LFFC that was able to obtain a 10 times smaller tracking error than the feedback controller. The time-indexed LFFC showed an even smaller tracking error than the parsimonious LFFC. This is probably caused by a phenomenon other than the ones considered in the selection of the BSN structure.

## 5.3 Control Loading System

### 5.3.1 Introduction

In this section, we will apply LFFC to a part of a flight simulator set-up. A flight simulator is a cost-effective and safe way to train a pilot in handling an aircraft. The simulator enables the pilot to experience all sorts of flight situations without running into the risks involved when using a real aircraft. An important part of a flight simulator is the so-called Control Loading System (CLS). This system comprises the command stick of the simulator and the hard- and software

connected to this stick that emulate the behaviour of an aircraft as experienced by the pilot through the stick. Here, we consider a specific realisation of such a control loading system. The plant part of the set-up is presented in figure 5.31.

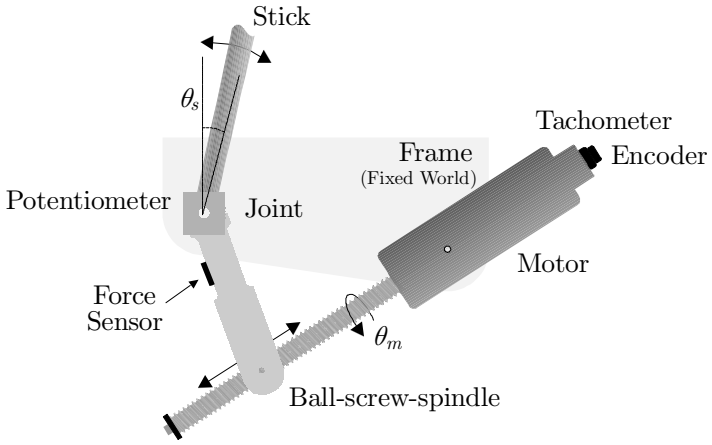



*Figure 5.31: Plant part of the set-up*

It consists of the following components:

- *The stick.* The stick has one rotational degree of freedom. Its angular position, $\theta_s$ [rad], is measured by a potentiometer.

- *A spindle and a ball-screw.* Together these form the transmission between motor and stick. The transmission between the motor and the stick is considered stiff. This means that the angular velocity of the stick, $\dot{\theta}_s$ [rad s$^{-1}$], depends on the angular velocity of the motor, $\dot{\theta}_m$ [rad s$^{-1}$], in a linear way:

$$\dot{\theta}_s = n\ \dot{\theta}_m\ [\text{rad}\,\text{s}^{-1}] \tag{5.16}$$

  In this set-up, the transmission coefficient $n$=0.004;

- *An electric motor.* A tachometer and an encoder are mounted on the motor to measure respectively the angular velocity and the angular position, $\theta_m \in [0\text{ rad}, 2\pi\text{ rad}]$.

- *An electric power amplifier* (not shown in figure 5.31).

– *A force sensor.* The force sensor measures the force, $F_{\text{ext}}$ [N], applied on the stick by the pilot.

By controlling the plant appropriately, the 'mechanical impedance' the pilot experiences at the stick can be controlled. This is done as shown in figure 5.32. The motor is contained in a velocity loop with a PI-controller that is incorporated in the off-the-shelf amplifier. This loop is called the inner loop. The commanded motor angular velocity, $\dot{\theta}_{m,d}$ [rad s$^{-1}$], for the inner loop is calculated by a model of the aircraft controls. Inputs to this model are the force, $F_{ext}$ [N], and the angular position of the simulator stick, $\theta_s$ [rad]. This forms a secondary MISO loop, called the outer loop.

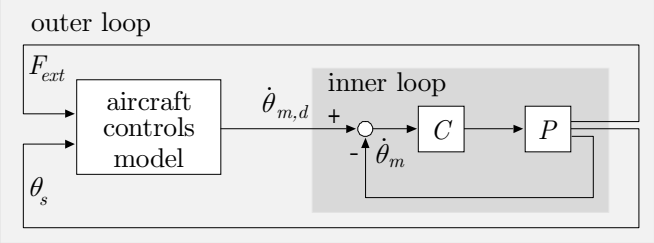



*Figure 5.32: Control Loading System set-up*

To make the simulation as realistic as possible, the impedance that one feels when manipulating the stick, should be determined completely by the model of the aircraft controls. To achieve this, the controller, $C$, in the inner loop has to compensate for all dynamic properties of the plant, $P$; only some (small) rotational inertia is acceptable, as any aircraft stick will feature this phenomenon. However, in the set-up used in this research, the inner loop is unable to fully realise this. When the user moves the stick, small irregularities can be felt. In the following, we will discuss possible causes of these irregularities and examine whether LFFC can overcome this problem.

### 5.3.2 Motivation for LFFC

The most important dynamic properties of the plant that the inner loop has to deal with are:

– *Cogging force.* The cogging force is the magnetic force between the permanent magnets and the iron in the motor. The magnitude of the cogging

force depends on the angular position of the motor. It is known that there are 32 poles in the motor and therefore we expect 32 cogging periods per $2\pi$ [rad] motor rotation.

- *Friction in the motor and the ball-screw.* It is assumed that the friction characteristic can be approximated by a Stribeck curve [Armstrong-Helouvry *et al.*, 1994]. The Coulomb friction and the stiction cause the so-called reversal bump. The reversal bump is the tracking error that occurs when the motion direction changes; the feedback controller is unable to compensate for the abruptly change in the Coulomb friction force and the stiction. When the reversal bump is significant, the pilot looses the illusion of handling a real system.
- *Inertia of the motor, the transmission and the stick.* Because of the small transmission coefficient ($n$=0.004) this phenomenon is small compared to the friction force.
- *Measurement noise.*

Of these phenomenons only measurement noise is stochastic in nature; all others are reproducible and static, their momentary value is related to the momentary state of the plant. Therefore, we propose to compensate the disturbances by an LFFC [Velthuis *et al.*, 1998; Vrielink, 1998; Bouwhuis, 1999]. In the following, the separate design steps will be presented. Because in normal operation the control loading system does not perform repetitive motions, only the design of a parsimonious LFFC is considered.

### 5.3.3 Design Procedure for a Parsimonious LFFC

**Step 1: Design the feedback controller**

As a feedback controller we use the standard PI controller that was present in the set-up (5.17).

$$C(s) = 0.005\left(1 + \frac{1}{0.01s}\right) \tag{5.17}$$

**Step 2: Choose the inputs of the feed-forward part**

In order to select the inputs of the feed-forward part, we will express the dynamics in the form that was proposed in chapter 3. This results in the following state-space description:

$$\begin{bmatrix}\dot{\theta}_m \\ \ddot{\theta}_m\end{bmatrix} = \begin{bmatrix}0 & 1 \\ 0 & \dfrac{-d}{J}\end{bmatrix}\begin{bmatrix}\theta_m \\ \dot{\theta}_m\end{bmatrix} + \begin{bmatrix}0 \\ \dfrac{-F_c(\theta_m) - F_f\left(\dot{\theta}_m\right)}{J}\end{bmatrix} + \begin{bmatrix}0 \\ \dfrac{1}{J}\end{bmatrix} u$$
$$\dot{\theta}_m = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\theta_m \\ \dot{\theta}_m\end{bmatrix} \tag{5.18}$$

With,

$J$: the inertia of the motor, the transmission and the stick;
$d$: the viscous friction in the transmission;
$F_c(\theta_m)$: the cogging force;
$F_f\left(\dot{\theta}_m\right)$: the Coulomb friction and the stiction.

Next, we check whether the control loading system can be controlled by means of LFFC.

- $\boldsymbol{C}$ *may contain only one element,* $\boldsymbol{C}_j$*, that is unequal to zero.* In (5.18), $\boldsymbol{C} = \begin{bmatrix}0 & 1\end{bmatrix}$ , which means that this condition is satisfied.
- $\boldsymbol{C}_j$ *must be invertible.* Since $\boldsymbol{C}_2 = 1$, it is invertible. This condition is also satisfied.
- $\boldsymbol{B}_n$ *must be invertible.* In $\boldsymbol{B}_2 = \dfrac{1}{J}$, which is invertible if $J > 0$. Since J>0, the last condition is also satisfied.

We can conclude that the control loading system can be controlled by means of LFFC. The desired feed-forward signal is given by:

$$\begin{aligned} u_d &= d\dot{\theta}_{m,d} + F_c\left(\theta_{m,d}\right) + F_f\left(\dot{\theta}_{m,d}\right) + J\ddot{\theta}_{m,d} \\ &\approx d\dot{\theta}_{m,d} + F_c\left(\theta_{m,d}\right) + F_f\left(\dot{\theta}_{m,d}\right) \end{aligned} \tag{5.19}$$

With,

$\theta_{m,d}$: the reference motor angular position;
$\dot{\theta}_{m,d}$: the reference motor angular velocity.

$\ddot{\theta}_{m,d}$ : the reference motor angular acceleration.

Due to the small transmission coefficient, the term $J\ddot{\theta}_{m,d}$ is much smaller than the rest of the terms and can be disregarded. This means that, in order to be able to generate the desired feed-forward signal (5.19), $\theta_{m,d}$ and $\dot{\theta}_{m,d}$ should be selected as inputs of the feed-forward part.

**Step 3: Choose the structure of the feed-forward part**

Because none of the terms in (5.19) depends on $\theta_{m,d}$ as well as $\dot{\theta}_{m,d}$, friction and cogging can be compensated by separate BSNs. The LFFC that results is depicted in figure 5.33.

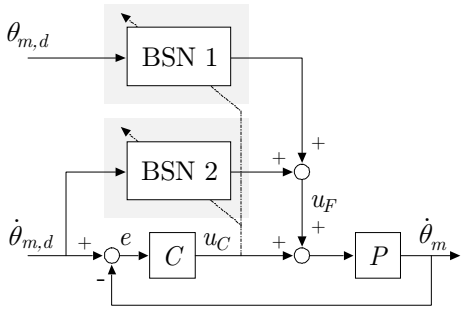



*Figure 5.33: LFFC structure for the CLS*

Experiments showed that the LFFC was able to compensate the unknown friction but not the cogging force. The fact that BSN 1 could not be trained properly can be understood as follows. In the control loading system, the reference angular velocity of the simulator stick is tracked accurately at the expense of errors in the angular position. This was chosen because the pilot does not interpret small errors in the angular position of the simulator stick as unrealistic. However, errors in the angular velocity of the stick are interpreted as unrealistic. In order to prevent too large errors in the angular position of the motor (and thus of the stick) these are compensated by the outer loop (figure 5.34). The aircraft controls model consists of a model of the plant that is to be simulated and a correction term for errors in the angular position of the motor. Because a correct angular velocity of the stick is the most important, $K$ is chosen small.

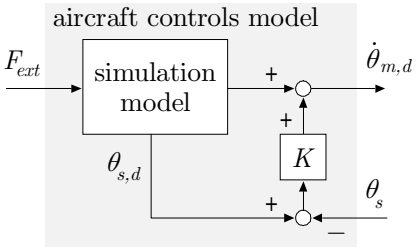



*Figure 5.34: Aircraft controls model*

Motor angular position errors of approximately 0.21 [rad] occur frequently, while in our set-up the period of the cogging force is 0.19 [rad]. So the error in the angular position of the motor is of the same order of magnitude as the period of the cogging force. Note that BSN 2, which compensates for friction forces, cannot prevent positional errors. Firstly, it is unable to exactly compensate the friction force, such that small errors in the angular velocity (and thus in the angular position) remain. Furthermore, random disturbances cause errors in the angular velocity. Because $K$ is chosen small, and consequently errors in the angular position of the motor are hardly compensated, the errors in the angular velocity may result in a large error in the angular position of the motor.

When this is the case, compensating the cogging force using a feed-forward controller is not possible. This is illustrated in figure 5.35, where the magnitude of the cogging force is shown as a function of the position of the stick.

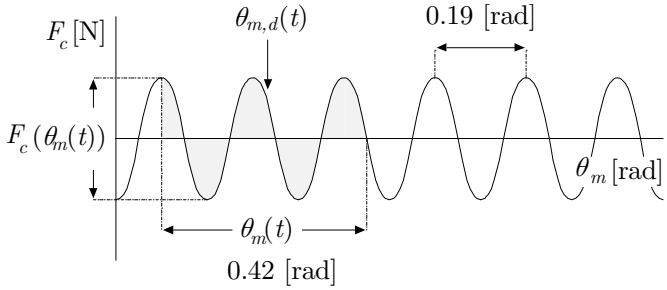



*Figure 5.35: Cogging force as a function of the motor angle*

At a certain point in time, $t$, the reference angle is $\theta_{m,d}(t)$. The expected cogging force at $\theta_{m,d}(t)$ is $F_c\left(\theta_{m,d}(t)\right)$. However, due to fact that the reference angular velocity is tracked instead of the reference angular position, the actual value of the motor angle is $\theta_m(t) \in \left[\theta_{m,d}(t) - 0.21, \theta_{m,d}(t) + 0.21\right]$ [rad]. It can be seen that there

is no direct relation between the actual cogging force, $F_c(\theta_m(t))$, and the expected cogging force, $F_c\left(\theta_{m,d}(t)\right)$.

In figure 5.36 an example of this is shown in the time-domain. The upper part of the figure shows $F_c\left(\theta_{m,d}(t)\right)$, whereas the lower part shows $F_c(\theta_m(t))$. No relation between $F_c\left(\theta_{m,d}(t)\right)$ and $F_c(\theta_m(t))$ exists.

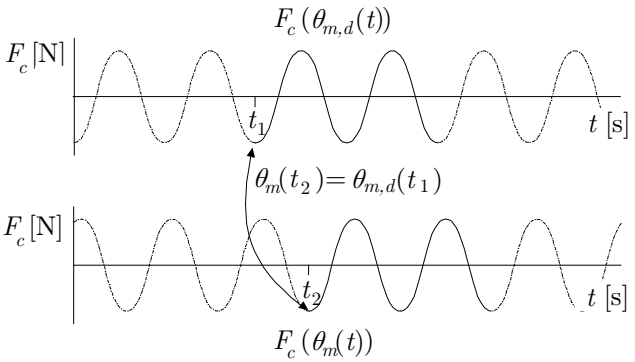



*Figure 5.36: Expected and actual cogging force.*

This means that the cogging force cannot be compensated in feed-forward. To overcome this, the measured angular position of the motor can be used as input of the BSN instead of the reference angular position [Gomi and Kawato, 1993]. In other words, as far as position-related effects are concerned, an inverse model is contained in a feedback loop instead of a feed-forward path. This results in the system shown in figure 5.37.

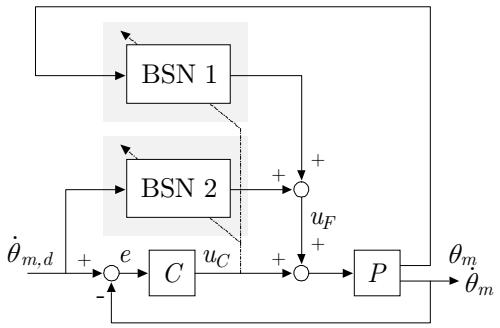



*Figure 5.37: Learning feed-forward / feedback controller*

**Step 4: Choose the B-Spline distribution**

The distribution of the B-splines is chosen by rule of thumb, based on the characteristics of the disturbances the BSNs have to compensate.

- For the BSN that has $\dot{\theta}_{m,d}$ as an input, we know that it has to compensate the unknown friction, which can be approximated by the Stribeck curve. To to be able to accurately compensate for friction (especially at low velocities) we define 160 B-splines on the input domain of $\dot{\theta}_{m,d}$ $[-262, 262]$ [rads$^{-1}$].
- Based on the configuration of the motor, we know that 32 cogging periods exist per motor rotation. We choose to define 500 B-splines on the input domain of $\theta_{m,d}$ $[0, 2\pi]$ [rad]. This means that each cogging period is mapped by approximately 15 B-splines.

**Step 5: Choose the learning rate**

The learning rate of both BSN's is chosen equal to 0.1.

**Step 6: Train the LFFC**

As before, the two BSNs have to be trained separately. First the friction BSN is trained and then the cogging BSN. This is motivated by the fact that due to friction, the existing PI controller is unable to smoothly control the motor at low velocities, which are the velocities that we wish to use for training the cogging BSN. The reference motion that is used to train the friction BSN, contains a wide range of reference velocities. The reference angular velocity and the reference angular position are presented in figure 5.38 and 5.39.

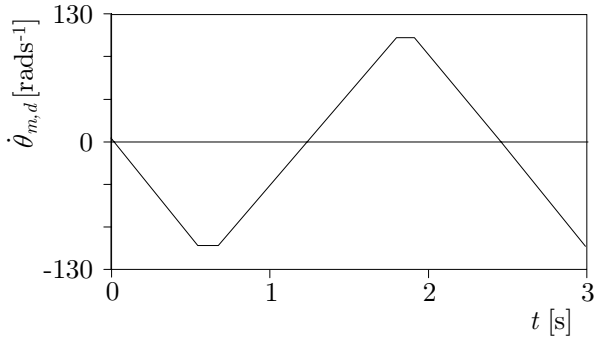



*Figure 5.38: Reference angular velocity used to train the friction BSN*

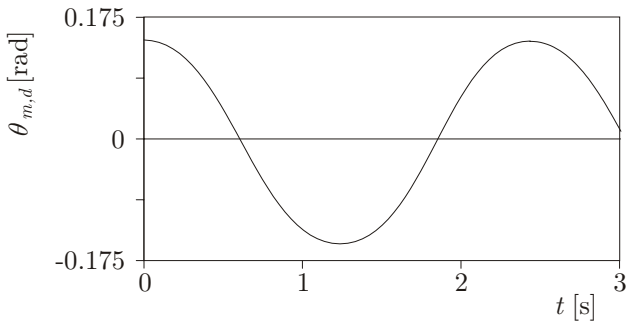



*Figure 5.39: Reference angular position used to train the friction BSN*

In figure 5.40, the input-output mapping of the friction BSN after learning is presented. The expected feed-forward signal consists of a linear component, compensating the viscous friction and of a constant component, of which the amplitude depends on the sign of the reference velocity, which compensates the Coulomb friction. The actual feed-forward indeed resembles the desired one, except for the parts where the velocity is high.

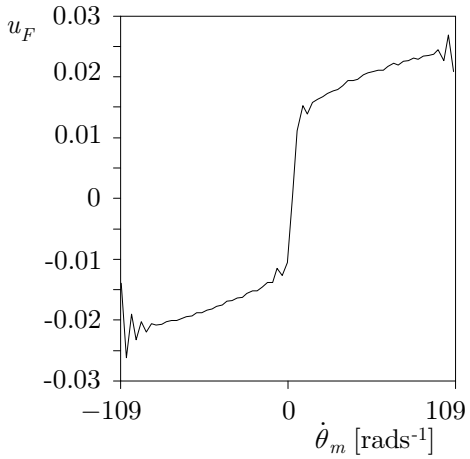



*Figure 5.40: IO-mapping of the friction BSN*

Next, the cogging BSN is trained. The is done by presenting a low-velocity reference motion, $\dot{\theta}_{m,d} = 2.18\,[\text{rads}^{-1}]$. The input-output mapping of the cogging BSN after learning has converged, is presented in figure 5.41.

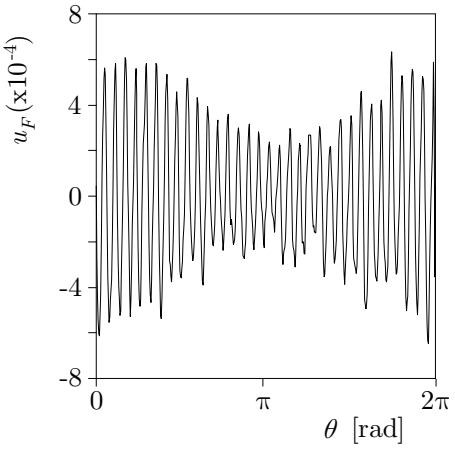



*Figure 5.41: IO-mapping of the cogging BSN*

Based on the configuration of the motor we expected 32 cogging periods per motor rotation. Figure 5.41 shows that the sine-shaped part of the feed-forward signal contains 32 periods, in accordance to what we expected.

### 5.3.4 Evaluation Experiments

To evaluate whether the parsimonious LFFC is able to compensate for friction and cogging, we present reference motions for which the LFFC has not been trained.

**Experiment 5.9 (Cogging validation, no LFFC)**

The goal of the first experiment is to show that the feedback controller is unable to compensate for cogging. The reference angular velocity and position of the motor that are presented in figure 5.42 and 5.43.

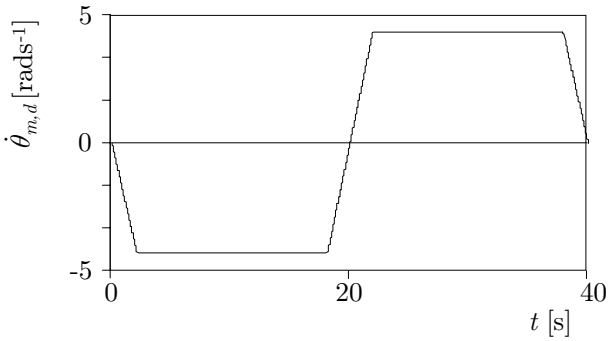



*Figure 5.42: Reference angular velocity*

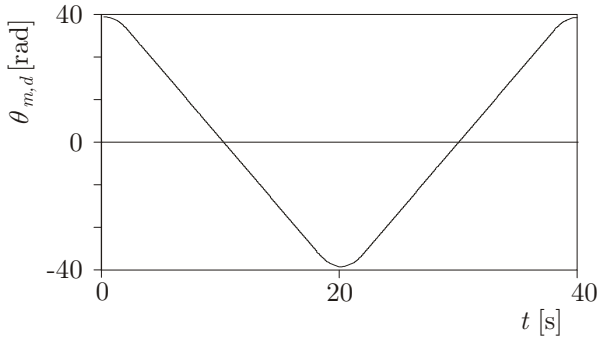



*Figure 5.43: Reference angular position*

To check whether the effects of cogging can be felt by the user, a frequency analysis of the motor angular velocity is performed. Figure 5.44 shows the power spectral density of $\dot{\theta}_m$. At $\dot{\theta} \approx 140[\text{rads}^{-1}]$ a peak exists. From the number of poles of the electro-motor and the long period of constant angular velocity it can be derived that this peak is caused by cogging forces. It is known that there are 32 cogging periods per $2\pi$ [rad] motor rotation. The reference angular velocity is equal to $\dot{\theta}_{m,d} = 4.36[\text{rads}^{-1}]$, which should result in an frequency of the cogging force of $32 \cdot 4.36[\text{rads}^{-1}] = 139[\text{rads}^{-1}]$. This frequency corresponds to the angular velocity of the peak in figure 5.44. Therefore, we conclude that the feedback controller is unable to compensate for cogging.

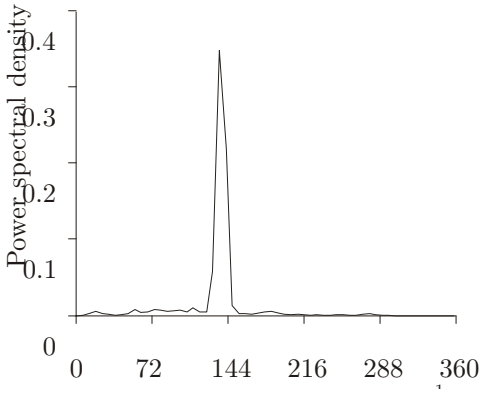



*Figure 5.44: Power spectral density of e, PI-control*

**Experiment 5.10 (Cogging validation, LFFC)**

Next, the parsimonious LFFC is used to control the set-up. Again, a frequency analysis of the resulting $\dot{\theta}_m$ is performed, see figure 5.45. From the absence of the peak, we conclude that LFFC has been able to properly compensate cogging.

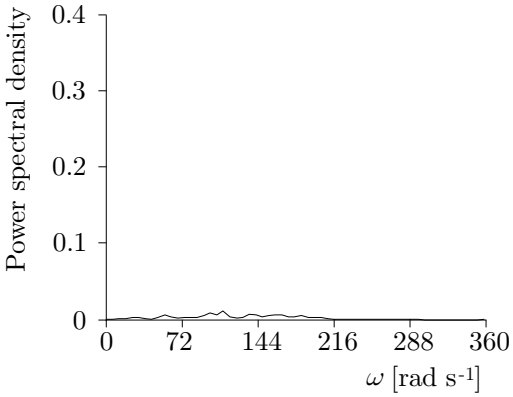



*Figure 5.45: Power spectral density of e, learning control*

**Experiment 5.11 (Friction validation, feedback only)**

To examine whether the LFFC is able to compensate friction, we compare its tracking error to the tracking error of a feedback controller. This is done by performing the following reference motion.

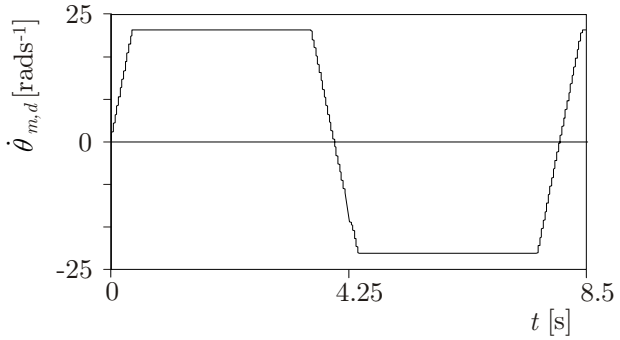



*Figure 5.46: Reference angular motor velocity to evaluate friction compensation*

The error in the angular velocity of the motor, that results when using only the feedback controller, is presented in figure 5.47. Friction causes large tracking errors when the reference motion changes sign.

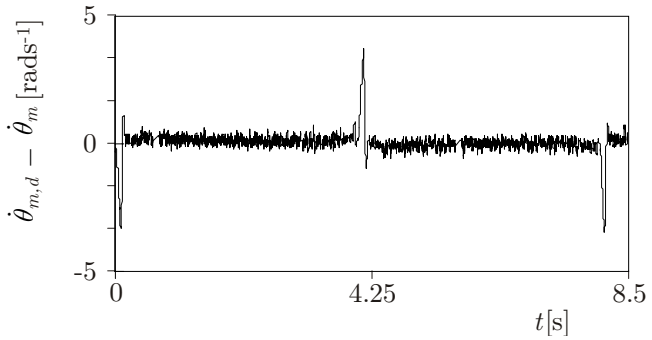



*Figure 5.47: Error in the angular velocity of the motor, feedback control only*

**Experiment 5.12 (Friction validation, LFFC)**

Next, the parsimonious LFFC is used for control. Figure 5.48 shows that the tracking error the LFFC is able to obtain. It can clearly be seen that the LFFC has not been able to fully compensate for the friction. Still significant tracking errors occur when the reference motion changes sign. Although, the magnitude of the tracking error is only slightly smaller than the magnitude in the previous experiment, the reduction of the peaks of a factor 0.6 gives already a much better feeling.

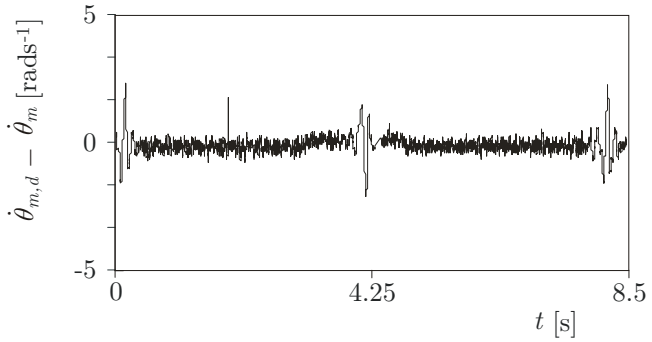



*Figure 5.48: Error in the angular velocity of the motor, LFFC*

## 5.3.5 Discussion

A parsimonious LFFC has been designed for the CLS. Experiments showed that the cogging force could not be compensated in feed-forward. This was caused by the fact that the positional tracking error is much larger than the period of the cogging force. Therefore, it was proposed to use the BSN in a feedback loop instead

of in feed-forward. The Learning FeedBack Controller (LFBC) was able to compensate for cogging. The learning behaviour of the LFBC was such that learning converged. However, more research is desirable to answer the following questions:

- Is there a minimum width of the B-splines in LFBC?
- What is the maximum learning rate?

The friction was compensated by means of an LFFC. For the CLS it is important that the error in the angular velocity is small when the motion direction changes. This tracking error is called the reversal bump. Experiments showed that the LFFC was able to obtain a somewhat smaller reversal bump than a feedback controller. Apparently, the LFFC was unable to accurately compensate stiction. However, in our opinion LFFC is an attractive option for friction compensation and more research on this topic should be performed [Spreeuwers, 1999].

# 6 Discussion

## 6.1 Review

In chapter 1, we have introduced the concept of LFFC, being a Feedback-Error-Learning (FEL) controller where the feed-forward part is a B-Spline Network (BSN). It is important to design the feed-forward part of the LFFC in such way that learning converges and that the problems that are associated with the curse of dimensionality are kept to a minimum. These two subjects are the main concerns of this work.

In chapter 2, we considered *time-indexed LFFC*, i.e., the case that the periodic motion time is the only input of the BSN. In this case, LFFC is similar to Iterative Learning Control (ILC) and Repetitive Control (RC). Therefore, we pursued the idea to use the convergence results of ILC and RC in the setting of LFFC.

In ILC and RC the feed-forward part is implemented as a memory loop. Instead of training the feed-forward part by the *output of the feedback controller*, the learning signal is obtained by filtering the *tracking error* using a learning filter. The learning filter is designed on the basis of a rough model of the controlled plant. A stability analysis shows that unmodelled high frequency dynamics may cause the feed-forward signal of the ILC and RC to diverge. Convergence can be obtained by adapting the memory loop in such a way that the frequency contents for which the model of the controlled plant (and thus the learning filter) is inaccurate are removed.

Time-indexed LFFC can be considered as a type of ILC / RC, in which learning is stabilised by replacing the memory loop by a BSN and in which the feedback controller is used as a learning filter. The frequency contents of the input-output

relation of a BSN depend on the *width* of the support of the B-splines. Choosing a large width results in a low-frequency feed-forward signal, while a small width yields a high-frequency feed-forward signal. Hence, the results from ILC and RC tell us that learning can be stabilised by choosing the width of the support of the B-splines in such way, that the BSN is unable to approximate the high-frequency contents of the learning signal. Under rather restrictive assumptions (one of which is that the plant is SISO LTI), we have been able to express this quantitatively; we derived stability conditions for the minimum width of the support of the B-splines, $d_{min}$, and for the maximum learning rate. The stability criteria were validated by means of simulations. The time-indexed LFFC was applied to two plants:

- *Mass-Spring-Mass (MSM) plant.* The MSM plant satisfies the assumptions that were made in the stability analysis. Simulations showed that the value of $d_{min}$ is accurate, while the maximum learning rate may be conservative.
- *LiMMS.* Because the LiMMS is a non-linear plant, it does not satisfy the assumptions that were made in the stability analysis. In spite of this, the results of the simulations are equivalent to the results of the simulations of the MSM plant.

In case of random reference motions, time-indexed LFFC is not applicable. Then, *path-indexed LFFC* can be applied, i.e., the reference signal and derivatives / integrals thereof should be used as inputs for the feed-forward part. This type of LFFC was considered in chapters 3 and 4.

We have not been able to derive convergence conditions for the design parameters of a BSN in case of path-indexed LFFC. It was argued that in some cases (e.g. when the reference velocity is constant), the stability conditions of the time-indexed LFFC can be transformed into stability conditions of path-indexed LFFC, basically by saying that the supports of the (multi-dimensional) B-splines have to cover at least $d_{min}$ [s] of the reference motion. Designing a B-spline distribution that satisfies this is difficult and may lead to very large B-spline widths, which is unattractive.

Therefore, another approach was considered. We proposed to select a B-spline distribution without regarding stability conditions and add a stabilising measure to the LFFC to make learning converge. One option to do so would be to change the cost-function that is minimised by the learning mechanism. This is a well-known technique in neuro-fuzzy modelling and is called regularisation. However, it proved to be difficult to select the parameters of the regularisation mechanism in such way that learning in LFFC converges. Therefore, an alternative option was pursued, again inspired by ILC. Regularisation is performed by filtering the learning signal of the LFFC instead of changing the learning mechanism. The main idea is that learning converges when the learning signal is such that all frequency contents above ${d_{min}}^{-1}$ [Hz] is removed. This can, for example, be achieved by filtering with

an appropriate time-indexed BSN. Simulations confirmed that by adding such a filter, convergence of path-indexed LFFC is obtained. However, a formal analysis that explains the observed results is still lacking.

Which specific inputs should be selected in case of path-indexed LFFC was determined on the basis of a (structurally correct) state-space representation of the plant dynamics. We showed that a plant can only be controlled by means of path-indexed LFFC when the state space representation fulfils a number of conditions. Hence, of the plants mentioned above, only the LiMMS can be controlled by means of path-indexed LFFC.

When the inputs of the BSN are chosen straightforwardly on basis of the plant model, a multi-dimensional BSN results. This confronts us with problems that are associated with the curse of dimensionality. In neuro-fuzzy modelling, it has been shown that these problems can be overcome by splitting the multi-dimensional BSN into several BSNs that have a lower input-dimension. This is done on the basis of the so-called analysis-of-variance (ANOVA) representation of a function. The resulting collection of BSNs is known as a parsimonious network structure. In line with this, a path-indexed LFFC is called *parsimonious* when it consists of multiple feed-forward parts instead of one. A second (less effective) way to keep the problems associated with the curse of dimensionality to a minimum, is to choose a B-spline distribution with as few B-splines as possible. Several methods were considered for designing a parsimonious LFFC.

Training a parsimonious LFFC is not straightforward. When all BSNs are trained simultaneously, it is not guaranteed that each BSN learns the feed-forward signal it is intended to learn. Therefore, we proposed to train such an LFFC successively, i.e., by selecting a series of training motions by which the BSNs are trained consecutively. However, this so-called *strategic training* approach is rather heuristic in nature.

In chapter 5, design procedures for LFFC were formulated and applied in real world experiments to two plants:

- *LiMMS.* The LiMMS was controlled by a time-indexed LFFC as well as by an LFFC. In the first series of experiments using the time-indexed LFFC, the stability criteria were validated. The results of the experiments were similar to the results of the simulations, i.e. the value of $d_{min}$ is accurate, while the maximum learning rate is conservative. Next, the B-spline distribution of the time-indexed LFFC was optimised. Finally, a parsimonious LFFC was designed and evaluated. The feed-forward part consisted of 3 BSNs, compensating respectively the inertia, the friction and the cogging. After performing strategic training experiments, the parsimonious LFFC obtained a small tracking error for a wide range of

reference motions. However, for repetitive motions the time-indexed LFFC still outperformed the LFFC.

– *Control Loading System (CLS).* In the CLS, the LFFC has to compensate for friction and cogging. Because the reference velocity is accurately tracked at the expense of relatively large positional errors, cogging could not be compensated by a feed-forward controller. Therefore, we compensated cogging by means of a Learning FeedBack Controller (LFBC); friction was compensated with a standard path-indexed feed-forward BSN. After performing strategic training motions, the LFFC / LFBC showed to be able to compensate for cogging and partly for friction.

## 6.2 Conclusions

The feed-forward part of the LFFC has been implemented as a BSN. Experiments and simulations show that the BSN offers:

– *Low computational costs.* Calculating the output of a BSN and adapting the weights are performed in a small number of calculations. In case of the LiMMS, the LFFC was implemented on a DSP. The LFFC consisted of 3 BSNs that were trained during control. In spite of this, the control loop could be calculated at a frequency of 1600 [Hz].

– *Fast convergence.* Learning generally converges after performing the reference motions less than 15 times.

– *Good generalising ability.* Both the parsimonious LFFC for the LiMMS and the CLS (chapter 5) were able to obtain a high tracking accuracy for a range of motions the LFFC was not trained for. The LFFC can thus be trained by presenting a small number of training motions that are characteristic for the reference motions which are to be tracked.

On the other hand the BSN has a number of disadvantages:

– *Problems associated with the curse of dimensionality.* The number of weights in a BSN depends on the dimension of the input in an exponential way. A large number of weights requires a large amount of computer memory and causes a poor learning behaviour. It is therefore important to keep the input dimension of the BSN(s) in an LFFC low.

– *Basis functions are placed in a lattice structure.* Because multi-dimensional B-splines are formed by the tensor product of 1-dimensional B-splines, the density of the multi-dimensional splines is the same over the whole input space. This implies that special measures are needed to guarantee convergence of the learning process and at the same time avoid a sparse lattice, which is unattractive from a function approximation perspective.

From the above we may conclude that BSNs are suited for real-time control when the input dimensions are kept low.

The inputs of the feed-forward part depend on the type of reference motion that has to be performed. In case of repetitive reference motions, the periodic motion time should preferably be used as an input, i.e., time-indexed LFFC should be applied. The time-indexed LFFC can be seen as a variant of ILC / RC, in which the feedback controller is used as a learning filter and that is stabilised by replacing the memory loop by a BSN. The frequency contents of the feed-forward signal that is stored in the BSN, are determined by the width of the B-splines. A convergence analysis of time-indexed LFFC applied to a SISO LTI plant shows that learning converges if:

– *The width of the support of the B-splines is larger than* $d_{min}$. The value of $d_{mia}$ is given by:

$$d_{min} = \frac{2\pi}{\omega_1}[\mathrm{rad\,s^{-1}}] \tag{6.1}$$

in which $\omega_1$ is the frequency at which the phase $\varphi_1$ of the negative complementary sensitivity function, $-T$ , is as follows:

$$\varphi_1 = \arccos\left(-0.0147 \frac{\left|-T\left(j\omega\right)\right|_\infty}{\min\limits_{\{\omega\in\mathbb{R}|\cos(\varphi)\leq 0\}} \left|-T(j\omega)\right|}\right) \tag{6.2}$$

– *The learning rate satisfies*:

$$\gamma \leq \frac{2}{\left|-T\left(j\omega\right)\right|_\infty} \tag{6.3}$$

The above conditions do not require exact knowledge of the high frequency dynamics of the controlled plant. Therefore, either a process model that describes the low-frequency dynamics well or a measured frequency response is sufficient to

design a stable time-indexed LFFC. The stability conditions were validated by means of experiments and simulations, both yielding the same result: the value of $d_{min}$ is accurate also for a non-linear plant, while the maximum learning rate is conservative. However, the conservative value of the learning rate does not lead to extensive training periods. Learning converges after performing a reference motion a limited number of times (typically 10 to 15).

In case of random reference motions, path-indexed LFFC should be applied, i.e., the reference signal and derivatives / integrals thereof should be selected as inputs for the feed-forward part. Whether or not a plant is controllable by means of LFFC can be determined by expressing the plant dynamics in a state space representation of the form:

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x}) &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{A}_1(\boldsymbol{x}) & \boldsymbol{A}_2(\boldsymbol{x}) \cdots \boldsymbol{A}_n(\boldsymbol{x}) \end{bmatrix}, \\ \boldsymbol{B}(\boldsymbol{x}) &= \begin{bmatrix} \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{B}_n(\boldsymbol{x}) \end{bmatrix}^T, \\ \boldsymbol{C} &= \begin{bmatrix} \boldsymbol{C}_1 & \cdots & \boldsymbol{C}_n \end{bmatrix} \end{aligned} \tag{6.4}$$

A plant can be controlled by means of LFFC, when $\boldsymbol{C}$ and $\boldsymbol{B}_n(\boldsymbol{x})$ fulfil the following conditions:

1. $\boldsymbol{C}$ may contain only one submatrix, $\boldsymbol{C}_j$, that is unequal to zero.
2. $\boldsymbol{C}_j$ must be invertible.
3. $\boldsymbol{B}_n(\boldsymbol{x})$ must be invertible.

Which derivatives / integrals should be chosen, can be determined on the basis of the above state space representation of the plant dynamics and the disturbances. For the state space representation, (qualitative) structural knowledge of the plant and the disturbances suffices; no numerical values of parameters are required.

We propose to stabilise path-indexed LFFC by filtering the learning signal in such way that signals that have a frequency content larger than $d_{min}^{-1}$ [Hz] are removed. The filter can be implemented as a time-indexed BSN that has B-splines of which the support is $2d_{min}$ [s]. Simulations show that learning converges when this type of regularisation is added to a BSN that has a B-spline distribution that otherwise causes learning to diverge.

The feed-forward part of a path-indexed LFFC should be implemented as multiple, additive BSNs that each have a low input dimension. This implementation is known as a parsimonious LFFC and keeps the problems associated with the curse of dimensionality to a minimum. The feed-forward parts and the B-spline distributions of a parsimonious LFFC can be determined in the following ways:

- *Using prior knowledge of the plant dynamics and the disturbances.* By writing the plant dynamics and the disturbances in a state space form, the feed-forward parts and their inputs follow in a straightforward way. The B-spline distribution follows from the expected shape of the terms in the state space model.
- *Iterative use of empirical modelling techniques.* Empirical modelling techniques are able to construct a model of a plant, on the basis of observed input-output data, without utilising any prior knowledge of the dynamics of the plant. A problem is that the "correct" input-output data is not immediately available in LFFC. However, it can be approximated as the total steering signal (i.e., feed-forward plus feedback) of a trained LFFC. Hence, by iteratively applying empirical modelling techniques, the appropriate LFFC can be designed. In this research, two empirical modelling techniques are considered for this purpose:
  - *ASMOD*, which can be used to determine the feed-forward parts as well as the B-spline distributions.
  - *Fuzzy clustering*, which is able to determine the B-spline distributions.

  Both ASMOD and fuzzy clustering are computationally intensive, but can in principle be used to obtain a parsimonious LFFC. Therefore, empirical modelling techniques should only be used when the manual selection of the parsimonious LFFC does not give the desired result.
- *Mixed way.* If the designer has partial knowledge of the plant dynamics and the disturbances, this knowledge can be used to create an initial parsimonious LFFC, which is subsequently improved by iteratively applying empirical modelling techniques.

For a parsimonious LFFC, a series of well-chosen training motions should be selected. In each training motion, only one BSN is trained, while the weights of the other BSNs are kept constant. The training motions should be selected in such way that they make the desired feed-forward signal of one of the *untrained* BSNs dominant. In this way, one can achieve that each BSN indeed approximates the function for which it was included in the LFFC. No more systematic approach to the selection of the training motions is yet available.

It is important to verify whether the above training procedure has resulted in an appropriate LFFC. When one BSN approximates an incorrect feed-forward signal, all BSNs that are trained next will also approximate an erroneous function. An LFFC with a poor generalising ability will result. Such an LFFC is able to accurately perform the training motions, but causes large tracking errors for all other reference motions. Hence, a trained path-indexed LFFC needs to be validated with a set of test reference motions.

An LFFC for a well-conditioned plant can be designed as follows.

1. *Design a feedback controller.* The feedback controller compensates for random disturbances and generates a learning signal for the feed-forward part. Since $-T$ depends on the feedback controller, this design step determines the value of $d_{min}$ (see above). When the value $d_{min}$ is too large, redesigning the feedback controller offers a solution.
2. *Select the inputs of the feed-forward part.* In case of repetitive motions, the periodic motion time should preferably be used as the input of the feed-forward part. For random motions, the inputs should consist of the reference position and eventually derivatives/integrals thereof. The inputs can be selected on the basis of a structurally correct state-space description of the plant (see above).
3. *Choose the structure of the feed-forward part.* To prevent problems associated with the curse of dimensionality, the path-indexed LFFC should be implemented in a parsimonious way, i.e. by means of multiple low-dimensional BSNs instead of one high-dimensional BSN. The parsimonious LFFC can be designed on the basis of prior knowledge of the plant dynamics and the disturbances, or in an automated way by using empirical modelling techniques (see above).
4. *Determine the B-spline distribution.* Knowledge of the desired feed-forward signal(s) of the BSN(s) in the LFFC, can be used to determine the B-spline distribution.
5. *Set the learning rate.* The learning rate should be chosen small (close to 0) when the plant is subject to considerable noise. Otherwise, a larger learning rate can be chosen.
6. *Specify training motions (parsimonious LFFC only).* The reference motions should be chosen such that the desired feed-forward signal of one of the untrained BSNs becomes dominant. Only the corresponding BSN is trained, the weights of the other BSNs are kept constant by setting the learning rate of these networks to zero.

Hence, the LFFC can largely be designed on the basis of qualitative knowledge of the plant and the disturbances. Only for convergence guarantees, a quantitative

model of the low-frequency dynamics of the controlled plant is needed, which may be obtained by means of identification.

An overall result of this work is that LFFC appears to be an attractive approach for controlling electromechanical motion systems that are subject to reproducible static input disturbances.

# A Neural Networks & Fuzzy Logic

## A.1 Multi Layer Perceptron Neural Network

### A.1.1 Neurons

The basic element of the Multi Layer Perceptron (MLP) neural network, is the artificial neuron. An artificial neuron, referred to as neuron, is a unit that performs a simple mathematical operation on its inputs. In figure A.1, the neuron is graphically presented.

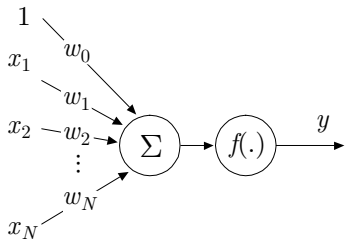



*Figure A.1: Artificial neuron*

The input, $\boldsymbol{x}$, of the neuron consists of the variables $x_1,...,x_N$ and a so called bias term, which is equal to 1:

$$\boldsymbol{x} = \begin{bmatrix} 1 & x_1 & x_2 & ... & x_N \end{bmatrix}^T \in \mathbb{R}^{N+1} \tag{A.1}$$

Each of the input values is multiplied by a weight, $w_i$, after which the results are added. On the result, a simple mathematical function, $f(\cdot)$, is performed. The calculations the neuron performs are thus given by (A.2):

$$y = f\left(\sum_{i=1}^{N} x_i w_i + w_0\right) \tag{A.2}$$

Numerous choices for the functions $f(\cdot)$ exist. Frequently used implementations are the Sigmoid function (A.3) (figure A.2a)

$$f(u) = \frac{1}{1+e^{-u}} \tag{A.3}$$

the hyperbolic tangent, $f(u)=\tanh(u)$ (figure A.2b) and the linear function, $f(u)=u$.

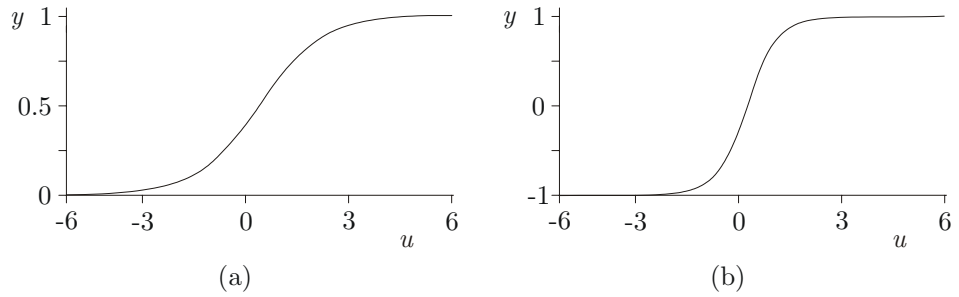



*Figure A.2: a) Sigmoid function*
*b) Hyperbolic tangent function*

## A.1.2 MLP Network

The MLP neural network, in the following referred to as MLP, consists of a number of neurons ordered in layers. In figure A.3 an MLP is shown that consists of 3 layers of neurons (for sake of simplicity, the bias terms and the weights are not shown).

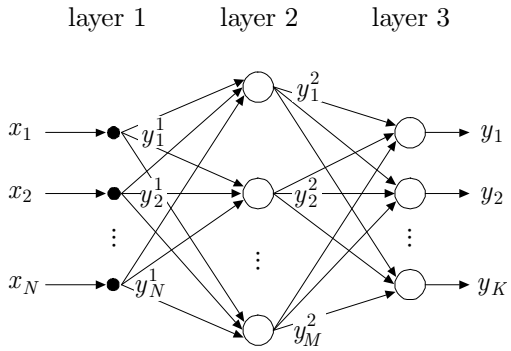



*Figure A.3: Multi Layer Perceptron Network*

Note that a neuron in layer $n$ has connections to each neuron in layer $n$+1. The neurons in layer 1 simply pass on the inputs of the MLP to all the neurons in the second layer.

$$y_n^1 = x_n \tag{A.4}$$

The neurons in the second layer, the so called hidden layer, perform a function, $f(\cdot)$, which is of the Sigmoid or the hyperbolic tangent type:

$$y_m^2 = f\left(\sum_{n=1}^{N} w_{nm}^1 y_n^1 + w_{0m}^1\right) \tag{A.5}$$

Where, $w_{nm}^1$ is the weight of the connections between neuron $n$ in the first layer and neuron $m$ in the second. Finally, the neurons in the third layer perform a function, $g(\cdot)$, which is usually of the linear type:

$$y_k = g\left(\sum_{m=1}^{M} w_{mk}^2 y_m^2 + w_{0k}^2\right) \tag{A.6}$$

Where, $w_{mk}^2$ , is weight of the connections between the neuron $j$ in the second layer and neuron $k$ in the third. Substituting (A.4) and (A.5) in (A.6) yields the following output of the MLP (for $g(\cdot)$ linear):

$$y_k = \sum_{m=1}^{M} w_{mk}^2 f\left(\sum_{n=1}^{N} w_{nm}^1 x_n^1 + w_{0m}^1\right) + w_{0k}^2 \tag{A.7}$$

More hidden layers can be added to the MLP. However, it has been shown that an MLP with one hidden layer can approximate any continuous function [Hornik *et al.*, 1989].

### A.1.3 Training the MLP Neural Network

Training the MLP network is performed by adapting the weights of the connections according to the back propagation mechanism [Rumelhart *et al.*, 1986; Haykin, 1994]. The weights are adapted in such way that a cost function, $J$, is minimised. A common choice for $J$ is the sum of the squared approximation error over all data:

$$J = \frac{1}{2}\sum_{j}\sum_{k=1}^{K}\left(y_{k,j}^{d} - y_{k,j}\right)^2 \tag{A.8}$$

Where $y_{k,j}^{d}$ is the desired output of neuron $k$, for the input $\boldsymbol{x}_j$, and $y_{k,j}$ is the actual output of the MLP. For small adaptations of the weights we may assume that

$$\frac{\partial J}{\partial w} \approx \frac{\Delta J}{\Delta w} \tag{A.9}$$

Which means that,

$$\Delta J \approx \frac{\partial J}{\partial w}\Delta w \tag{A.10}$$

The cost function will decrease, when we choose the adaptation of the weights according to (A.11),

$$\begin{aligned}
\Delta w &= -\gamma\frac{\partial J}{\partial w} \\
&= -\gamma\frac{\partial\left(\frac{1}{2}\sum_{j}\sum_{k=1}^{K}\left(y_{k,j}^{d} - y_{k,j}\right)^2\right)}{\partial w} \\
&= \gamma\sum_{j}\sum_{k=1}^{K}\left(y_{k,j}^{d} - y_{k,j}\right)\frac{\partial y_{k,j}}{\partial w}
\end{aligned} \tag{A.11}$$

where $\gamma$ is the learning rate. This can be verified as follows:

$$\Delta J \approx \frac{\partial J}{\partial w}\Delta w = \frac{\partial J}{\partial w}\left(-\gamma\frac{\partial J}{\partial w}\right) = -\gamma\left(\frac{\partial J}{\partial w}\right)^2 < 0 \tag{A.12}$$

The adaptation of the weight of a connection between the neuron $a$ in the hidden layer and output $b$, $w^2_{ab}$, is given by:

$$\begin{aligned}\Delta w^2_{ab} &= \gamma\sum_j\sum_{k=1}^{K}\left(y^d_{k,j}-y_{k,j}\right)\frac{\partial\left(\sum_{m=1}^{M}w^2_{mk}f\left(\sum_{n=1}^{N}w^1_{nm}x^1_n+w^1_{0m}\right)+w^2_{0k}\right)}{\partial w^2_{ab}}\\ &= \gamma\sum_j\left(y^d_{b,j}-y_{b,i}\right)\frac{f\left(\sum_{n=1}^{N}w^1_{na}x^1_n+w^1_{0a}\right)}{\partial w^2_{ab}}\end{aligned} \tag{A.13}$$

The adaptation of the weights of the connections between the input and the hidden layer, is more complicated. For weight $w^1_{cd}$ the adaptation is given by:

$$\begin{aligned}\Delta w^1_{cd} &= \gamma\sum_j\sum_{k=1}^{K}\left(y^d_{k,j}-y_{k,j}\right)\frac{\partial\left(\sum_{m=1}^{M}w^2_{mk}f\left(\sum_{n=1}^{N}w^1_{nm}x^1_n+w^1_{0m}\right)+w^2_{0k}\right)}{\partial w^1_{cd}}\\ &= \gamma\sum_j\sum_{k=1}^{K}\left(y^d_{k,j}-y_{k,j}\right)\frac{\partial w^2_{dk}f\left(\sum_{n=1}^{N}w^1_{nd}x^1_n+w^1_{0d}\right)}{\partial w^1_{cd}}\\ &= \gamma\sum_j\sum_{k=1}^{K}\left(y^d_{k,j}-y_{k,j}\right)w^2_{dk}\frac{\partial f\left(\sum_{n=1}^{N}w^1_{nd}x^1_n+w^1_{0d}\right)}{\partial\left(\sum_{n=1}^{N}w^1_{nd}x^1_n+w^1_{0d}\right)}\frac{\partial\left(\sum_{n=1}^{N}w^1_{nd}x^1_n+w^1_{0d}\right)}{\partial w^1_{cd}}\\ &= \gamma\sum_j\sum_{k=1}^{K}\left(y^d_{k,j}-y_{k,j}\right)w^2_{dk}f'\left(\sum_{n=1}^{N}w^1_{nd}x^1_n+w^1_{0d}\right)x^1_c\end{aligned} \tag{A.14}$$

# A.2 B-spline Network

## A.2.1 B-splines

A B-Spline Network (BSN) uses basis functions, known as B-splines, for approximation. An $n$-th order 1-dimensional B-spline consists of piece-wise polynomial functions of order $n$-1 (figure A.4). The height of a B-spline is known as the membership and is denoted as $\mu \in [0,1]$.

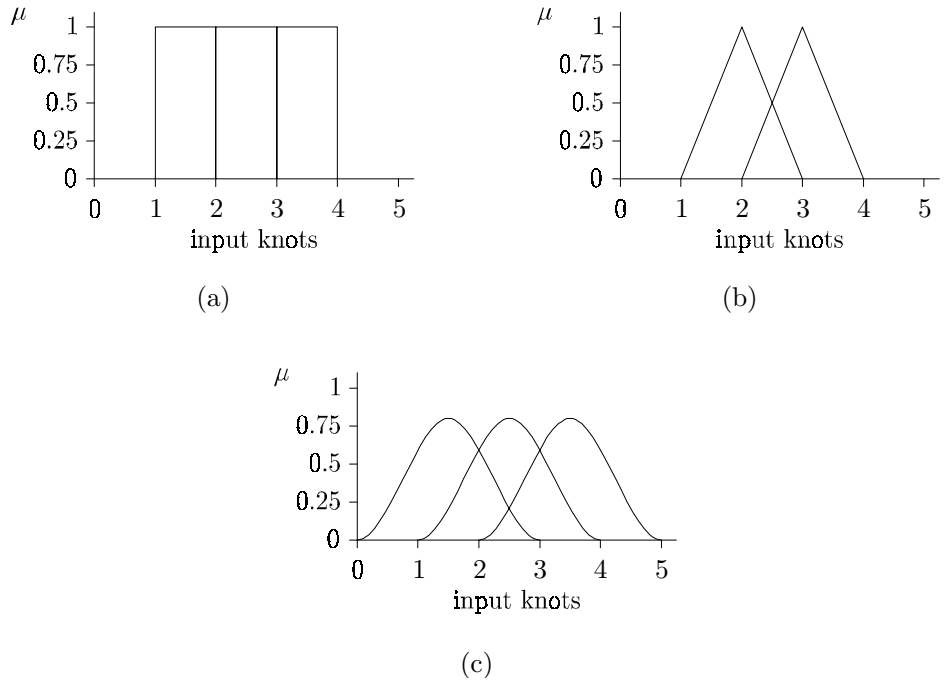



*Figure A.4* *a): 1st order B-splines*
*b): 2nd order B-splines*
*c): 3rd order B-splines*

The shape and the position of the B-splines is such that at each point in the input space, the sum of the membership of the B-splines equals 1. The position of the B-splines on the input domain is determined by a so-called knot vector:

$$\lambda = \left[\lambda_0 \ \lambda_1 \ ... \ \lambda_{N_k}\right] \in \mathbb{R}^{N_k} \tag{A.15}$$

The membership of the $k$-th B-spline of order $n$ is given by:

$$\mu_i^n(x) = \left(\frac{x - \lambda_{i-n}}{\lambda_{i-1} - \lambda_{i-n}}\right)\mu_{i-1}^{n-1}(x) + \left(\frac{\lambda_i - x}{\lambda_i - \lambda_{i-n+1}}\right)\mu_i^{n-1}(x) \tag{A.16}$$

$$\mu_i^1(x) = \begin{cases} 1 & \text{if } \lambda_{i-1} \le x < \lambda_i \\ 0 & \text{otherwise} \end{cases} \tag{A.17}$$

In this research, we only consider 2nd order B-splines. The membership of the $i$-th 2nd order B-spline will be denoted by $\mu_i(x)$. A B-spline has a membership that is larger than zero only on a compact part of the input space. This part of the input space is known as the support of a B-spline. Multi-dimensional B-splines are constructed by the tensor product of 1-dimensional B-splines (figure A.5).

$$\mu_{i_1,i_2}(x_1, x_2) = \mu_{i_1}(x_1)\,\mu_{i_2}(x_2) \tag{A.18}$$

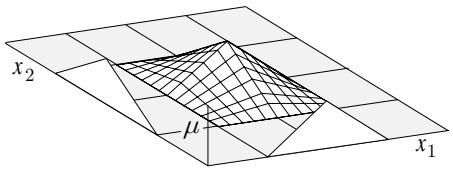
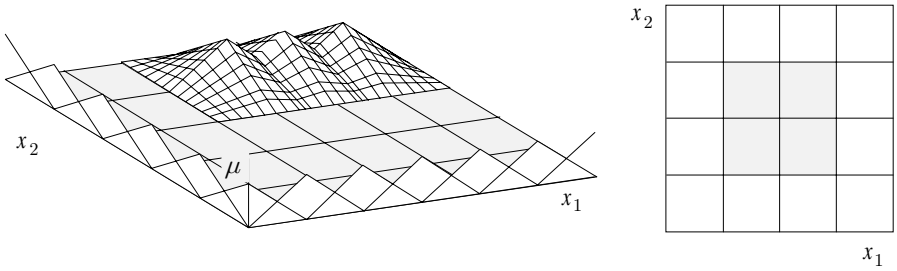



*Figure A.5: 2-dimensional B-spline*

The supports of the B-splines form a grid (figure A.6a).




(a) (b)

*Figure A.6: a) 2-dimensional B-splines*
*b) Grid formed by the supports of the B-splines*

## A.2.2 B-spline Network

The output of a BSN is a weighted sum of the B-spline evaluations. In figure A.7 this is shown in a form similar to the MLP.

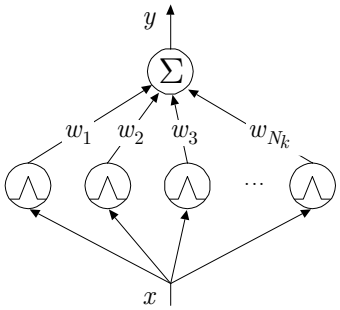



*Figure A.7: MLP representation of a BSN*

In the 1-dimensional case the output of the BSN is given by:

$$y = \sum_{i=1}^{N_k} \mu_i(x) \cdot w_i \tag{A.19}$$

where,

| | |
|---|---|
| $x$: | the input, $x \in \mathbb{R}$ ; |
| $y$: | the output of the network; |
| $N_k$: | number of B-splines; |
| $\mu_i(x)$: | membership of the $i$-th B-spline; |
| $w_i$: | network weight of the $i$-th B-spline; |

An example of an input-output mapping is given in figure A.8. Please observe that the BSN can also be regarded as an adaptive fuzzy logic controller. It is equal to a Mamdani type fuzzy logic controller [Lee, 1990] which has fuzzy singletons as consequence sets. Another way to look at the BSN, is as a look-up table with an interpolation mechanism.

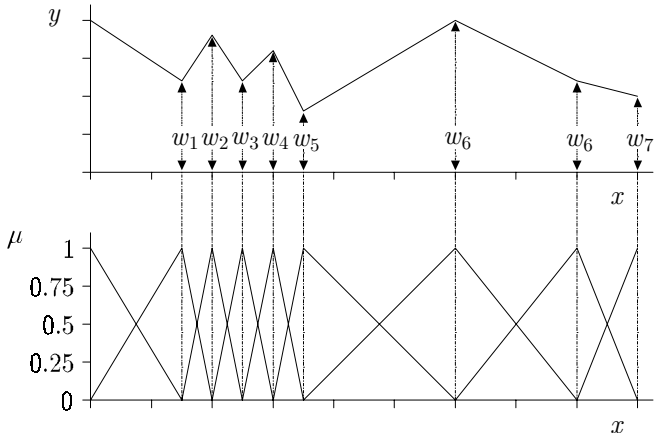



*Figure A.8: $2^{nd}$ order B-spline network mapping*

### A.2.3 Training the BSN

The network is trained by adapting weights, not the basis functions. This can either be done after each sampling interval, which is known as on-line learning, or after a motion has been completed, known as off-line learning. In the on-line case, the cost function, $J$, that is minimised is the squared approximation error (A.20):

$$J = \frac{1}{2}(y_d - y)^2 \tag{A.20}$$

Where $y_d$ is the desired output of the BSN. Applying the back propagation rule (A.11) yields,

$$\Delta w_i = -\gamma \frac{\partial J}{\partial w_i} = \gamma (y_d - y) \frac{\partial y}{\partial w_i} \tag{A.21}$$

Substituting (A.19) in (A.21) gives:

$$\begin{aligned} \Delta w_i &= \gamma (y_d - y) \frac{\partial \sum_{j=1}^{N_k} \mu_j(\boldsymbol{x}) \cdot w_j}{\partial w_i} \\ &= \gamma (y_d - y)\, \mu_i(\boldsymbol{x}) \end{aligned} \tag{A.22}$$

with, $\Delta w_i$: the adaptation of the weight of the $i$-th B-spline;
$\gamma$ : the learning rate, $0 < \gamma \leq 1$;

In the off-line case the cost function is given by:

$$J = \frac{1}{2}\sum_{j}\left(y_{d,j} - y_j\right)^2 \tag{A.23}$$

Where $y_{d,i}$ is the desired output for input $\boldsymbol{x}_i$ and $y_i$ is the output of the BSN. Performing the same calculations as above gives the following adaptation of the weights:

$$\Delta w_i = \gamma\sum_{j}\left(y_{d,j} - y_j\right)\mu_i^n(\boldsymbol{x}_j) \tag{A.24}$$

In order to prevent large weight adaptations, we propose to normalise (A.24):

$$\Delta w_i = \gamma\frac{\sum_{j}\left(y_{d,j} - y_j\right)\mu_i^n(\boldsymbol{x}_j)}{\sum_{j}\mu_i^n(\boldsymbol{x}_j)} \tag{A.25}$$

Learning rule (A.25) implies that in LFFC the weights are updated after a reference motion has been completed. Since (in theory) the reference motion may have infinite time length, implementing the off-line learning rule as (A.25) is undesired. Therefore, we propose to implement the off-line learning mechanism in the following way:

**Algorithm A.1 (Off-line learning mechanism)**

1. Present inputs $x_j$ and a learning signal $y_{d,j} - y_j$ to the BSN ($j$=1,2,….). While the membership of B-spline $i$ for input $x_j$ $\mu_i(\boldsymbol{x}_j) > 0$, calculate the adaptation of $w_i$ according to (A.25).
2. At the point in time where $\mu_i(\boldsymbol{x}_j) = 0$, $w_i$ is adapted, $w_i = w_i + \Delta w_i$.

In figure A.9 an example is given. For $x_{1..5}$, the memberships of B-spline 1 and 2 are unequal to 0. At $x_6$, the membership of B-spline 1 equals 0, which means that it no longer contributes to the output of the BSN. Therefore, we choose to update the weight of B-spline 1.

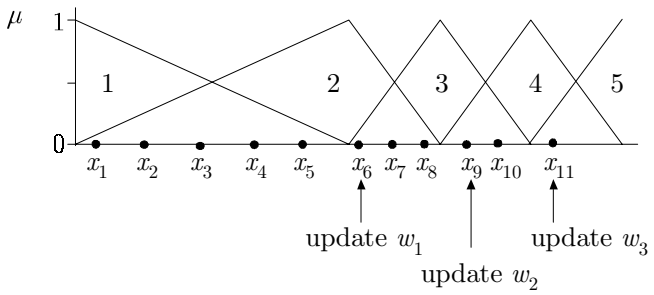



*Figure A.9: Implementation of the off-line learning mechanism*

# A.3 Fuzzy Logic

## A.3.1 Introduction

In the classical set theory, an object is either a member of a set or a non-member. For some types of objects, classification problems may occur. This can be easily shown by the classification of people in the sets "bald" and "not bald". A person with 100.000 hairs on his head is classified in the set "not bald". If one hair is torn out, the person will still be classified in the set "not bald". A person does not become "bald" if he loses just one of his hairs. This process can be repeated until the man has just one hair left. If the last hair is torn out the man suddenly becomes "bald". This way of classification does not correspond to the human intuition. In normal life a man with 10 hairs on his head is considered bald.

The reason for the second classification to fail, is that the sets "bald" and "not bald" do not have sharp boundaries. A person does not become bald after he has lost one of his hairs, but gradually becomes bald after he looses more and more hairs. To be able to describe this kind of phenomenons, Zadeh invented the fuzzy set theory [Zadeh, 1973; Lee, 1990]. In the fuzzy set theory an object can partially belong to a set. A man with 5000 hairs on his head could, for example, be a member of the set "bald" for 50% and a member of the set "not bald" for another 50%.

## A.3.2 Fuzzy Sets

To deal with fuzziness the grade of membership of membership was introduced.

**Definition A.1 (Grade of membership)** *The grade of membership is the relative degree to which an element x is a member of set F and is denoted as* $\mu_F \in [0,1]$.

The domain on which fuzzy sets are defined is called the universe of discourse.

**Definition A.2 (Universe of discourse)** *The universe of discourse X, is the collection of all objects x, suitable for classification.*

Using definition A.1 and A.2 we are now able to give a definition of a fuzzy set.

**Definition A.3 (Fuzzy set)** *A fuzzy set F on X is a collection over all* $x \in X$ *of x and* $\mu_F(x)$.

$$F = \int_X \mu_F(x)/(x) \qquad \text{in which } \mu_F(x) \in [0,1] \tag{A.26}$$

In (A.26), $\int_X$ must be interpreted as the collection over $X$ and $\mu_F(x)/(x)$ as a pair $\left(x, \mu_F(x)\right)$. In figure A.10 an example of a trapezoidal fuzzy set is presented. A special type of fuzzy set, is the so called fuzzy singleton.

**Definition A.4 (Fuzzy singleton)** *A fuzzy singleton is a fuzzy set that has only one member with* $\mu_F \neq 0$.

$$F = \int_X \delta(a - x)/(x) \tag{A.27}$$

In which $\delta$ is the Dirac function.

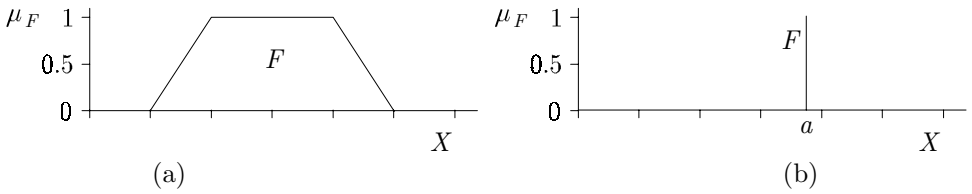



*Figure A.10: a) Trapezoidal fuzzy set F*
*b) Fuzzy singleton F*

### A.3.3 Operations on Fuzzy Sets

Several operations on fuzzy sets have been defined, which can be roughly be divided in the following classes. The class of T-norm operations , denoted by *, is characterised by the fact that the resulting fuzzy set is generally smaller than or equal to each of the operands. Examples of the T-norm are the fuzzy intersection and the algebraic product, respectively:

$$\mu_A(x) * \mu_B(x) = \min\Big(\mu_A(x), \mu_B(x)\Big) \tag{A.28}$$

$$\mu_A(x) * \mu_B(x) = \mu_A(x)\mu_B(x) \tag{A.29}$$

The T-conorm operations, denoted by $\dot{+}$, yield a fuzzy set that is generally larger than or equal to each of the operands. Examples of the T-conorm are fuzzy union and the bounded sum respectively:

$$\mu_A(x) \dot{+} \mu_B(x) = \max\Big(\mu_A(x), \mu_B(x)\Big) \tag{A.30}$$

$$\mu_A(x) \dot{+} \mu_B(x) = \min\Big(1, \mu_A(x) + \mu_B(x)\Big) \tag{A.31}$$

Finally, we would like to define the sup-$*$ composition. When $C$ and $D$ are fuzzy sets in respectively $X$ and $X{\times}Y$, the sup-$*$ composition of $C$ and $D$, denoted as $C \circ D$, is a fuzzy set in $Y$, with the following membership function:

$$\mu_{C\circ D}(y) = \sup_{\forall x \in X} \mu_C(x) * \mu_D(x, y) \tag{A.32}$$

### A.3.4 Mamdani Fuzzy Controller

Fuzzy logic controllers operate in an environment, in which the variables have a real, non fuzzy, value. This means that, to be able to apply a fuzzy logic controller, the values of the inputs need to be transformed to fuzzy sets. This process is known as fuzzification. Fuzzification is often performed by constructing a fuzzy singleton given by:

$$A' = \int_X \delta(x - a)/(x) \tag{A.33}$$

where,

$a$: the value of the input variable, $x$.
$A'$: the fuzzy input set, representing the value of input variable

In the Mamdani fuzzy controller, the input-output behaviour is specified by a number of IF-THEN rules:

$$\text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } B_i \qquad (A.34)$$

In which,

$A_i$: the fuzzy antecedent set, representing input values of the controller
$B_i$: the fuzzy consequence set, representing the actions that should be taken when these inputs occur

The reasoning mechanism, according to which the output of the fuzzy logic controller is calculated, can be described by the generalised modus ponens (A.35).

$$\begin{array}{ll} \text{premise:} & x \text{ is } A' \\ \text{implication:} & \text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } B_i \\ \text{consequence:} & y \text{ is } B' \end{array} \qquad (A.35)$$

In which,

$y$: the output variable;
$B'$: fuzzy output set, representing the value of the output variable.

The implication "IF $x$ is $A_i$ THEN $y$ is $B_i$" can be defined as a relation between fuzzy set $A_i$ and $B_i$. This relation is referred to as the fuzzy implication. A fuzzy implication is a special kind of fuzzy relation.

**Definition A.5 (fuzzy relation)** *Let $X$ and $Y$ be two universes of discourse. A fuzzy relation $R$ is a fuzzy set in the product space $X \times Y$ with membership function $\mu_R(x,y)$, where $x \in X$ and $y \in Y$.*

**Definition A.6 (fuzzy implication)** *Let $A_i$ be a fuzzy set in $X$ and $B_i$ be a fuzzy set in $Y$. The fuzzy implication, denoted as $A_i \rightarrow B_i$, is a fuzzy relation in $X \times Y$ with the following membership functions:*

$$\mu_{A_i \rightarrow B_i}(x,y) = \mu_{A_i}(x) * \mu_{B_i}(y) \qquad (A.36)$$

In the following we will denote $A_i \rightarrow B_i$ as $R_i$. In figure A.11 an example of fuzzy implication (A.36), using the min-operator, is given.

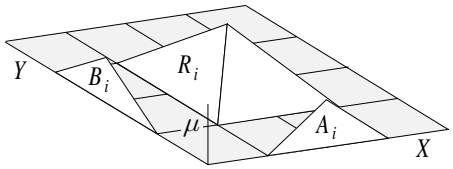



*Figure A.11: Fuzzy implication, min-operator*

The fuzzy output set $B_i'$ is calculated from $A'$ and the fuzzy implication $R_i$. This calculation is performed by the fuzzy inference engine.

**Definition A.7 (fuzzy inference)** *Let $A'$ be a fuzzy input set in X and $R_i$ be a fuzzy implication in X×Y. The fuzzy inference calculates fuzzy output set $B_i'$ on the basis of $A'$ and $R_i$. The membership function of $B_i$' is given by:*

$$B_i' = A' \circ R_i \tag{A.37}$$

which is,

$$B_i' = \int_Y \sup_{\forall x \in X} \left( \mu_{A'}(x) * \mu_{A_i}(x) * \mu_{B_i}(y) \right) / (y) \tag{A.38}$$

The fuzzy inference engine is illustrated in figure A.12 (* is implemented as the min-operator).

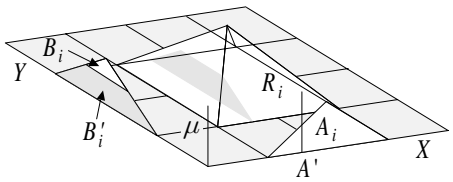



*Figure A.12: Fuzzy inference engine*

So far fuzzy IF-THEN rules had one fuzzy input set. In case of fuzzy IF-THEN rules with more than one fuzzy antecedent set, e.g..:

$$\text{IF } \left(x_1 \text{ is } A_{1,i}\right) \text{ AND } \left(x_2 \text{ is } A_{2,i}\right) \text{ THEN } y \text{ is } B_i \tag{A.39}$$

the AND connective is used. A fuzzy implication with fuzzy antecedent sets $A_{1,i}$ in $X_1$ and $A_{2,i}$ in $X_2$ and fuzzy consequence set $B_i$ in $Y$ is expressed as:

$$R_i = \left(A_{1,i} \text{ AND } A_{2,i}\right) \rightarrow B_i \tag{A.40}$$

$$\mu_{R_i}(x_1, x_2, y) = \mu_{A_{1,i}}(x_1) *_{A_{2,i}} (x_2) * \mu_{B_i}(y) \tag{A.41}$$

The calculation of $B_i$' using the AND operator and fuzzy input sets $A_1$' and $A_2$' can be expressed as follows:

$$B_i' = \left(A_1' \text{ AND } A_2'\right) \circ R_i \tag{A.42}$$

which gives

$$B_i' = \int_Y \sup_{\forall x_1 \in X_1, \forall x_2 \in X_1} \mu_{A_1'}(x_1) * \mu_{A_2'}(x_2) * \mu_{A_{1,i}}(x_1) * \mu_{A_{2,i}}(x_2) * \mu_{B_i}(y)/(y) \tag{A.43}$$

Each of the rules gives one fuzzy output set. In order to calculate the output of the fuzzy controller, all separate output sets, $B_i$', have to be combined in one output set, $B$'. This is done by the ALSO operator, which is implemented as a T-conorm:

$$B' = B_1' \dot{+} B_2' \dot{+} .. B_n' \tag{A.44}$$

Where $n$ is the number of fuzzy relations. Since the fuzzy logic controller operates in a real-valued environment, the fuzzy output set $B$' has to be transformed in a single value, $b$. This process is known as defuzzification. The most frequently used method is the Centre Of Area (COA) defuzzification, which is given by:

$$b = \frac{\int_Y \mu_{B'}(y) y \ dy}{\int_Y \mu_{B'}(y) \ dy} \tag{A.45}$$

In figure A.13, an example is shown where the fuzzy output set $B'$, which is the combination of two fuzzy output sets $B_1'$ and $B_2'$, is converted into a single value, $b$.

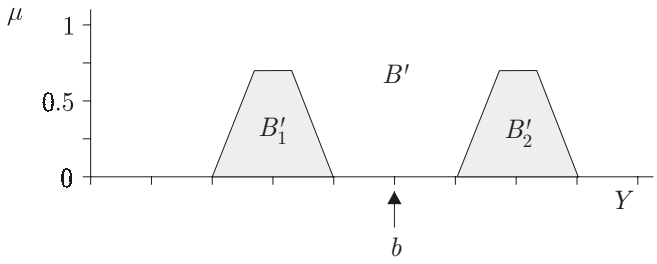



*Figure A.13: Fuzzy output sets*

For special choices of the fuzzy premise and the fuzzy consequence sets, the fuzzy controller is equivalent to a 2nd order BSN. These choices are:

- The fuzzy antecedent sets are triangular. The positions of the fuzzy antecedent sets is such that for each input value, the sum of the memberships equals 1.
- The fuzzy consequence sets are fuzzy singletons.
- The *-operator is implemented as a product.
- The $\dot{+}$ operator is implemented as a bounded sum.

In figure A.14 an example of such a fuzzy controller is shown.

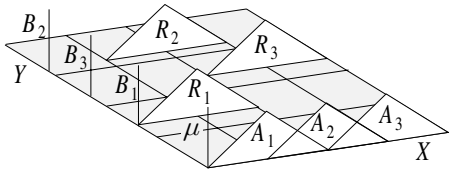



*Figure A.14: Fuzzy controller that is equivalent to a BSN*

The output of a fuzzy controller is calculated as follows. The fuzzy relations are given by:

$$R_i = \int_{X\times Y} \mu_{A_1}(x)\,\delta(y-b_i)/(x,y) \tag{A.46}$$

The fuzzy output sets are determined given,

$$B_i' = A' \circ R_i \tag{A.47}$$

$$B_i' = \int_Y \sup_{\forall x \in X} \left( \delta(x-a)\mu_{A_i}(x)\delta(y-b_i) \right) / (y) \tag{A.48}$$

$$B_i' = \int_Y \mu_{A_i}(a)\delta(y-b_i) / (y) \tag{A.49}$$

Combining the separate output sets $B_i'$ to one output set, $B'$, yields:

$$B' = \int_Y \min\left(1, \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n)\right) / (y) \tag{A.50}$$

Since the sum of the memberships of the fuzzy antecedent sets equals 1 for each $a$, (A.50) can be written as:

$$B' = \int_Y \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n) / (y) \tag{A.51}$$

Defuzzification yields:

$$b = \frac{\int_{-\infty}^{\infty} \left( \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n) \right) y \ dy}{\int_{-\infty}^{\infty} \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n) \ dy} \tag{A.52}$$

As before, the fact that the sum of the memberships of the fuzzy antecedent sets equals 1 for each $a$, can be used to simplify (A.52):

$$\begin{aligned} \int_{-\infty}^{\infty} \left( \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n) \right) \ dy &= \mu_{A_1}(a) + \ldots + \mu_{A_n}(a) \\ &= 1 \end{aligned} \tag{A.53}$$

This gives the following output of the fuzzy controller:

$$b = \int_{-\infty}^{\infty} \left( \mu_{A_1}(a)\delta(y-b_1) + \ldots + \mu_{A_n}(a)\delta(y-b_n) \right) y \ dy \tag{A.54}$$

$$b = \int_{-\infty}^{\infty} \mu_{A_1}(a)b_1 + \ldots + \mu_{A_n}(a)b_n \; dy \tag{A.55}$$

When comparing the output of the fuzzy controller (A.55) to the output of the BSN (A.19), it can be seen that these are equivalent. The fuzzy consequence sets fulfil the role of the weights in the BSN. The BSN can thus be regarded as an adaptive fuzzy controller, of which the fuzzy consequence sets are adapted.

### A.3.5 Takagi-Sugeno Fuzzy Controller

The TS fuzzy model consists of a collection of rules that have the following form:

$$\text{Rule } i\text{:} \quad \begin{array}{l} \text{IF } \left(x_1 \text{ is } A_{1,i}\right) \text{ AND } \cdots \text{ AND } \left(x_m \text{ is } A_{n,i}\right) \\ \text{THEN } y = a_{1,i}x_1 + \cdots + a_{m,i}x_m + b_i \end{array} \tag{A.56}$$

Where, $x_1 \ldots x_m$: the input variables, $x_1 \ldots x_m \in \mathbb{R}$;
$y$: the output variable, $y \in \mathbb{R}$;
$A_{1,i} \ldots A_{m,i}$: fuzzy sets defined on the inputs of the model, also known as fuzzy antecedent sets

First we evaluation $\text{IF } \left(x_1 \text{ is } A_{1,i}\right) \text{ AND } \cdots \text{ AND } \left(x_m \text{ is } A_{m,i}\right)$, which yields a support variable, $\beta_i$:

$$\beta_i = \sup_{\forall x_1 \in X_1, \ldots, \forall x_m \in X_m} \mu_{A_1'}(x_1) * \ldots * \mu_{A_m'}(x_m) * \mu_{A_{1,i}}(x_1) * \ldots \mu_{A_{m,i}}(x_m) \tag{A.57}$$

Next, the output of the fuzzy controller is calculated according to:

$$y = \frac{\sum_i \beta_i \left(a_{1,i}x_1 + \cdots + a_{n,i}x_n + b_i\right)}{\sum_i \beta_i} \tag{A.58}$$

An example of a 1-dimensional TS fuzzy model is shown in figure A.15. In the lower part, the fuzzy antecedent sets are shown. In the upper part, the resulting input-output mapping is presented. It can be seen that the mapping consists of locally linear functions with smooth transitions.

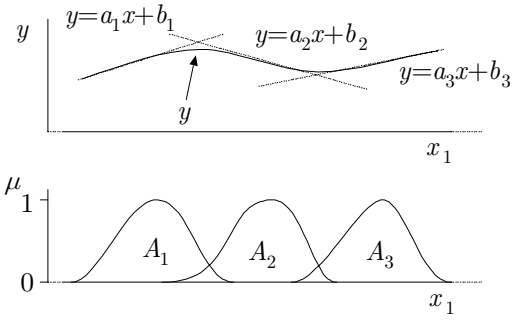



*Figure A.15: Takagi-Sugeno fuzzy model*

# References


Albus, J.S., 1975, A new approach to manipulator control: The cerebellar model controller (CMAC), *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, **97**, 220-227.

Amann, N. and D.H. Owens, 1996, An H-inf Approach to Linear Iterative Learning Control Design, *International Journal of Adaptive Control and Signal Processing*, **10** (6), 767-781.

Amerongen, J. van, 1982, *Adaptive steering of ships: a model- reference approach to improved manoeuvring and economical course keeping*, Ph.D. Thesis, University of Delft, Delft, The Netherlands.

Ananthraman, S. and D.P. Grag, 1993, Training Backpropagation and CMAC Neural Networks for Control of a SCARA Robot, *Engineering Applications of Artificial Intelligence*, **6** (2), 105-115.

Aragwal, M., 1997, A Systematic Classification of Neural-Network-Based Control, *IEEE Control Systems Magazine*, **17** (2), 75-93.

Arif, M., T. Ishihara and H. Inooka, 1999, Iterative Learning Control Using Information Database, *Journal of Intelligent and Robotic Systems*, **25** (1), 27-41.

Arimoto, S., 1998, A Brief History of Iterative Learning Control, Iterative Learning Control: Analysis, Design, Integration and Applications, Kluwer Academic Publishers, 3-7.

Arimoto, S., S. Kawamura and F. Miyazaki, 1984, Bettering operation of robots by learning, *Journal of Robotic Systems*, 123-140.

Armstrong-Helouvry, B., P. Dupont and C.C. De Wit, 1994, A Survey of Models, Analysis Tools and Compensation Methods for the Cotnrol of Machines with Friction, *Automatica*, **30** (7), 1083-1138.

Åström, K.J. and B. Wittenmark, 1989, Adaptive Control, Addison-Wesley.

Babuška, R., 1997,b, *Fuzzy Modelling and Identification*, Ph.D. Thesis, University of Delft, Delft, The Netherlands.

Backer, E., 1995, Computer-assisted Reasoning in Clustering Analysis, Prentice Hall, New York, U.S.A.

Basak, A., 1996, Permanent-magnet DC linear motors, Clarendon, Oxford, UK.

Bellman, R.E., 1961, Adaptive Control Processes, Princeton University Press.

Bezdek, J.C. and S.K. Pal, 1992, Fuzzy Models for Pattern Recognition, New York, U.S.A., IEEE Press.

Bien, Z. and J.X. Xu, 1998, Interative Learning Control: Analysis, Design, Integration and Applications, Kluwer Academic Publishers, Norwell, Massachusetts, U.S.A..

Bishop, C.M., 1991, Improving the Generalisation Properties of Radial Basis Function Neural Networks, *Neural Computation*, **3**, 579-588.

Bossley, K.M., 1997, *Neurofuzzy Modelling Approaches in System Identification*, Ph.D. Thesis, Faculty of Engineering and Applied Science, University of Southampton, U.K.

Bossley, K.M., M. French, E. Rogers, C.J. Harris and M. Brown, 1996, Recent advances in neurofuzzy control system theory and design, Proc. ViCAM, EFAM, Guimarães, Portugal, 117-133.

Bouwhuis, B.R., 1999, *Learning control of a flight simulator stick*, M.Sc. Report 007R99, Control Laboratory, University of Twente, Enschede, The Netherlands

Brown, M. and C.J. Harris, 1994, Neurofuzzy adaptive modelling and control, Prentice Hall, Hemel-Hempstead, UK.

Bruske, J., M. Hansen, L. Riehn and G. Sommer, 1997, Biologically inspired calibration-free adaptive saccade control of a binocular camera-head, *Biological Cybernetics*, **77**, 433-446.

Chen, Y., J.X. Xu and T.H. Lee, 1996, Current iteration tracking error assisted iterative learning control of uncertain nonlinear discrete-time-systems, 35th IEEE Conference on Decisin and Control, Kobe, Japan, 3040-3045.

Chien, C.J. and J.S. Liu, 1996, P-type iterative learning controller for robust output tracking of nonlinear time-varying systems, *International Journal of Control*, **64** (2), 319-334.

Coelingh, H.J., 2000, *Design Support for Motion Control Systems: a Mechatronic Approach*, Ph.D. Thesis, University of Twente, Enschede, The Netherlands.

Craig, J.J., 1989, Introduction to Robotics: Mechanics and Control, Addison-Wesley.

Dean, P., J.E.W. Mayhew, N. Thacker and P.M. Langdon, 1991, Saccade control in a simulated robot camera-head system: neural net architectures for efficient learning of inverse kinematics, *Biological Cybernetics*, **66**, 27-36.

Dibb, S, L Simkin, W.M. Pride and O.C. Ferrell, 1991, Houghton Mifflin Company, Marketing, Boston, USA.

Er, M.J. and K.C. Liew, 1997, Control of Adept One SCARA Robot Using Neural Networks, *IEEE Transactions on Industrial Electronics*, **44** (6),

Francis, B.A. and W.M. Wonham, 1975, The Internal Model Principle for Linear Multivariable Regulators, *Applied Mathematics and Optimizatin*, **2** (2), 175-194.

Girosi, F., M. Jones and T. Poggio, 1995, Regularisation Theory for Neural Networks, *Neural Computation*, **7**, 219-269.

Gomi, H. and M. Kawato, 1993, Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning, *Neural Networks*, **6** (7), 933-946.

Groenhuis, Herman, 1991, *A design tool for electromechanical servo systems*, Ph.D. Thesis, University of Twente, Enschede, The Netherlands.

Gustafson, D.E. and W.C. Kessel, 1979, Fuzzy Clustering with a Fuzzy Covariance Matrix, Proceedings of the IEEE CDC, San Diego, U.S.A., 761-766.

Hara, S., Y. Yamamoto, T. Omata and M. Nakano, 1988, Repetitive control system: A new type servo system for periodic exogenous signals, *IEEE Transactions on Automatic Control*, **33** (7), 402-411.

Haring, M., 1998, *Regularisation applied to learning feed-forward control*, M.Sc. Report 042R98, Control Laboratory, University of Twente, Enschede, The Netherlands.

Harris, C.J., C.G. Hoore and M. Brown, 1993, Intelligent Control: Aspects of Fuzzy Logic and Neural Nets, World Scientific Publishing Co. Pte. Ltd., Singapore.

Haykin, S., 1994, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, Toronto, Canada.

Heinzinger, G., D. Fenwick, B. Paden and F. Miyazaki, 1992, Stability of learning control with disturbances and uncertain initial conditions, *IEEE transactions on Automatic Control*, **37** (1), 110-114.

Hertz, J., A. Krogh and R.G. Palmer, 1991, Introduction to the Theory of Neural Computation, Addison-Wesley.

Hornik, K., M. Stinchcombe and H. White, 1989, Multilayer Feedforward Neurworks Are Universal Approximators, *Neural Networks*, **2**, 359-366.

Horowitz, R., 1993, Learning Control of Robot Manipulators, *Journal of Dynamic Systems, Measurement and Control*, **115** (2), 402-411.

Horowitz, R., W. Messner and J.B. Moore, 1991, Exponential convergence of a learning controller for robot manipulators, *IEEE Transactions on Automatic Control*, **36** (7), 890-894.

Horowitz, R., W.W. Ka, M. Boals and N. Sadegh, 1989, Digital implementation of repetitive controllers for robotic manipulators, Proceedings of the IEEE Conference on Robotics and Automation, 1497-1503.

Hrycej, T., 1997, Neurocontrol: Toward an Industrial Control Methodology, John Wiley & Sons, Inc., New York, USA.

Hunt, K.J., D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, 1992, Neural Networks for Control Systems - A Survey, *Automatica*, **28** (6), 1083-1112.

Idema, L.J., 1996, *Design of Parsimonious Learning Feed-Forward Controllers*, M.Sc. report 051R96, Control Laboratory, University of Twente, Enschede, The Netherlands.

Jacobs, R.A. and M.I. Jordan, 1993, Learning Piecewise Control Strategies in a Modular Neural Network Architecture, *IEEE Transactions on Systems, Man, and Cybernetics*, **23** (2), 337-345.

Jang, J.S.R. and C.T. Sun, 1995, Neuro-Fuzzy Modelling and Control, *The Proceedings of IEEE*, **83**, 378-406.

Jang, J.S.R., C.T. Sun and E. Mizutani, 1997, Neuro-Fuzzy and Soft Computing: A computational approach to learning and machine intelligence, Prentice Hall.

Katic, D.M. and K. Vukobratovic, 1995, Highly Efficient Robot Dynamics Learning by Decomposed Connectionist Feedforward Control Structure, *IEEE Transactions on Systems, Man and Cybernetics*, **25** (1), 145-158.

Kavli, T. and E. Weyer, 1995, On asmod, an algorithm for empirical modelling using spline functions,83-104, Neural network engineering in dynamic control systems, Springer Verlag, Berlin, Germany.

Kavli, T., 1992, Frequency Domain Synthesis of Trajectory Learning Controllers for Robot Manipulators, *Journal of Robotic Systems*, **9** (5), 663-680.

Kavli, T., 1992, *Learning Principles in Dynamic Control*, Ph.D. Thesis, University of Oslo, Olso, Norway.

Kavli, T., 1993, ASMOD: an algorithm for Adaptive Spline Modelling of Observation Data, *International Journal of Control*, **58** (4), 947-968.

Kawamura, S., F. Miyazaki and S. Arimoto, 1988, Realization of robot motions based on learning method, *IEEE Transactions on Systems, Man and Cybernetics*, **18** (1), 125-134.

Kawato, M., 1990, Feedback-Error-Learning Neural Network for Supervised Motor Learning,365-372, Advanced Neural Computers, Amsterdam, The Netherlands, Elsevier Science Publishers B.V.

Kawato, M., K. Furukawa and R. Suzuki, 1987, A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement, *Biological Cybernetics*, **57**, 169-185.

Kim, S.W. and J.J. Lee, 1996, Filtered-Error-Learning Neural Networks for Stable Trajectory Tracking Control of Robot Manipulators, *Mechatronics*, **6** (2), 181-192.

Kraft, L.G. and D.P. Campagna, 1990, A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems, *IEEE Control Systems Magazine*, 36-43.

Landau, I.D., 1979, Adaptive Control: The Model Reference Approach, Marcel Dekker, New York, U.S.A.

Lee, C.C., 1990, Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part 1 + 2, *IEEE Transactions on Systems, Man and Cybernetics*, **20** (2), 404-435.

Lee, H.S. and Z. Bien, 1996, Study on robustness of iterative learning control with non-zero initial error, *International Journal of Control*, **64**, 345-359.

Lee, H.S. and Z. Bien, 1998, Robustness and Convergence of a PD-Type Iterative Learning Controller,39-55, Iterative Learning Control: Analysis, Design, Integration and Applications, Kluwer Academic Publishers.

Lewis, F.L., C.T. Abdallah and D.M. Dawson, 1993, Control of robot manipulators, New York, U.S.A., Macmillan Publishing Company.

Longman, R.W., 1998, Designing Iterative Learning and Repetitive Controllers,107-146, Iterative Learning Control: Analysis, Design, Integration and Applications, Kluwer Academic Publishers.

Longman, R.W., 1998, Some Perspectives on Iterative Learning Control, Iterative Learning Control Workshop and Roundtable, IEEE C DC, Tampa, FL, USA, 29-33.

Luenen, T.C. van, 1993, *Neural Networks for Control: On Knowledge Representation and Learning*, Ph.D. Thesis, Department of Electrical Engineering, Univerity of Twente, Enschede, The Netherlands.

Mareels, I. and J.W. Polderman, 1996, Adaptive Systems: an Introduction, Boston, USA, Birkhaeuser.

Messner, W., R. Horowitz, W.W. Kao and W. Boals, 1991, A new adaptive learning rule, *IEEE trans. on Automatic Control*, **36** (2), 188-197.

Miyamoto, H., M. Kawato, T. Setoyama and R. Suzuki, 1988, Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator, *Neural Networks*, **1**, 251-265.

Moore, K.L., 1992,a, Iterative learning control for deterministic systems, Springer Verlag, Berlin, Germany.

Moore, K.L., 1999, Iterative Learning Control: an Expository Overview,http://www.engineering.usu.edu/ece/faculty/moorek/mooreweb.html,

Moore, K.L., M. Dahleh and S.P. Bhattacharyya, 1992,b, Iterative Learning Control: A Survey and New Results, *J. Robotic Systems*, **9** (5), 563-584.

Nasar, S.A. and I. Boldea, 1987, Linear electric motors: theory, design and practical application, Prentice-Hall, Englewood Cliffs, NJ, USA.

Ng, G.W., 1997, Application of Neural Networks to Adaptive Control of Nonlinear Systems, Research Studies Press LTD, Somerset, England.

Ohno, H., T. Suzuki, K. Aoki, A. Takashi and G. Sugimoto, 1994, Neural Network Control for Automatic Braking Control System, *Neural Networks*, **7** (8), 1303-1312.

Otten, G., T.J.A. de Vries, J. van Amerongen, A.M. Rankers and E.W. Gaal, 1997, Linear Motor Motion Control Using a Learning Feedforward Controller, *IEEE/ASME Trans. Mechatronics*, **2** (3), 179-187.

Poggio, T. and F. Girosi, 1990, Networks for approximation and learning, *Proceedings IEEE*, **78**, 1481-1497.

Prabhu, S.M. and D.P. Grag, 1996, Artificial Neural Network Based Robot Control: An Overview, *Journal of Intelligent Robotic Systems*, **15**, 333-365.

Roover, D. de, 1996, Synthesis of a Robust Iterative Learning Controller Using an H-inf Approach, Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, 3044-3049.

Roover, D. de, 1997, *Motion Control of a Wafer Stage: A Design Approach for Speeding Up IC Production*, Ph.D. Thesis, Technical Univesity of Delft, Delt University Press.

Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986, Learning Representations by Back-Propagating Errors, *Nature*, **323**, 533-536.

Schaak, P., 1996, *Stability Analysis of a Learning Feed-Forward Controller*, M.Sc. report 036R96, Control Laboratory, University of Twente, Enschede, The Netherlands.

Sjöberg, J., 1995, *Non-linear System Identification with Neural Networks*, Ph.D. Thesis, Linkoping University, Linkoping, Sweden.

Spreeuwers, L., 1999, *Learning Friction Compensation*, M.Sc. report 040R99, Control laboratory, University of Twente, Enschede, The Netherlands.

Starrenburg, J.G., W.T.C. van Luenen, W. Oelen and J. van Amerongen, 1996, Learning feed forward controller for a mobile robot, *Control Eng. Practise*, **4** (9), 1221-1230.

Steenkuijl, J.C., 1999, *Design of a learning feed-forward controller for a robot manipulator*, M.Sc. Report 016R99, Control Laboratory, University of Twente, Enschede, The Netherlands.

Steinbuch, M. and G. Schootstra, 1996, Robust and Adaptive Repetitive Control, *Journal A*, **37** (2), 9-15.

Takagi, T. and M. Sugeno, 1985, Fuzzy Identification of Systems and its Application to Modelling and Control, *IEEE Transactions on Systems, Man and Cybernetics*, **15** (1), 116-132.

Tomizuka, M., T.C. Tsao and K.K. Chew, 1989, Analysis and Synthesis of Discrete-Time Repetitive Controllers, *Journal of Dynamic Systems, Measurement, and Control*, **111**, 353-358.

Tzafestas, S.G., G.G. Rigatos and E.J. Kyriannakis, 1997, Geometry and Thermal Regulation of GMA Welding via Conventional and Neural Adaptive Control, *Journal of Intelligent and Robotic Systems*, **19**, 153-186.

Veelenturf, L.P.J., 1995, Analysis and Application of Artificial Neural Networks, Prentice Hall International.

Velthuis, W.J.R., T.J.A. de Vries and J. van Amerongen, 1996, Learning feed forward control of a flexible beam, Proc. IEEE Int. Symp. on Intelligent Control, Dearborn, MI, USA, 103-108.

Velthuis, W.J.R., P. Schaak, T.J.A. de Vries and E. Gaal, 2000, Stability analysis of learning feed-forward control, to appear in Automatica.

Velthuis, W.J.R., T.J.A. de Vries, K.H.J. Vrielink, G.J. Wierda and A. Borghuis, 1998, Learning control of a flight simulator stick, *Journal A*, **39** (3), 29-34.

Vidyasagar, M. and C.A. Desoer, 1975, Feedback systems: input-output properties, Academic Press, New York, U.S.A.

Vrielink, K.H.J., 1998, *Learning Control Applied to a Control Loading System*, M.Sc. Report 007R98, Control Laboratory, University of Twente, Enschede, The Netherlands.

Vries, T.J.A. de and W.J.R. Velthuis, 1998, Toward exploiting the benefits of ILC in non-repetitive-motion applications, Iterative Learning and Control Workshop and Roundtable, IEEE CDC, Tampa, Florida, USA, 87-88.

Vries, T.J.A. de, L.J. Idema and W.J.R. Velthuis, 1998, Parsimonious Learning Feed-Forward Control, Proc. 6th European Symposium on Artificial Neural Networks ESANN'98, Bruges, Belgium, 85-90.

Wang, L.X., 1994, Adaptive fuzzy systems and control: design and stability analysis, Prentice-Hall, Englewood Cliffs, NJ, USA.

Xu, J.X. and Y. Song, 1998, Direct Learning Control of Non-Uniform Trajectories,261-283, Iterative Learning Control: Analysis, Design, Integration and Applications, Kluwer Academic Publishers.

Xu, J.X., 1997, Direct Learning of Control Efforts for Trajectories with Different Magnitude Scales, *Automatica*, **33** (12), 2191-2196.

Xu, J.X., 1998, Direct Learning of Control Efforts for Trajectories with Different Time Scales, *IEEE Transactions on Automatic Control*, **43** (7), 1027-1029.

Zadeh, L.A., 1973, Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, *IEEE Transactions on Systems, Man and Cybernetics*, **3** (1), 28-44.

# Summary

Learning Feed-Forward Control (LFFC) is a form of Feedback-Error-Learning (FEL) control, i.e. the control system consists of a feedback controller, which compensates for random disturbances, and a feed-forward part that is implemented as a function approximator, which compensates the plant dynamics and the reproducible disturbances. In LFFC, a B-Spline Network (BSN) is chosen for the feed-forward part. During control, the input-output mapping of the BSN is adapted such that the tracking error decreases. It is important to design the feed-forward part of the LFFC in such way that learning converges and that the problems that are associated with the curse of dimensionality are kept to a minimum. These two subjects are the main concerns of this work.

When the plant has to perform repetitive motions, *time-indexed LFFC* should be considered, i.e. the case that the periodic motion time is the only input to the feed-forward part. In this case, LFFC is similar to Iterative Learning Control (ILC) and Repetitive Control (RC). Based on the convergence results of ILC and RC, a stability analysis of time-indexed LFFC has been performed. We have derived conservative stability conditions for the B-spline distribution and for the learning rate. The stability criteria were validated by means of simulations and experiments.

In case of random reference motions, path-indexed LFFC should be applied, i.e. the reference signal and derivatives / integrals thereof should be selected as inputs for the feed-forward part. Which specific inputs should be selected can be determined on the basis of a (structurally correct) state-space representation of the plant dynamics. We showed that a plant can only be controlled by means of path-indexed LFFC when the state space representation meets a number of conditions.

Designing a B-spline distribution for which the path-indexed LFFC is stable may lead to a low-frequency feed-forward signal, which is unattractive. To allow for more dense B-spline distributions, we proposed to add a stabilising measure to the

LFFC, known as regularisation. Regularisation is performed by filtering the learning signal with a time-indexed BSN. Simulations showed that this form of regularisation yielded accurate and stable learning.

When the inputs of the BSN are chosen straightforwardly on basis of the plant model, a multi-dimensional BSN results. This confronts us with problems that are associated with the curse of dimensionality. Splitting the multi-dimensional BSN into several BSNs that have a lower input-dimension offers a solution. The obtained LFFC is known as a *parsimonious* LFFC. Training a parsimonious LFFC is not straightforward. When all BSNs are trained simultaneously, it is not guaranteed that each BSN learns the feed-forward signal it is intended to learn. Therefore, it is proposed to train such an LFFC successively, i.e., by selecting a series of training motions by which the individual BSNs are trained consecutively.

A design procedure for LFFC has been formulated. The LFFC can largely be designed on the basis of qualitative knowledge of the plant and the disturbances. Only for convergence guarantees, a quantitative model of the low-frequency dynamics of the controlled plant is needed, which may be obtained by means of frequency domain identification.

An overall result of this work is that LFFC appears to be an attractive approach for controlling electromechanical motion systems that are subject to reproducible static input disturbances.

# Samenvatting

Learning Feed-Forward Control (LFFC) is een vorm van Feedback-Error-Learning (FEL) control. Het regelsysteem bestaat uit een feedback regelaar en een feed-forward regelaar, die geïmplementeerd is als een functie-approximator. De feed-forward regelaar dient na een leerperiode de dynamica van het proces en de reproduceerbare verstoringen te compenseren, de feedback regelaar compenseert de stochastische verstoringen. Het type functie-approximator dat gebruikt wordt in LFFC is het B-Spline Netwerk (BSN). Tijdens het regelen wordt de ingang-uitgang relatie van het BSN aan de hand van de uitgang van de feedback regelaar op zodanige wijze aangepast dat de volgfout afneemt. Het is belangrijk het feed-forward gedeelte van de LFFC zo te ontwerpen dat het leermechanisme convergeert en dat de problemen die veroorzaakt worden door de zogenaamde 'curse of dimensionality' tot een minimum beperkt blijven. Deze twee onderwerpen vormen het hoofdbestanddeel van dit proefschrift.

Wanneer het re regelen proces zich herhalende bewegingen uitvoert, is het voordelig de tijd als ingang van het feed-forward gedeelte te kiezen. Deze vorm van LFFC heet *tijdgeïndexeerde LFFC* en is nauw verwant aan Iterative Learning Control (ILC) en Repetitive Control (RC). Op basis van convergentiecriteria van ILC en RC is een convergentieanalyse van tijdgeïndexeerde LFFC uitgevoerd. Convergentiecriteria voor de breedte van de B-spline basisfuncties van het BSN en de leerfactor zijn afgeleid. Deze criteria zijn gevalideerd door middel van simulaties en experimenten.

Om willekeurige bewegingen te kunnen uitvoeren, dient de ingang van het BSN te bestaan uit het referentiesignaal en eventueel afgeleiden / geïntegreerden daarvan. Er is sprake van een *padgeïndexeerde* LFFC. Een (structureel correcte) toestandsbeschrijving van het proces geeft aan welke ingangen men dient te kiezen. Aangetoond is dat een proces alleen door middel van LFFC geregeld kan worden wanneer de toestandsbeschrijving aan een aantal voorwaarden voldoet.

Het ontwerpen van een B-spline verdeling zodanig dat de padgeïndexeerde LFFC stabiel is, kan resulteren in een laagfrequent feed-forward signaal, hetgeen niet gewenst is. Om een kleinere breedte van de B-splines toe te staan, is voorgesteld een stabiliserend mechanisme, genaamd regularisatie, aan de LFFC toe te voegen. Het regularisatiemechanisme filtert het leersignaal met een tijdgeïndexeerd BSN dat voldoet aan de bovengenoemde convergentiecriteria. Simulaties laten zien dat het regularisatiemechanisme nauwkeurig en stabiel leren mogelijk maakt.

In het algemeen zal het BSN in de een padgeïndexeerde LFFC meerdere ingangen hebben. Dit kan tot problemen leiden die worden veroorzaakt door de 'curse of dimensionality'. Het splitsen van het multi-dimensionale BSN in meerdere BSN-en beperkt deze problemen tot een minimum. Deze vorm van LFFC heet *parsimonious* (spaarzaam) LFFC. Het trainen van een parsimonious LFFC is niet triviaal. Wanneer alle BSN-en tegelijkertijd worden getraind, is het niet gegarandeerd dat elk BSN het juiste feed-forward signaal leert. Daarom is voorgesteld de BSN-en afzonderlijk te trainen.

Er is een ontwerpprocedure voor LFFC geformuleerd. Het ontwerp van een LFFC is grotendeels gebaseerd op kwalitatieve kennis van het proces en de verstoringen. Alleen het convergentiecriterium vereist een kwantitatief model van de laagfrequente dynamica van het geregelde systeem. Dit model kan zonodig verkregen worden door middel van identificatie.

Het belangrijkste resultaat van dit werk is dat LFFC een aantrekkelijke methode lijkt voor het regelen van elektromechanische processen die onderhevig zijn aan reproduceerbare ingangverstoringen.