Feedback Linearization of Multilinear Time-invariant Systems using Tensor Decomposition Methods

Kai Kruppa and Gerwald Lichtenberg
Faculty of Life Sciences, Hamburg University of Applied Sciences, 21033 Hamburg, Germany

Keywords: Feedback Linearization, MTI Systems, Tensors, Decomposition Methods, Polynomial Methods.

Abstract: Multilinear Time-Invariant (MTI) Systems can represent or approximate nonlinear systems behaviour in a tensor framework. The paper derives efficient algorithms for feedback linearization of MTI systems. Basic operations like products and partial derivatives of multilinear or higher order polynomials and Lie derivatives can be defined in terms of Canonical Polyadic (CP) tensors, as well as the parameters of MTI systems.
The feedback linearizing controller design algorithm given in the paper results in a controller of a known fixed structure with predictable memory demand, which is a great advantage in case of implementation. The structure does not depend on the MTI plant and has to be adjusted to the application by setting its parameters only. Moreover, the MTI algorithm only involves numerical and no symbolic computations. This is relevant for large scale applications like heterogeneous networks e.g. for heat and power distribution, since the model can be represented by decomposed tensors to reduce the storage demand.

1 INTRODUCTION

Controlling nonlinear systems by either linear or nonlinear controllers are challenges of today's applications. Nonlinear control problems result in complex controllers or design procedures in many cases, (Khalil, 1996). Some of them focus on special classes of systems, like bilinear, (Ekman, 2005). Usually specialization on specific classes of systems lowers complexity and improves robustness of the design procedures. For some applications, smaller model classes are too restrictive, e.g. bilinear for heating systems, (Pangalos et al., 2015).
The class of multilinear time-invariant (MTI) systems extend the linear systems by allowing all polynomial terms which ensure the system is linear if all but one variable is held constant, (Pangalos et al., 2015). It also extends the class of bilinear systems. MTI systems are suitable, e.g. for heating systems or chemical systems that are modeled by mass or power balances. Even though the latter systems have no inherent multilinear structure, it was shown that the system behavior can be approximated adequately by an MTI model, (Kruppa et al., 2014).
Feedback linearization is a controller design method for nonlinear systems, where a linear closed loop behavior to the reference input is achieved by nonlinear state feedback, (Isidori, 1995). Computation of the feedback control law can either be done symbolically. There are different approaches that try to compute the controller numerically by automatic differentiation, (Röbenack, 2005) or by using multivariable Legendre polynomials, (Deutscher, 2005). Feedback linearization design techniques were investigated for special system classes like systems that can be approximated well by bilinear models, (Müller and Deutscher, 2005). The methods for general nonlinear systems have in common, that the structure of the controller law concerning the necessary mathematical operations is arbitrary and depends on the plant model.
In this paper a feedback linearization method for MTI systems without symbolic computations is introduced resulting in a controller of fixed structure, where only parameters have to be adjusted for application to a specific plant. The computation is based on tensor algebra since MTI system can be described and simulated very efficiently by canonical polyadic (CP) decomposed tensors, (Pangalos et al., 2015). All system parameters are stored in CP tensors. The applicability of CP tensors in the fields of controller design and diagnosis was shown in (Pangalos, 2016) or (Müller et al., 2015). Therefore, the essential arithmetic operations: multiplication, differentiation and Lie derivatives are derived here based on operational tensors, i.e. the parameter tensors of the result of these oper-

Kruppa, K. and Lichtenberg, G.
Feedback Linearization of Multilinear Time-invariant Systems using Tensor Decomposition Methods.
DOI: 10.5220/0006852802320243
In *Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2018)*, pages 232-243
ISBN: 978-989-758-323-0

ations are computed by the parameter tensors of the operands. The controller law can be computed numerically by standard tensor techniques, e.g. given in (Cichocki et al., 2009) or (Kolda and Bader, 2009). This opens a wider application domain for comparable simple modern controller design techniques. The storage effort for the controller parameter set is predictable and can be described in a decomposed way, which allows a design for large scale systems of multilinear structure.
The paper is organized as follows. Sections 2, 3 and 4 give an introduction to tensor algebra, MTI systems and feedback linearization respectively. Section 5 derives the computation of multiplication, differentiation and Lie derivatives of polynomials based on operational tensors. These methods are used in Section 6 to compute a feedback linearizing control law for MTI systems that are CP decomposed. A complexity analysis is provided in Section 7. An example shows the validity of the approach in Section 8. In Section 9 conclusions are drawn.

2 TENSOR ALGEBRA

This paper focusses on a subclass of polynomial systems. State space models of MTI systems can be represented in a tensor framework. The following definitions for tensors can be found in (Cichocki et al., 2009) or (Kolda and Bader, 2009).

Definition 1 (Tensor)**.** *A tensor of order n*

$$\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$$

is an n-way array. Elements $x(i_1, i_2, \ldots, i_n)$ are indexed by $i_j \in \{1, 2, \ldots, I_j\}$ in each dimension $j = 1, \ldots, n$.

Although this framework has been developed for complex domain $\mathbb{C}$, here only real domains $\mathbb{R}$ are assumed because the numbers result from real physical parameters. Here a Matlab-like tensor notation is used. There are several arithmetic operations available for tensor computations, some are given here.

Definition 2 (Outer Product)**.** *The outer product*

$$\mathsf{Z} = \mathsf{X} \circ \mathsf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times J_1 \times J_2 \times \cdots \times J_m}, \tag{1}$$

of two tensors $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ and $\mathsf{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_m}$ is a tensor of order $n+m$ with elements

$$z(i_1, \ldots, i_n, j_1, \ldots, j_m) = x(i_1, \ldots, i_n) y(j_1, \ldots, j_m). \tag{2}$$

As the outer product is associative, it can be applied in sequence denoted by

$$\bigcirc_{i=1}^{N} \mathsf{X}_i = \mathsf{X}_1 \circ \mathsf{X}_2 \circ \cdots \circ \mathsf{X}_N .$$

Proposition 1 (Product Rule of Differentiation)**.** *The partial derivative with respect to one variable x_j, with $j = 1, \ldots, k$ of an outer product of two tensors $\mathsf{A}(\mathbf{x}) \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ and $\mathsf{B}(\mathbf{x}) \in \mathbb{R}^{J_1 \times \cdots \times J_m}$ depending on k variables $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_k \end{pmatrix}^T$ is given by*

$$\frac{\partial}{\partial x_j}(\mathsf{A}(\mathbf{x}) \circ \mathsf{B}(\mathbf{x})) =$$
$$\frac{\partial}{\partial x_j}(\mathsf{A}(\mathbf{x})) \circ \mathsf{B}(\mathbf{x}) + \mathsf{A}(x) \circ \frac{\partial}{\partial x_j}(\mathsf{B}(\mathbf{x})) . \tag{3}$$

The proposition is proven by applying the standard product rule to the elementwise description of the outer product (2) and rearranging as tensors, which leads to (3).

Definition 3 (Contracted Product)**.** *The* contracted product *of two tensors $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times I_{n+1} \times \cdots \times I_{n+m}}$ and $\mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$*

$$\langle \mathsf{X} \,|\, \mathsf{Y} \rangle (k_1, \ldots, k_m) =$$
$$\sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_n} x(i_1, \ldots, i_n, k_1, \ldots, k_m) y(i_1 \ldots i_n), \tag{4}$$

with $k_i \in \{1, 2, \ldots, I_{n+i}\}$, $i = 1, \ldots, m$, is a tensor of dimension $I_{n+1} \times \cdots \times I_{n+m}$.

Definition 4 (*k*-mode Product)**.** *The k-*mode product *of a tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and a matrix $\mathbf{W} \in \mathbb{R}^{J \times I_k}$ is a tensor*

$$\mathsf{Y} = \mathsf{X} \times_k \mathbf{W} \in \mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times I_N} . \tag{5}$$

The resulting tensor is given elementwise by

$$y(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_N) = \sum_{i_k=1}^{I_k} x(i_1, \ldots, i_N) w(j, i_k).$$

For tensors, the number of elements increases exponentially with the order, such that they have a very high memory demand. Decomposition techniques can significantly reduce complexity. Many tensor decomposition techniques have been developed during the last decades, (Grasedyk et al., 2013). In this paper the *canonical polyadic* (CP) decomposition is used because it is suitable for representation of MTI systems. In CP decomposition a tensor is factorized to a sum of rank 1 elements.

Definition 5 (Rank 1 Tensor)**.** *A n^{th} order tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ is a* rank 1 tensor *if it can be computed by the outer product of n vectors $\mathbf{x}_i \in \mathbb{R}^{I_i}$*

$$\mathsf{X} = \bigcirc_{i=1}^{n} \mathbf{x}_i . \tag{6}$$

Definition 6 (CP Tensor)**.** *A canonical polyadic (CP) tensor of dimension $I_1 \times \cdots \times I_n$ reads*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n] \cdot \lambda = \sum_{k=1}^{r} \lambda(k) \mathbf{X}_1(:,k) \circ \cdots \circ \mathbf{X}_n(:,k)$$
$$= \sum_{k=1}^{r} \lambda(k) \bigcirc_{i=1}^{n} \mathbf{X}_i(:,k) , \tag{7}$$

where elements are computed by the sums of the outer products of the column vectors of so-called factor matrices $\mathbf{X}_i \in \mathbb{R}^{I_i \times r}$*, weighted by the elements of the so-called* weighting *or* parameter vector λ*. An element of the tensor* K *is given by*

$$k(i_1,\ldots,i_n) = \sum_{k=1}^{r} \lambda(k) x_1(i_1,k) \cdots x_n(i_n,k). \quad (8)$$

The minimal number of rank 1 tensors that are summed up in (7) to get K is defined as rank of the CP tensor, (Cichocki et al., 2014).

Example 1. *A 3rd order CP tensor is given by*

$$\begin{aligned}\mathsf{K} &= [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3] \cdot \lambda \\ &= \sum_{k=1}^{r} \lambda(k) \mathbf{X}_1(:,k) \circ \mathbf{X}_2(:,k) \circ \mathbf{X}_3(:,k).\end{aligned}$$

Figure 1 shows the tensor as the sum of outer products of the column vectors $\mathbf{X}_i(:,k)$ *of the factor matrices.*

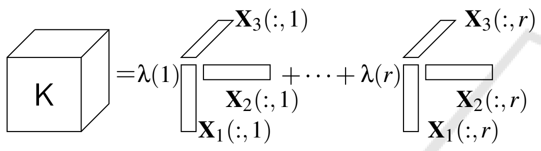


Figure 1: Third order CP tensor.

Standard tensor products like (2) or (4) can be computed very efficiently by simple matrix operations when tensors are represented in a CP format, as e.g. implemented in the Tensor Toolbox, (Bader et al., 2015) or Tensorlab, (Vervliet et al., 2016).

Proposition 2 (Outer Product in CP Form)**.** *The outer product* $\mathsf{Z} = \mathsf{X} \circ \mathsf{Y}$ *of two CP tensors*

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_n] \cdot \lambda_X \in \mathbb{R}^{I_1 \times \cdots \times I_n}, \quad (9)$$

$$\mathsf{Y} = [\mathbf{V}_1, \ldots, \mathbf{V}_m] \cdot \lambda_Y \in \mathbb{R}^{J_1 \times \cdots \times J_m}, \quad (10)$$

with r_X *and* r_Y *rank 1 components respectively gives*

$$\mathsf{Z} = [\mathbf{W}_1, \ldots, \mathbf{W}_{n+m}] \cdot \lambda_Z \in \mathbb{R}^{I_1 \times \cdots \times I_n \times J_1 \times \cdots \times J_m}, \quad (11)$$

that can be represented in a CP format. The factor matrices and weighting vector are given by

$$\mathbf{W}_i = \mathbf{U}_i \otimes \mathbf{1}_{r_Y}^T, \; \forall\, i = 1, \ldots, n, \quad (12)$$

$$\mathbf{W}_{n+i} = \mathbf{1}_{r_X}^T \otimes \mathbf{V}_i, \; \forall\, i = 1, \ldots, m, \quad (13)$$

$$\lambda_Z = \lambda_X \otimes \lambda_Y, \quad (14)$$

with $\mathbf{1}_k$ *denoting a column vector full of ones of length k. Thus, the CP format of* Z *follows directly from the factors of* X *and* Y*. The number of rank 1 components of the resulting tensor* Z *is* $r_Z = r_X r_Y$*.*

The proof 1 of the proposition is given in the Appendix. Although the CP decomposition (11) is the exact outer product of (9) and (10), it could be that a decomposition with a lower number $r_Z < r_X r_Y$ of elements exist. Finding this memory saving representation is an NP-hard problem, (Kolda and Bader, 2009).

Proposition 3 (Contracted Product in CP Form)**.** *The contracted product*

$$\mathbf{z} = \langle \mathsf{X} \,|\, \mathsf{Y} \rangle \in \mathbb{R}^{I_{n+1}}$$

of a tensor of order $n+1$

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_{n+1}] \cdot \lambda_X \in \mathbb{R}^{I_1 \times \cdots \times I_{n+1}},$$

with arbitrary rank and a nth order rank 1 tensor with the same sizes of the dimensions in the first n modes

$$\mathsf{Y} = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{I_1 \times \cdots \times I_n},$$

can be computed based on factor matrices by

$$\mathbf{z} = \mathbf{U}_{n+1} \left(\lambda_X \circledast \left(\mathbf{U}_1^T \mathbf{v}_1 \right) \circledast \cdots \circledast \left(\mathbf{U}_n^T \mathbf{v}_n \right) \right), \quad (15)$$

where $\circledast$ *denotes the Hadamard (elementwise) product, (Pangalos et al., 2015).*

Proposition 4 (k-mode Product in CP Form)**.** *The k-mode product, (Bader et al., 2015)*

$$\mathsf{Y} = \mathsf{X} \times_k \mathbf{W} \quad (16)$$

of a nth order tensor

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_n] \cdot \lambda_{\mathsf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_n} \quad (17)$$

and a matrix $\mathbf{W} \in \mathbb{R}^{J \times I_k}$ *is a CP tensor*

$$\mathsf{Y} = [\mathbf{V}_1, \ldots, \mathbf{V}_n] \cdot \lambda_{\mathsf{Y}} \quad (18)$$

with weighting vector and factor matrices

$$\lambda_{\mathsf{Y}} = \lambda_{\mathsf{X}}, \quad (19)$$

$$\mathbf{V}_i = \begin{cases} \mathbf{W}\mathbf{U}_i & \text{, for } i = k, \\ \mathbf{U}_i & \text{, else.} \end{cases} \quad (20)$$

3 MTI SYSTEMS

Multilinear time-invariant systems have been introduced in (Lichtenberg, 2010) or (Pangalos et al., 2015). MTI systems can be realized by state space models with multilinear functions as right hand sides. With vector spaces $\mathcal{X}_i$, $i = 1, \ldots, n$ of each variable, the overall vector space of the variables is given by

$$\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n.$$

A multilinear mapping $h : \mathcal{X} \to \mathcal{H}$ with vector space $\mathcal{H}$ is linear in each argument, i.e. it fulfills the condition

$$\begin{aligned} &h(x_1, \ldots, a \cdot \tilde{x}_j + b \cdot \hat{x}_j, \ldots, x_n) = \\ &a \cdot h(x_1, \ldots, \tilde{x}_j, \ldots, x_n) + b \cdot h(x_1, \ldots, \hat{x}_j, \ldots, x_n), \quad (21) \end{aligned}$$

for all $x_i \in \mathcal{X}_i$, $\tilde{x}_j, \hat{x}_j \in \mathcal{X}_j$, $1 \leq j \leq n$, $a, b \in \mathbb{R}$, (Hackbusch, 2012). Using multilinear functions as right hand side functions of nonlinear state space models, results in the coordinate-free representation of an MTI system. The multilinear mappings are given by

$$f : \mathcal{X} \times \mathcal{U} \to \mathcal{X},$$
$$g : \mathcal{X} \times \mathcal{U} \to \mathcal{Y},$$

with state $\mathcal{X}$, input $\mathcal{U}$ and output $\mathcal{Y}$ space. Both mappings fulfill the multilinearity condition (21). Next, coordinates $\mathbf{x} \in \mathcal{X}$, $\mathbf{u} \in \mathcal{U}$ and $\mathbf{y} \in \mathcal{Y}$ are defined to describe multilinear functions.

Definition 7 (Multilinear Function)**.** *A multilinear function $h : \mathbb{R}^n \to \mathbb{R}$*

$$h(\mathbf{x}) = \alpha^T \mathbf{m}(\mathbf{x}) \tag{22}$$

with coefficient vector $\alpha = \begin{pmatrix}\alpha_1 & \cdots & \alpha_{2^n}\end{pmatrix}^T \in \mathbb{R}^{2^n}$ and monomial vector $\mathbf{m}(\mathbf{x}) = \begin{pmatrix}1\\ x_n\end{pmatrix} \otimes \cdots \otimes \begin{pmatrix}1\\ x_1\end{pmatrix} \in \mathbb{R}^{2^n}$ is a polynomial in n variables, which is linear, when all but one variable are held constant.

Example 2. *A multilinear function with 2 variables x_1 and x_2 is given by*

$$h(x_1,x_2) = \alpha^T \mathbf{m}(x_1,x_2) = \begin{pmatrix}\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4\end{pmatrix}\begin{pmatrix}1\\ x_1\\ x_2\\ x_1x_2\end{pmatrix}$$
$$= \alpha_1 \cdot 1 + \alpha_2 \cdot x_1 + \alpha_3 \cdot x_2 + \alpha_4 \cdot x_1x_2.$$

Introduction of coordinates leads to the state space representation of MTI systems that can be written in a tensor structure. The multilinear monomials can be rearranged in a tensor framework. The monomial tensor

$$\mathsf{M}(\mathbf{x},\mathbf{u}) = \left[\begin{pmatrix}1\\ u_m\end{pmatrix}, \ldots, \begin{pmatrix}1\\ u_1\end{pmatrix}, \begin{pmatrix}1\\ x_n\end{pmatrix}, \ldots, \begin{pmatrix}1\\ x_1\end{pmatrix}\right], \tag{23}$$

of dimension $\mathbb{R}^{\times^{(n+m)}2}$ is rank one.[1] By rearranging the parameters of the MTI system in the same way than the monomial tensor, one gets its tensor representation.

Definition 8 (MTI System in Tensor Form)**.** *Using the monomial tensor (23), the tensor representation of an MTI system with n states, m inputs and p outputs reads*

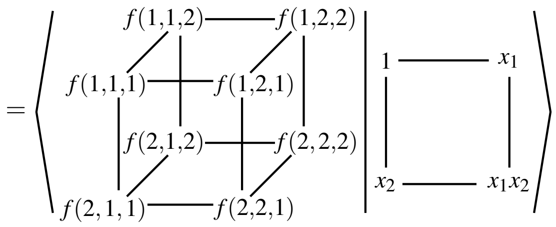
$$\dot{\mathbf{x}} = \langle \mathsf{F} \,|\, \mathsf{M}(\mathbf{x},\mathbf{u}) \rangle, \tag{24}$$
$$\mathbf{y} = \langle \mathsf{G} \,|\, \mathsf{M}(\mathbf{x},\mathbf{u}) \rangle, \tag{25}$$

with the transition tensor $\mathsf{F} \in \mathbb{R}^{\times^{(n+m)}2 \times n}$ and the output tensor $\mathsf{G} \in \mathbb{R}^{\times^{(n+m)}2 \times p}$.

Example 3. *The state equation of an MTI system with two states is given by*

[1]The notation $\mathbb{R}^{\times^{(n+m)}2}$ denotes the space $\mathbb{R}^{\overbrace{2\times\ldots\times 2}^{n+m \text{ times}}}$.

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \langle \mathsf{F} \,|\, \mathsf{M}(x_1,x_2) \rangle = \left\langle \mathsf{F} \,\middle|\, \left[\begin{pmatrix}1\\ x_2\end{pmatrix}, \begin{pmatrix}1\\ x_1\end{pmatrix}\right] \right\rangle$$



$$= \begin{pmatrix} f(1,1,1)+f(1,2,1)x_1+f(2,1,1)x_2+f(2,2,1)x_1x_2\\ f(1,1,2)+f(1,2,2)x_1+f(2,1,2)x_2+f(2,2,2)x_1x_2\end{pmatrix}.$$

The number of parameters of an MTI system given by the numbers of elements in the tensors F and G increases exponentially with the numbers of states and inputs, i.e. the transition tensor F contains $n \cdot 2^{n+m}$ elements. This leads to problems in memory demand when dealing with large models. Decomposition methods can be used to reduce the complexity of high dimensional models. Therefore, a CP decomposition of the parameter tensors

$$\mathsf{F} = [\mathbf{F}_{u_m}, \ldots, \mathbf{F}_{u_1}, \mathbf{F}_{x_n}, \ldots, \mathbf{F}_{x_1}, \mathbf{F}_{\Phi}] \cdot \lambda_F, \tag{26}$$
$$\mathsf{G} = [\mathbf{G}_{u_m}, \ldots, \mathbf{G}_{u_1}, \mathbf{G}_{x_n}, \ldots, \mathbf{G}_{x_1}, \mathbf{G}_{\Phi}] \cdot \lambda_G, \tag{27}$$

is desired, which can be found for every MTI system, (Kruppa and Lichtenberg, 2016). During simulation, by using (15), the contracted product of the CP decomposed MTI system (24) and (25) can be computed very efficiently, since the monomial tensor is a rank one tensor.

Example 4. *For an MTI system with two states and state equation*

$$\dot{\mathbf{x}} = \begin{pmatrix} 2x_2 - 5x_1x_2\\ -x_1 + 3x_2 - 4x_1x_2u_1\end{pmatrix},$$

the factor matrices and the weighting vector of the transition tensor $\mathsf{F} = [\mathbf{F}_{u_1}, \mathbf{F}_{x_2}, \mathbf{F}_{x_1}, \mathbf{F}_{\Phi}] \cdot \lambda_F$ reads

$$\mathbf{F}_{x_1} = \begin{pmatrix}1&0&0&1&0\\0&1&1&0&1\end{pmatrix}, \mathbf{F}_{x_2} = \begin{pmatrix}0&0&1&0&0\\1&1&0&1&1\end{pmatrix},$$
$$\mathbf{F}_{u_1} = \begin{pmatrix}1&1&1&1&0\\0&0&0&0&1\end{pmatrix}, \mathbf{F}_{\Phi} = \begin{pmatrix}1&1&0&0&0\\0&0&1&1&1\end{pmatrix},$$
$$\lambda_F = \begin{pmatrix}2 & -5 & -1 & 3 & -4\end{pmatrix}^T.$$

4 FEEDBACK LINEARIZATION

This section highlights the most important features of feedback linearization as described in (Isidori, 1995) or (Khalil, 1996). The main idea of this method is to design a state feedback controller for an input affine nonlinear plant such that the reference behavior in

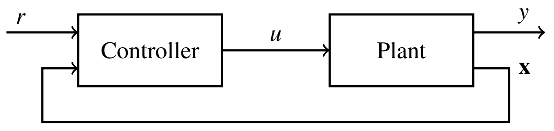


Figure 2: Closed loop state feedback.

closed loop from reference input r to output y is linear as shown in Figure 2.

Consider a nonlinear, input affine, single-input single-output (SISO) system

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u, \tag{28}$$

$$y = c(\mathbf{x}), \tag{29}$$

with nonlinear functions $\mathbf{a}, \mathbf{b} : \mathbb{R}^n \to \mathbb{R}^n$ and output function $c : \mathbb{R}^n \to \mathbb{R}$. It is assumed that the functions are sufficiently smooth, which means that all later appearing partial derivatives of the functions are defined and continuous, (Khalil, 1996). A nonlinear state feedback controller is constructed by, (Isidori, 1995)

$$u = \frac{-\sum_{i=0}^{\rho} \mu_i L_{\mathbf{a}}^i c(\mathbf{x}) + \mu_0 r}{L_{\mathbf{b}} L_{\mathbf{a}}^{\rho-1} c(\mathbf{x})}, \tag{30}$$

such that the closed loop shows a linear behavior from r to y prescribed by the linear differential equation

$$\mu_\rho \frac{\partial^\rho}{\partial t^\rho} y + \mu_{\rho-1} \frac{\partial^{\rho-1}}{\partial t^{\rho-1}} y + \ldots + \mu_1 \dot{y} + \mu_0 y = \mu_0 r. \tag{31}$$

Without loss of generality the factor μ_ρ is set to one. The Lie derivative of a scalar function $g(\mathbf{x})$ along a vector field $\mathbf{h}(\mathbf{x})$ is defined as, (Isidori, 1995)

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial g(\mathbf{x})}{\partial x_i}. \tag{32}$$

An i-times application of the Lie derivative to a function is denoted by $L_{\mathbf{h}} \cdots L_{\mathbf{h}} g(\mathbf{x}) = L_{\mathbf{h}}^i g(\mathbf{x})$.

5 ARITHMETIC OPERATIONS FOR POLYNOMIALS

To determine the Lie derivatives, multiplication and differentiation of functions have to be computed. In the following, these arithmetic operations are investigated, if the functions are polynomial and given in terms of a tensor structure by a contracted product of a parameter and a monomial tensor. A multilinear function (22) in tensor representation is written as

$$h(\mathbf{x}) = \langle \mathsf{H} \,|\, \mathsf{M}(\mathbf{x}) \rangle. \tag{33}$$

The multiplication of two multilinear functions is not a multilinear function anymore, since in general higher order terms occur in the result. Because of that, a representation of higher order polynomials in the tensor structure has to be introduced.

Definition 9 (Polynomial in Tensor Form)**.** *A polynomial with maximal order N of the monomials in n variables $\mathbf{x} \in \mathbb{R}^n$ is given by*

$$h(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{34}$$

with parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2}$ and the monomial tensor $\mathsf{M}_p^N(\mathbf{x}) \in \mathbb{R}^{\times^{(nN)}2}$, which is constructed as rank 1 tensor, analogous to (23), by

$$\mathsf{M}_p^N(\mathbf{x}) = \bigcirc_{j=1}^{N} \mathsf{M}(\mathbf{x})$$
$$= \underbrace{\left[\begin{pmatrix}1\\x_n\end{pmatrix}, \ldots, \begin{pmatrix}1\\x_1\end{pmatrix}, \ldots, \begin{pmatrix}1\\x_n\end{pmatrix}, \ldots, \begin{pmatrix}1\\x_1\end{pmatrix}\right]}_{N\ times}. \tag{35}$$

A maximal order N of the monomials means that variables with exponents up to N, i.e. x_i^N could occur. Obviously by construction some monomials appear multiple times inside the monomial tensor. This choice of the monomial tensor, which is not unique, is not optimal regarding the storage effort but helps developing the algorithms. This redundancy is still acceptable, since the monomial tensor is a rank 1 tensor. Using this tensor representation of multilinear and polynomial functions, the aim here is to compute the parameter tensor of the result of multiplication and differentiation, just by using the parameter tensors of the operands, without evaluating the contracted product with the monomial tensor.

5.1 Multiplication

With Definition 9 it is possible to derive the multiplication based on the parameter tensors.

Proposition 5 (Multiplication in Tensor Form)**.** *The multiplication of two polynomials in n variables of orders N_1 and N_2 in tensor representation*

$$h_1(\mathbf{x}) = \left\langle \mathsf{H}_1 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \right\rangle, \tag{36}$$

$$h_2(\mathbf{x}) = \left\langle \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle, \tag{37}$$

with parameter tensors $\mathsf{H}_i \in \mathbb{R}^{\times^{(nN_i)}2}$ and monomial tensors $\mathsf{M}_p^{N_i}(\mathbf{x}) \in \mathbb{R}^{\times^{(nN_i)}2}$, $i = 1, 2$,

$$h_1(\mathbf{x}) \cdot h_2(\mathbf{x}) = \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1+N_2}(\mathbf{x}) \right\rangle, \tag{38}$$

is a polynomial of order $N_1 + N_2$.

The proposition is proven in proof 2 in the Appendix. If the parameter tensors H_1 and H_2 are given as CP tensors, the parameter tensor of their product is efficiently computed by (11) – (14). Thus building the full tensors of H_1 and H_2 is not required.

Example 5. *The multiplication of two multilinear functions with tensor representations*

$$h_1(x_1,x_2) = 1+2x_1x_2 = \langle \mathsf{H}_1 \,|\, \mathsf{M}(x_1,x_2) \rangle$$
$$= \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle, \tag{39}$$
$$h_2(x_1,x_2) = x_1 - 3x_1x_2 = \langle \mathsf{H}_2 \,|\, \mathsf{M}(x_1,x_2) \rangle$$
$$= \left\langle \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle. \tag{40}$$

is given by

$$h(x_1,x_2) = \langle \mathsf{H}_1 \,|\, \mathsf{M}(x_1,x_2) \rangle \langle \mathsf{H}_2 \,|\, \mathsf{M}(x_1,x_2) \rangle$$
$$= \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^2(x_1,x_2) \right\rangle.$$

The result of the multiplication is a polynomial of maximal order 2 *as shown by the monomial tensor* $\mathsf{M}_p^2(x_1,x_2)$ *of the result. Computing the outer product* $\mathsf{H} = \mathsf{H}_1 \circ \mathsf{H}_2$ *gives the parameter tensor of the result with slices*

$$\mathbf{H}(:,:,1,1) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,1,2) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix},$$
$$\mathbf{H}(:,:,2,1) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,2,2) = \begin{pmatrix} -3 & 0 \\ 0 & -6 \end{pmatrix}.$$

The evaluation of the parameter tensor $\mathsf{H}_1 \circ \mathsf{H}_2$ *of the multiplication and the monomial tensor gives*

$$h(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^2(\mathbf{x}) \right\rangle = x_1 - 3x_1x_2 + 2x_1^2x_2 - 6x_1^2x_2^2,$$

which shows that the tensor representation of the result is computed correctly.

5.2 Differentiation

In this section the computation of the partial derivative of a polynomial in tensor representation (34) with respect to one variable x_j is investigated.

Proposition 6 (Differentiation in Tensor Form)**.** *The partial derivative of a polynomial in n variables of maximal monomial order N with respect to one variable* $x_j,\ j=1,\dots,n$ *in tensor representation is given by*

$$\frac{\partial}{\partial x_j} h(\mathbf{x}) = \frac{\partial}{\partial x_j} \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = \left\langle \mathsf{H}_j \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{41}$$

with the parameter tensor of the differentiated function and $\Theta = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$

$$\mathsf{H}_j = \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \Theta \in \mathbb{R}^{\times^{nN} 2}. \tag{42}$$

The proof 3 of the proposition can be found in the Appendix. Assuming the parameter tensor H is in CP form, as desired the CP representation of H_j is constructed using the factors of H only, without building the full tensors, since the *k*-mode product is defined for CP tensors as given in Proposition 4.

Example 6. *Consider a polynomial with* 2 *variables*

$$h(\mathbf{x}) = 1 + 2x_1 + 3x_2 + 6x_1x_2$$
$$= \langle \mathsf{H} \,|\, \mathsf{M}(\mathbf{x}) \rangle = \left\langle \begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle.$$

By applying (42) the parameter tensor of the differentiated function with respect to x_1 *results in*

$$\mathsf{H}_1 = \mathsf{H} \times_2 \Theta = \begin{pmatrix} 2 & 0 \\ 6 & 0 \end{pmatrix},$$

which describes the function

$$\frac{\partial}{\partial x_1} h(\mathbf{x}) = \langle \mathsf{H}_1 \,|\, \mathsf{M}(\mathbf{x}) \rangle = 2 + 6x_2.$$

The 2*-mode multiplication with* Θ *maps the elements of* H *belonging to the monomials* x_1 *and* x_1x_2 *to the monomials* 1 *and* x_2*. All others are zero, which is exactly the case when differentiating this function with respect to* x_1*.*

The parameter tensor

$$\mathsf{H} = \left[\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \right] \cdot \begin{pmatrix} 1 & 2 & 3 & 6 \end{pmatrix}^T$$

can be given in a CP tensor structure. By multiplying the second factor matrix from the right by Θ*, the CP representation of the differentiated function reads*

$$\mathsf{H}_1 = \left[\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 1 & 2 & 3 & 6 \end{pmatrix}^T$$
$$= \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 2 & 6 \end{pmatrix}^T,$$

where the number of rank 1 components can be reduced by removing the zero columns of the factor matrices to get a more efficient CP tensor representation.

5.3 Lie Derivatives

The multiplication and differentiation methods (38) and (41) are used to compute the Lie derivative (32) for the case that polynomials are given in CP form

$$\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{43}$$
$$g(\mathbf{x}) = \left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle. \tag{44}$$

Theorem 1 (Lie Derivative in Tensor Form)**.** *The tensor representation of the Lie derivative (32) of the scalar polynomial (44) along the polynomial vector field (43) yields*

$$L_{\mathbf{h}}^l g(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},l} \,\middle|\, \mathsf{M}_p^{N(l+1)}(\mathbf{x}) \right\rangle. \tag{45}$$

with the parameter tensor

$$\mathsf{L}_{\mathsf{F},\mathsf{G},l} = \begin{cases} \mathsf{G} & \text{for } l=0, \\ \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \Theta \right) & \text{for } l=1, \\ \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{lN} \mathsf{L}_{\mathsf{F},\mathsf{G},l-1} \times_{kn-i+1} \Theta \right) & \text{else,} \end{cases}$$

with subtensor $\mathsf{H}_i = \mathsf{H}(:,\ldots,:,i)$*, where the last dimension of* H *is fixed.*

In the Appendix the proof 4 of the theorem is given.

Example 7. *The vector function*

$$\mathbf{h}(x_1,x_2) = \begin{pmatrix} h_1(x_1,x_2) \\ h_2(x_1,x_2) \end{pmatrix} = \langle\, \mathsf{H} \,|\, \mathsf{M}(x_1,x_2)\,\rangle$$

composed of polynomials (39) and (40) has a parameter tensor

$$\mathsf{H}_1 = \mathbf{H}(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix},\ \mathsf{H}_2 = \mathbf{H}(:,:,2) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix}.$$

The first Lie derivative $L_{\mathbf{h}}g(\mathbf{x})$ *of the scalar function*

$$\begin{aligned} g(\mathbf{x}) &= x_2 - x_1x_2 = \langle\, \mathsf{G} \,|\, \mathsf{M}(\mathbf{x})\,\rangle \\ &= \left\langle \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle, \end{aligned}$$

along the vector field $\mathbf{h}(\mathbf{x})$ *should be computed by operational tensors. For comparison the Lie derivative with the standard approach (32) gives*

$$\begin{aligned} L_{\mathbf{h}}g(\mathbf{x}) &= h_1(\mathbf{x})\frac{\partial}{\partial x_1}g(\mathbf{x}) + h_2(\mathbf{x})\frac{\partial}{\partial x_2}g(\mathbf{x}) \\ &= x_1 - x_2 - 3x_1x_2 - x_1^2 + 3x_1^1x_2 - 2x_1x_2^2. \end{aligned} \tag{46}$$

With the method introduced in Theorem 1 the parameter tensor of the Lie derivative

$$\mathsf{L}_{\mathsf{H},\mathsf{G},1} = \mathsf{H}_1 \circ (\mathsf{G} \times_2 \Theta) + \mathsf{H}_2 \circ (\mathsf{G} \times_1 \Theta),$$

results in a fourth order tensor

$$\mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,1,1) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix},\quad \mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,1,2) = \begin{pmatrix} 0 & -1 \\ 0 & 3 \end{pmatrix},$$

$$\mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,2,1) = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}, \mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,2,2) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

From (45) follows that the monomial tensor $\mathsf{M}_p^2(x_1,x_2)$ *for the Lie derivative is of maximal order 2. Evaluating the contracted product of the parameter tensor with the monomial tensor* $\langle\, \mathsf{L}_{\mathsf{H},\mathsf{G},1} \,|\, \mathsf{M}_p^2(x_1,x_2)\,\rangle$ *shows that the result with the operational tensor is equal to (46). Thus, the parameter tensor is computed correctly.*

6 FEEDBACK LINEARIZATION FOR CP DECOMPOSED MTI SYSTEMS

The controller design by feedback linearization introduced in Section 4 is investigated in this section for MTI systems with CP decomposed parameter tensors. An input affine SISO MTI system is given by

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u = \langle\, \mathsf{F} \,|\, \mathsf{M}(\mathbf{x},u)\,\rangle \\ &= \langle\, \mathsf{A} \,|\, \mathsf{M}(\mathbf{x})\,\rangle + \langle\, \mathsf{B} \,|\, \mathsf{M}(\mathbf{x})\,\rangle\, u, \end{aligned} \tag{47}$$

$$\begin{aligned} y &= c(\mathbf{x}) = \langle\, \mathsf{G} \,|\, \mathsf{M}(\mathbf{x},u)\,\rangle \\ &= \langle\, \mathsf{C} \,|\, \mathsf{M}(\mathbf{x})\,\rangle. \end{aligned} \tag{48}$$

It is assumed that the system is controllable, observable and feedback linearizable with well-defined relative degree ρ. Theorem 1 is used to compute the Lie derivatives of the multilinear model functions in tensor form. This leads to the following lemma.

Lemma 1 (MTI System Lie Derivatives)**.** *The Lie derivative (32) of the multilinear output function given by tensor* C *along the multilinear vector field* $\mathbf{a}(\mathbf{x})$ *given by tensor* A *yields*

$$\begin{aligned} L_{\mathbf{a}}^l c(\mathbf{x}) &= \sum_{i=1}^n a_i(\mathbf{x})\frac{\partial}{\partial x_i}L_{\mathbf{a}}^{l-1}c(\mathbf{x}) \\ &= \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},l} \,\middle|\, \mathsf{M}_p^{l+1}(\mathbf{x}) \right\rangle, \end{aligned} \tag{49}$$

with parameter tensor

$$\mathsf{L}_{\mathsf{A},\mathsf{C},l} = \begin{cases} \mathsf{C} & \text{for } l=0, \\ \sum\limits_{i=1}^n \mathsf{A}_i \circ (\mathsf{C} \times_{n-i+1} \Theta) & \text{for } l=1, \\ \sum\limits_{i=1}^n \mathsf{A}_i \circ \left(\sum\limits_{k=1}^l \mathsf{L}_{\mathsf{A},\mathsf{C},l-1} \times_{kn-i+1} \Theta \right) & \text{else}, \end{cases}$$

with $l = 0,\ldots,\rho$ *and subtensor* $\mathsf{A}_i = \mathsf{A}(:,\ldots,:,i)$ *of* A*. Using the parameter tensor*

$$\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} = \sum_{i=1}^n \mathsf{B}_i \circ \left(\sum_{k=1}^{l+1} \mathsf{L}_{\mathsf{A},\mathsf{C},l} \times_{kn-i+1} \Theta \right), \tag{50}$$

with $\mathsf{B}_i = \mathsf{B}(:,\ldots,:,i)$*, the Lie derivatives along* $\mathbf{b}(\mathbf{x})$ *are given by*

$$L_{\mathbf{b}}L_{\mathbf{a}}^l c(\mathbf{x}) = \sum_{i=1}^n b_i(\mathbf{x})\frac{\partial}{\partial x_i}L_{\mathbf{a}}^l c(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} \,\middle|\, \mathsf{M}_p^{l+2}(\mathbf{x}) \right\rangle.$$

The representations of the Lie derivatives of the system (47) and (48) follows obviously from Theorem 1. Since all operations, i.e. summation, outer product and *k*-mode product, used to get the parameter tensors of the Lie derivatives are introduced for CP tensors, no full tensor has to be build during computation of the Lie derivatives. This is important especially for large scale systems. With the tensor description of the Lie derivatives given in Lemma 1 the controller law for feedback linearization can be constructed as shown in the following lemma.

Lemma 2 (MTI Feedback Linearization)**.** *The feedback linearizing controller for an input affine MTI* *system given by (47) and (48) reads*

$$u = \frac{-\left\langle \sum_{i=0}^{\rho} \mu_i \mathsf{L}_{\mathsf{A,C},i} \,\middle|\, \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle + \mu_0 r}{\left\langle \mathsf{L}_{\mathsf{B,A,C},\rho-1} \,\middle|\, \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle}, \qquad (51)$$

when the Lie derivatives are defined by parameter tensors as in Lemma 1.

The proof 5 of the previous lemma is described in the Appendix. Thus, the feedback linearizing controller has got a fixed structure. The factors μ_i and tensors $\mathsf{L}_{\mathsf{A,C},i}$ and $\mathsf{L}_{\mathsf{B,A,C},\rho-1}$ have to be provided, to adapt the controller for a given plant. Setting these parameters is like setting a static state feedback gain matrix for pole placement in the linear case. The structure of the controller for MTI systems is fixed and does not depend on the plant. This structural invariance is an advantage for application of the controller. Compared to the general nonlinear case it is not necessary to provide an arbitrary set of mathematical operations, that depends on the plant model. Here it is limited to the given structure and the operations can be defined efficiently on the coefficient spaces.

7 COMPLEXITY ANALYSIS

In today's applications systems get more and more complex, like in smart grids or heating systems. Plants in these application areas can be modeled by MTI systems with a large number of states. With dense parameter tensors these models can capture very complex dynamics. But a large order n leads to a very large number of system parameters in full representations. The number of terms also in a symbolic representation increases exponentially with the number of states. Thus, for large scale systems a system description is impossible because of curse of dimensionality. A system of order 100 has got more than 10^{32} terms in full representation, which does not fit in memory anymore. The state space representation and thus the computation of a controller of such big system is not possible in a symbolical way.
Tensor decomposition methods like CP decomposition allow to compute low rank approximations of the parameter tensors. This breaks the curse of dimensionality and allows an approximate representation of large scale systems with a number of parameters that is lower by orders of magnitude. Figure 3 shows in a logarithmic scale how the number of parameters increases with the order of the system and the rank of the parameter tensors.

Using low rank approximation techniques for parameter tensors, makes it possible to represent large

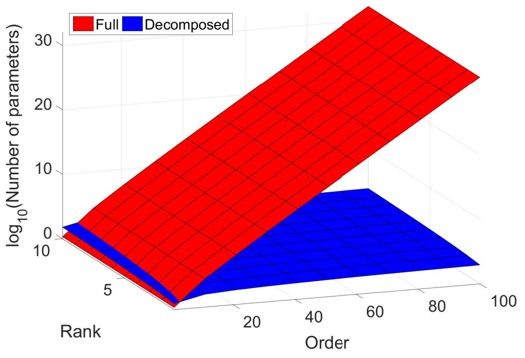


Figure 3: Number of parameter for MTI systems.

systems and to compute controllers as shown in this paper. E.g. the number of values to be stored for a system of order 100 is reduced from 10^{32} to 6200 for a rank 10 approximation, since the parameters scale linearly with order and rank. The proposed feedback linearization approach works with the decomposed representations of the system and all operations are defined for CP decomposed tensors such that no full tensor representation has to be constructed during the whole controller design process leading to a controller in a decomposed structure. The storage demand is still capable as shown in Figure 4, where an upper bound for the number of elements to be stored for the parameter tensors of the controller are depicted.

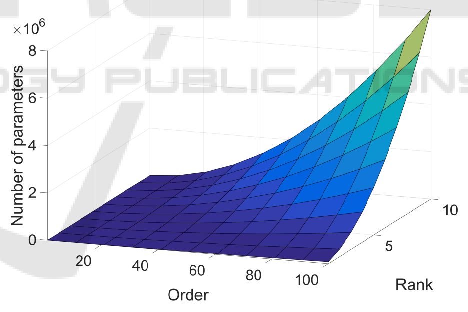


Figure 4: Upper bound for number of parameters for a feedback linearizing controller of decomposed MTI systems.

The figure considers different orders and different ranks of the system tensors A, B and C for a system with relative degree of one. Often the results can be represented by tensors of lower rank which reduces the storage effort further. Also during application the controller law can be evaluated based on the decomposed factor matrices. This makes it possible to use this feedback linearizing scheme also for large scale MTI systems, where the full representation can not processed but low rank approximations exist. This break of the dimensionality problem for large scale systems is the big advantage of the introduced feedback linearization design approach by CP decomposed tensors.

8 APPLICATION

In this section the proposed method for feedback linearization is applied first to a SISO, MTI system with 2 states for giving all steps of the design procedure explicitly. Consider the system

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \begin{pmatrix}2x_2\\ -x_1+0.2x_1x_2\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix}u,$$
$$y = 1+2x_1.$$

The systems belongs to the class of input affine MTI systems (47) and (48) where the factor matrices are represented as CP tensors

$$\mathsf{A} = \left[\begin{pmatrix}0&1&0\\1&0&1\end{pmatrix},\begin{pmatrix}1&0&0\\0&1&1\end{pmatrix},\begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}\right]\cdot\begin{pmatrix}2\\-1\\0.2\end{pmatrix},$$
$$\mathsf{B} = \left[\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix}\right]\cdot 1,$$
$$\mathsf{C} = \left[\begin{pmatrix}1&1\\0&0\end{pmatrix},\begin{pmatrix}0&1\\1&0\end{pmatrix}\right]\cdot\begin{pmatrix}2\\1\end{pmatrix}.$$

To design the controller, the Lie derivatives of the system must be derived. Using (49) the parameter tensors of the Lie derivatives along $\mathbf{a}(\mathbf{x})$ in CP representation are given by

$$\mathsf{L}_{\mathsf{A,C},1} = \left[\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right]\cdot 4,$$
$$\mathsf{L}_{\mathsf{A,C},2} = \left[\begin{pmatrix}1&0\\0&1\end{pmatrix},\begin{pmatrix}0&0\\1&1\end{pmatrix},\begin{pmatrix}1&1\\0&0\end{pmatrix},\dots\right.$$
$$\left.\begin{pmatrix}1&1\\0&0\end{pmatrix},\begin{pmatrix}1&1\\0&0\end{pmatrix},\begin{pmatrix}1&1\\0&0\end{pmatrix}\right]\cdot\begin{pmatrix}-4\\0.8\end{pmatrix}$$

resulting in the derivatives

$$L_{\mathbf{a}}c(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{A,C},1} \,\middle|\, \mathsf{M}_p^2 \right\rangle = 4x_2,$$
$$L_{\mathbf{a}}^2c(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{A,C},2} \,\middle|\, \mathsf{M}_p^3 \right\rangle = -4x_1+0.8x_1x_2.$$

The Lie derivatives along $\mathbf{b}(\mathbf{x})$ leads to $L_{\mathbf{b}}c(\mathbf{x}) = 0$ and $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4$ by the parameter tensors (50)

$$\mathsf{L}_{\mathsf{B,A,C},0} = \left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix}\right]\cdot 0,$$
$$\mathsf{L}_{\mathsf{B,A,C},1} = \left[\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right]\cdot 4.$$

Because of $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4 \neq 0\ \forall\mathbf{x}$ the relative degree ρ of the system is two, which is equal to the number of states. Thus the system has full degree and the zero dynamics needs not to be checked. Additionally the system is controllable and observable. In closed loop, the linear behavior

$$\mu_2\ddot{y}+\mu_1\dot{y}+\mu_0 y = \mu_0 r$$

is desired. As an example $\mu_1 = \mu_0 = 10$ and $\mu_2 = 1$ are chosen. Using the parameter tensors of the Lie derivatives, the feedback linearizing controller is given by

$$u = \frac{-\left\langle \mu_0\mathsf{C}+\mu_1\mathsf{L}_{\mathsf{A,C},1}+\mu_2\mathsf{L}_{\mathsf{A,C},2} \,\middle|\, \mathsf{M}_p^3(\mathbf{x})\right\rangle + \mu_0 r}{\left\langle \mathsf{L}_{\mathsf{B,A,C},1} \,\middle|\, \mathsf{M}_p^3(\mathbf{x})\right\rangle}$$
$$= -\frac{10+16x_1+40x_2+0.8x_1x_2+10r}{4}.$$

Figure 5 shows the closed loop simulation result with $\mathbf{x}_0 = \begin{pmatrix}-0.5 & 0\end{pmatrix}^T$ for a step change in reference signal r. The system behaves as the specified 2nd order linear system.

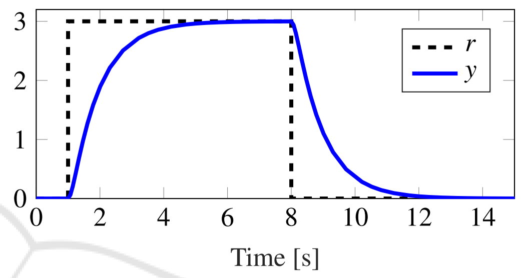


Figure 5: Closed loop simulation.

9 CONCLUSION AND OUTLOOK

In this paper a feedback linearization approach for CP decomposed MTI systems was derived. The controller is determined numerically without any symbolic computation. Therefore multiplication and differentiation of polynomial functions are investigated that are necessary for computation of the Lie derivatives of the system. It is shown that the parameter tensors of the result of these operations can be computed efficiently by the parameter tensors of the operands if in CP. With that the controller law of the feedback linearizing controller is computed. This is an important advantage when focusing large systems where a standard representation of right hand sides is not possible, since the use of decomposition techniques leads to a significant reduction in storage demand by orders of magnitude. This construction of the linearizing feedback gain leads to a controller with a structure that is independent of the plant. Only a parameter set has to be computed to use the controller for a special application.

Here the standard method of feedback linearization for SISO systems is investigated. This approach could be extended to MIMO systems. From a mathematical point of view it is interesting to use other tensor decomposition techniques for the design algorithm like Tensor Trains, (Oseledets, 2011). Another interesting point is to compare the proposed method to other approaches of feedback linearization to determine a threshold for the system size, where other methods fail and the tensor methods is the only applicable algorithm.

ACKNOWLEDGEMENTS

This work was partly sponsored by the project OBSERVE of the Federal Ministry of Economic Affairs and Energy, Germany (Grant No.: 03ET1225B).

REFERENCES

Bader, B. W., Kolda, T. G., et al. (2015). Matlab tensor toolbox version 2.6. Available online.

Cichocki, A., Mandic, D. P., Phan, A. H., Caiafa, C. F., Zhou, G., Zhao, Q., and Lathauwer, L. D. (2014). Tensor decompositions for signal processing applications from two-way to multiway component analysis. *CoRR*, abs/1403.4462.

Cichocki, A., Zdunek, R., Phan, A., and Amari, S. (2009). *Nonnegative matrix and tensor factorizations*. Wiley, Chichester.

Deutscher, J. (2005). Input-output linearization of nonlinear systems using multivariable legendre polynomials. *Automatica*, 41:299–304.

Ekman, M. (2005). *Modeling and Control of Bilinear Systems: Application to the Activated Sludge Process*. PhD thesis, Uppsala University.

Grasedyk, L., Kressner, L., and Tobler, C. (2013). A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36:53–78.

Hackbusch, W. (2012). *Tensor Spaces and Numerical Tensor Calculus*, volume 42 of *Springer Series in Computational Mathematics*. Springer-Verlag Berlin Heidelberg.

Isidori, A. (1995). *Nonlinear Control Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition.

Khalil, K. H. (1996). *Nonlinear Systems*. Prentice-Hall, Inc.

Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Kruppa, K. and Lichtenberg, G. (2016). Comparison of cp tensors, tucker tensors and tensor trains for representation of right hand sides of ordinary differential equations. In *Workshop on Tensor Decomposition and Applications, Leuven*.

Kruppa, K., Pangalos, G., and Lichtenberg, G. (2014). Multilinear approximation of nonlinear state space models. In *19th IFAC World Congress, Cape Town*, pages 9474–9479. IFAC.

Lichtenberg, G. (2010). Tensor representation of boolean functions and zhegalkin polynomials. In *International Workshop on Tensor Decompositions, Bari*.

Müller, B. and Deutscher, J. (2005). Approximate input-output linearization using l2-optimal bilinearization. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*.

Müller, T., Kruppa, K., Lichtenberg, G., and Réhault, N. (2015). Fault detection with qualitative models reduced by tensor decomposition methods. *IFAC-PapersOnLine*, 48(21):416 – 421. 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS.

Oseledets, I. (2011). Tensor-train decomposition. *SIAM Journal of Scientific Computing*, 33(5):2295–2317.

Pangalos, G. (2016). *Model-based controller design methods for heating systems*. Dissertation, TU Hamburg, Hamburg.

Pangalos, G., Eichler, A., and Lichtenberg, G. (2015). *Hybrid Multilinear Modeling and Applications*, pages 71–85. Springer International Publishing, Cham.

Röbenack, K. (2005). Automatic differentiation and nonlinear controller design by exact linearization. *Future Generation Computer Systems*, 21:1372–1379.

Vervliet, N., Debals, O., Sorber, L., Van Barel, M., and De Lathauwer, L. (2016). Tensorlab 3.0. Available online.

APPENDIX

In the Appendix the proofs of the arithmetic operations, i.e. multiplication, differentiation and Lie derivative, for polynomials in tensor representation is given. Furthermore, the proofs of th eouter product of CP tensors and the control law for the feedback linearizing controller for CP decomposed MTI systems are presented here.

Proof of the Outer Product

Proof 1. *Inserting the elementwise descriptions of* X *and* Y *into the outer product (2), leads to*

$$
\begin{aligned}
&z(i_1,\dots,i_n,j_1,\dots,j_m) = x(i_1,\dots,i_n)y(j_1,\dots,j_m)\\
&=\sum_{k=1}^{r_X}\lambda_X(k)u_1(i_1,k)\cdots u_n(i_n,k)\\
&\quad\cdot\sum_{k=1}^{r_Y}\lambda_Y(k)v_1(j_1,k)\cdots v_m(j_m,k)\\
&=\lambda_X(1)\lambda_Y(1)u_1(i_1,1)\cdots u_n(i_n,1)v_1(j_1,1)\cdots v_m(j_m,1)\\
&\quad+\lambda_X(1)\lambda_Y(2)u_1(i_1,1)\cdots u_n(i_n,1)v_1(j_1,2)\cdots v_m(j_m,2)\\
&\quad+\cdots+\lambda_X(r_X)\lambda_Y(r_Y)u_1(i_1,r_X)\cdots v_m(j_m,r_Y).
\end{aligned}
$$

Comparing this representation of the outer product with the elementwise description of the resulting ten-

sor in CP decomposition

$$z(i_1,\dots,j_m)=\sum_{k=1}^{r_X+r_Y}\lambda_Z(k)w_1(i_1,k)\cdots w_{n+m}(j_m,k),$$

leads to the factors in (12) to (14). □

Proof of the Multiplication

Proof 2. *Considering (35) the monomial tensor of the result can be decomposed by*

$$\left\langle \mathsf{H}_1\circ\mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1+N_2}(\mathbf{x}) \right\rangle=\left\langle \mathsf{H}_1\circ\mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x})\circ\mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle.$$

Using the elementwise representations (2) and (4) of outer and contracted product and rearranging gives

$$\begin{aligned}&\left\langle \mathsf{H}_1\circ\mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x})\circ\mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle\\ &=\sum_{i_1=1}^{2}\cdots\sum_{i_{nN_1}=1}^{2}\sum_{j_1=1}^{2}\cdots\sum_{j_{nN_2}=1}^{2}h_1(i_1,\dots,i_{nN_1})h_2(j_1,\dots,j_{nN_2})\\ &\cdot m_p^{N_1}(\mathbf{x})(i_1,\dots,i_{nN_1})m_p^{N_2}(\mathbf{x})(j_1,\dots,j_{nN_2})\\ &=\sum_{i_1=1}^{2}\cdots\sum_{i_{nN_1}=1}^{2}h_1(i_1,\dots,i_{nN_1})m_p^{N_1}(\mathbf{x})(i_1,\dots,i_{nN_1})\\ &\cdot\sum_{j_1=1}^{2}\cdots\sum_{j_{nN_2}=1}^{2}h_2(j_1,\dots,j_{nN_2})m_p^{N_2}(\mathbf{x})(j_1,\dots,j_{nN_2})\\ &=\left\langle \mathsf{H}_1 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \right\rangle\cdot\left\langle \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle\\ &=h_1(\mathbf{x})\cdot h_2(\mathbf{x})\quad\square\end{aligned}$$

Proof of the Differentiation

Proof 3. *All terms depending on the variables x_i, with $i=1,\dots,n$ of a polynomial in tensor form are inside the monomial tensor. The parameter tensor H contains constant elements only. Thus, the partial derivative of the monomial tensor is investigated first. In the multilinear case the partial derivative with respect to one variable x_j can be found using the product rule of differentiation as stated in Proposition 1*

$$\begin{aligned}\frac{\partial}{\partial x_j}\mathsf{M}(\mathbf{x})&=[\mathbf{w}_1,\dots,\mathbf{w}_n]=\frac{\partial}{\partial x_j}\left(\begin{pmatrix}1\\x_n\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_1\end{pmatrix}\right)\\ &=\begin{pmatrix}1\\x_n\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_{j+1}\end{pmatrix}\circ\begin{pmatrix}0\\1\end{pmatrix}\circ\begin{pmatrix}1\\x_{j-1}\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_1\end{pmatrix}.\end{aligned}\tag{52}$$

The factor matrices of the derivative of the monomial tensor reads

$$\mathbf{w}_i=\begin{cases}\begin{pmatrix}0 & 1\end{pmatrix}^T & \text{, for } i=n-j+1,\\ \begin{pmatrix}1 & x_{n-i+1}\end{pmatrix}^T & \text{, else.}\end{cases}$$

The differentiation influences the dimension belonging to the differentiation variable x_j of the monomial tensor only. Thus, the differentiation can be expressed in terms of a k-mode product

$$\frac{\partial}{\partial x_j}\mathsf{M}(\mathbf{x})=\mathsf{M}(\mathbf{x})\times_{n-j+1}\Theta^T,$$

setting the factor matrix of $\mathsf{M}(\mathbf{x})$ belonging x_j to $\begin{pmatrix}0 & 1\end{pmatrix}^T$.
Since the partial derivative can be found for the multilinear monomial tensor, the concept is extended to polynomials with higher monomial orders $N\geq 1$. In contrast to the multilinear case, the variable x_j does not occur in one factor matrix only but in N factor matrices. According to the product rule of differentiation the differentiation of $\mathsf{M}_p^N(\mathbf{x})$ leads to

$$\frac{\partial}{\partial x_j}\mathsf{M}_p^N(\mathbf{x})=\sum_{k=1}^{N}\left[\mathbf{w}_1^k,\dots,\mathbf{w}_{nN}^k\right],$$

with

$$\mathbf{w}_i^k=\begin{cases}\begin{pmatrix}0 & 1\end{pmatrix}^T & \text{, for } i=nk-j+1,\\ \begin{pmatrix}1 & x_{nk-i+1}\end{pmatrix}^T & \text{, else.}\end{cases}$$

As before, this change in the factor matrices can be expressed by k-mode product leading to the partial derivative of the monomial tensor

$$\frac{\partial}{\partial x_j}\mathsf{M}_p^N(\mathbf{x})=\sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x})\times_{kn-j+1}\Theta^T.\tag{53}$$

With the derivative of the monomial tensor (53) the derivative of the function is given by

$$\begin{aligned}\frac{\partial}{\partial x_j}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle&=\left\langle \mathsf{H} \,\middle|\, \sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x})\times_{kn-j+1}\Theta^T \right\rangle\\ &\overset{!}{=}\left\langle \mathsf{H}_j \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle.\end{aligned}\tag{54}$$

To find the parameter tensor H_j, (54) can be written as

$$\begin{aligned}\left\langle \mathsf{H} \,\middle|\, \sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x})\times_{kn-j+1}\Theta^T \right\rangle&=\\ \sum_{k=1}^{N}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x})\times_{kn-j+1}\Theta^T \right\rangle&,\end{aligned}\tag{55}$$

because of the linearity property of the inner product. The elements of the k-mode product read

$$\begin{aligned}&\left(\mathsf{M}_p^N(\mathbf{x})\times_l\Theta^T\right)(i_1,\dots,l,\dots,i_{nN})\\ &=m_p^N(i_1,\dots,1,\dots,i_{nN})\Theta^T(l,1),\end{aligned}$$

because $\Theta^T(l,2)=0$, $l=1,2$. With that, the terms of the sum in (55) are written as

$$\begin{aligned}&\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x})\times_l\Theta^T \right\rangle\\ &=\sum_{i_1=1}^{2}\cdots\sum_{i_{nN}=1}^{2}h(i_1,\dots,i_{nN})m_p^N(i_1,\dots,1,\dots,i_{nN})\Theta^T(l,1).\end{aligned}$$

The aim is to isolate the monomial tensor on the right side of the contracted product such that

$$\langle \mathsf{H} \,|\, \mathsf{M}_p^N(\mathbf{x}) \times_l \Theta^T \rangle \overset{!}{=} \langle \tilde{\mathsf{H}} \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle = \sum_{i_1=1}^{2} \cdots \sum_{i_{nN}=1}^{2} \tilde{h}(i_1,\dots,i_{nN}) m_p^N(i_1,\dots,i_{nN}).$$

Therefore, the elements of $\tilde{\mathsf{H}}$ *are given by*

$$\tilde{h}(i_1,\dots,i_l,\dots,i_{nN}) = \begin{cases} h(i_1,\dots,2,\dots,i_{nN}), & \text{for } i_l=1, \\ 0 & \text{, for } i_l=2, \end{cases}$$

since

$$\sum_{i_l=1}^{2} h(i_1,\dots,i_l,\dots,i_{nN}) \Theta^T(l,1) = h(i_1,\dots,2,\dots,i_{nN}).$$

Using Definition 4 of the k-mode product, the parameter tensor $\tilde{\mathsf{H}}$ *can be directly computed by*

$$\tilde{\mathsf{H}} = \mathsf{H} \times_l \Theta,$$

leading to

$$\langle \mathsf{H} \,|\, \mathsf{M}_p^N(\mathbf{x}) \times_l \Theta^T \rangle = \langle \mathsf{H} \times_l \Theta \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle .$$

Inserting this to (55) shows that the partial derivative of a polynomial $h(\mathbf{x})$ *is computed by*

$$\frac{\partial}{\partial x_j} \langle \mathsf{H} \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle = \sum_{k=1}^{N} \langle \mathsf{H} \times_{kn-j+1} \Theta \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle = \left\langle \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \Theta \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,$$

such that the parameter tensor of the derivative reads

$$\mathsf{H}_j = \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \Theta. \quad \square$$

Proof of the Lie Derivative

Proof 4. *The first Lie derivative, i.e.* $l=1$, *of* $g(\mathbf{x})$ *along* $\mathbf{h}(\mathbf{x})$ *is defined by*

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} g(\mathbf{x}). \tag{56}$$

Since the scalar function $g(\mathbf{x})$ *is given as tensor function, equation (41) is applied to get the partial derivative*

$$\frac{\partial}{\partial x_i} g(\mathbf{x}) = \left\langle \sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \Theta \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, i=1,\dots,n.$$

The multiplication with $h_i(\mathbf{x}) = \langle \mathsf{H}_i \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle$, *where* $i=1,\dots,n$ *using (38) yields*

$$h_i(\mathbf{x}) \frac{\partial}{\partial x_i} g(\mathbf{x}) = \left\langle \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \Theta \right) \middle| \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle .$$

Summing up these elements to get the Lie derivative (56) leads to

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} \left\langle \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \Theta \right) \middle| \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle = \left\langle \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \Theta \right) \middle| \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle .$$

This approach is extended to multiple Lie derivatives along $\mathbf{h}(\mathbf{x})$ *as given by (45) for arbitrary* $l \in \mathbb{N}$ *by*

$$L_{\mathbf{h}}^l g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{h}}^{l-1} g(\mathbf{x})$$
$$= \sum_{i=1}^{n} \langle \mathsf{H}_i \,|\, \mathsf{M}_p^N(\mathbf{x}) \rangle \cdot \left\langle \sum_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \Theta \middle| \mathsf{M}_p^{lN}(\mathbf{x}) \right\rangle$$
$$= \left\langle \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \Theta \right) \middle| \mathsf{M}_p^{(l+1)N}(\mathbf{x}) \right\rangle . \square$$

Proof of the Feedback Linearizing Controller

Proof 5. *Inserting the tensor representation of the Lie derivatives defined in Lemma 1 into the controller function (30) and rearranging leads to*

$$u = \frac{-\sum_{i=0}^{\rho} \mu_i \langle \mathsf{L}_{\mathsf{A},\mathsf{C},i} \,|\, \mathsf{M}_p^{i+1}(\mathbf{x}) \rangle + \mu_0 r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho-1} \middle| \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle} = \frac{-\left\langle \sum_{i=0}^{\rho} \mu_i \mathsf{L}_{\mathsf{A},\mathsf{C},i} \middle| \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle + \mu_0 r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho-1} \middle| \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle}. \quad \square$$