URN (Paper): urn:nbn:de:gbv:ilm1-2014iwk-125:0

58th ILMENAU SCIENTIFIC COLLOQUIUM
Technische Universität Ilmenau, 08 – 12 September 2014
URN: urn:nbn:de:gbv:ilm1-2014iwk:3

NONLINEAR CONTROL OF COMPLEX SYSTEMS USING ALGORITHMIC DIFFERENTIATION

Klaus Röbenack, Jan Winkler, Mirko Franke

Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Institut für Regelungs- und Steuerungstheorie
D-01062 Dresden, Germany
Email: {klaus.roebenack,jan.winkler}@tu-dresden.de, mirko.franke@mailbox.tu-dresden.de

ABSTRACT

In mechatronics and robotics one often encounters highly complex nonlinear systems, whose dynamics are described by a system of nonlinear ordinary differential equations. In simulation, one can use very sophisticated models, whereas controllers are typically designed on strongly simplified models. Quite often, this results in linear controllers which are unsuitable for the original nonlinear system. We suggest an approach allowing nonlinear controller design even for very complicated models. Our approach is based on an alternative differentiation method called algorithmic or automatic differentiation. With this computation method we can circumvent problems that may arise in symbolic computation.

Index Terms— Nonlinear control, algorithmic differentiation, automatic differentiation, operator overloading, differential geometry, Lie derivatives, tank reactor

1. INTRODUCTION

In mechatronics and robotics one often encounters highly complex nonlinear systems, whose dynamics can be written as a system of ordinary differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}), \tag{1}$$

provided appropriate minimal coordinates are found. The state vector $\boldsymbol{x}$ describes the actual position in this coordinate frame. System (1) can be influenced via the input $\boldsymbol{u}$. We introduce an additional variable $\boldsymbol{y}$ representing the output of the system (e.g. measurement or control variable):

$$\boldsymbol{y} = \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{u}). \tag{2}$$

There are several approaches how to deal with a nonlinear control system (1)-(2). A very promising direction is the use of differential geometry. Many specify control problems have been solved using differential geometric or differential algebraic concepts. These approaches often require some types of Lie derivatives [1, 2].

The computation of Lie derivatives occurring in nonlinear control laws is usually carried out symbolically [3–6]. An alternative approach using automatic/algorithmic differentiation has been suggested in [7–9]. An efficient implementation of these algorithms has been presented in [10]. Based on this implementation, we discuss an application on a continuous-stirred tank reactor [11].

Equations (1) and (2) provide a very general description of nonlinear systems. For several classes of systems, the nonlinear map $\boldsymbol{F}$ occurring in (1) is affine w.r.t. the input. For most examples the output map $\boldsymbol{H}$ in (2) does not depend explicitly on the input $\boldsymbol{u}$, i.e., the system does not have a direct throughput.

The paper is structured as follows: In Section 2 we remind the reader of the linearization by feedback approach from nonlinear control theory. The computation of derivatives is discussed in Section 3. Our approach is illustrated on an example system in Section 4. We finally provide a summary in Section 5.

2. NONLINEAR CONTROL SYSTEMS

2.1. Single-Input Single-Output Systems

An input-affine single-input single-output system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u, \quad y = h(\boldsymbol{x}) \tag{3}$$

is described by the drift vector field $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, the input vector field $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ and the output map $h : \mathbb{R}^n \to \mathbb{R}$, which is a scalar field. We assume that these fields are sufficiently smooth. Modern control algorithms often require Lie derivatives of these different types of fields. In particular, the exact input-output linearization by feedback employs Lie derivatives of the scalar field h along the vector field $\boldsymbol{f}$ or $\boldsymbol{g}$, which are defined by

$$L_{\boldsymbol{f}}h(\boldsymbol{x}) = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\boldsymbol{f}(x) \quad \text{and} \quad L_{\boldsymbol{g}}h(\boldsymbol{x}) = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\boldsymbol{g}(x), \tag{4}$$

respectively. Higher order Lie derivatives along the same vector field are defined by the recursion

$$L_{\boldsymbol{f}}^{k+1}h(\boldsymbol{x}) = \frac{\partial L_{\boldsymbol{f}}^{k}h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\boldsymbol{f}(\boldsymbol{x}) \quad \text{with} \quad L_{\boldsymbol{f}}^{0}h(\boldsymbol{x}) = h(\boldsymbol{x}), \tag{5}$$

see [1]. Similarly, we can also define the Lie derivative of a scalar field along different vector fields such as

$$L_{\boldsymbol{g}}L_{\boldsymbol{f}}h(\boldsymbol{x}) = \frac{\partial L_{\boldsymbol{f}}h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\boldsymbol{g}(\boldsymbol{x}). \tag{6}$$

Definition 2.1 ([1,2]) *System* (3) *is said to have* relative degree r *at the point* $\boldsymbol{x}^0 \in \mathbb{R}^n$ *if*

1. $L_{\boldsymbol{g}}L_{\boldsymbol{f}}^{k}h(\boldsymbol{x}) = 0$ *for all* $\boldsymbol{x}$ *in a neighborhood of* $\boldsymbol{x}^0$ *and all* $k = 0, \ldots, r-2$*, and*
2. $L_{\boldsymbol{g}}L_{\boldsymbol{f}}^{r}h(\boldsymbol{x}^0) \neq 0$.

The first order time derivative of the output can be expressed in terms of Lie derivatives:

$$\begin{aligned}
\dot{y}(t) &= \frac{\mathrm{d}}{\mathrm{d}\,t}y(t) \\
&= \frac{\mathrm{d}}{\mathrm{d}\,t}h(\boldsymbol{x}(t)) \\
&= \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\dot{\boldsymbol{x}}(t) \\
&= \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\left(\boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{g}(\boldsymbol{x}(t))\,u(t)\right) \\
&= L_{\boldsymbol{f}}h(\boldsymbol{x}(t)) + L_{\boldsymbol{g}}h(\boldsymbol{x}(t))\,u(t).
\end{aligned}$$

In case of $L_{\boldsymbol{g}}h(\boldsymbol{x}^0) \neq 0$ we have relative degree $r = 1$. Otherwise, i.e., with $L_{\boldsymbol{g}}h(\boldsymbol{x}) \equiv 0$, we compute the second order time derivative

$$\begin{aligned}
\ddot{y}(t) &= \frac{\mathrm{d}}{\mathrm{d}\,t}\dot{y}(t) \\
&= \frac{\mathrm{d}}{\mathrm{d}\,t}L_{\boldsymbol{f}}h(\boldsymbol{x}(t)) \\
&= \frac{\partial L_{\boldsymbol{f}}h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\dot{\boldsymbol{x}}(t) \\
&= \frac{\partial L_{\boldsymbol{f}}h(\boldsymbol{x})}{\partial \boldsymbol{x}}\,\left(\boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{g}(\boldsymbol{x}(t))\,u(t)\right) \\
&= L_{\boldsymbol{f}}^{2}h(\boldsymbol{x}(t)) + L_{\boldsymbol{g}}L_{\boldsymbol{f}}h(\boldsymbol{x}(t))\,u(t).
\end{aligned}$$

If $L_{\boldsymbol{g}}L_{\boldsymbol{f}}h(\boldsymbol{x}^0) \neq 0$ we have relative degree $r = 2$. Otherwise, we continue this process and calculate further time derivatives of the output along the trajectory of system (3). In general, the time derivatives of the output can be written as

$$\begin{aligned}
y(t) &= h(\boldsymbol{x}(t)) \\
\dot{y}(t) &= L_{\boldsymbol{f}}h(\boldsymbol{x}(t)) \\
&\vdots \\
y^{(r-1)}(t) &= L_{\boldsymbol{f}}^{r-1}h(\boldsymbol{x}(t)) \\
y^{(r)}(t) &= L_{\boldsymbol{f}}^{r}h(\boldsymbol{x}(t)) + L_{\boldsymbol{g}}L_{\boldsymbol{f}}^{r-1}h(\boldsymbol{x}(t))\,u(t).
\end{aligned} \tag{7}$$

Roughly speaking, the relative degree is the minimum order of a time derivative of the output which depends directly on the input u. For a well-defined relative degree, this fact can be formulated as

$$r = \arg\min_k \left\{ \frac{\mathrm{d}\, y^{(k)}}{\mathrm{d}\, u} \neq 0 \right\}.$$

The last line of (7) can be employed to carry out an exact linearization by feedback. For this reason, we define a virtual input v by

$$y^{(r)} = L_{\boldsymbol{f}}^r h(\boldsymbol{x}) + L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{r-1} h(\boldsymbol{x})\, u \stackrel{!}{=} v. \tag{8}$$

Resolving (8) w.r.t. the original input u yields the state-feedback

$$u = \frac{v - L_{\boldsymbol{f}}^r h(\boldsymbol{x})}{L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{r-1} h(\boldsymbol{x})}. \tag{9}$$

The resulting input-output behavior $y^{(r)} = v$ corresponds to a chain of r integrators, i.e., we obtain a linear controllable system. The coordinates of the associated controller canonical form are given by the output and its time derivatives $y, \ldots, y^{(r-1)}$. To obtain stable input-output dynamics described by a given characteristic polynomial

$$\rho(s) = s^r + p_{r-1} s^{r-1} + \cdots + p_1 s + p_0 \tag{10}$$

we use the additional feedback

$$v = -p_0\, y - p_1\, \dot{y} - \cdots - p_{r-1} y^{(r-1)}. \tag{11}$$

The time derivatives of y occurring in (11) can be expressed in terms of Lie derivatives, see (7). Combining (9) and (11) yields the linearizing and stabilizing feedback law

$$u = -\frac{1}{L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{r-1} h(\boldsymbol{x})} \sum_{k=0}^{r} p_k\, L_{\boldsymbol{f}}^k h(\boldsymbol{x}) \quad \text{with} \quad p_r = 1. \tag{12}$$

For $r < n$ we linearize the system's dynamics only w.r.t. an subsystem. In this case, the feedback derived above achieves a partial linearization. For $r = n$, we obtain a completely linear system, i.e., the feedback law carries out a full linearization.

2.2. Multi-Input Multi-Output Systems

The dynamics of an input-affine multi-input multi-output system is given by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \sum_{i=1}^{m} \boldsymbol{g}_i(\boldsymbol{x}) u_i, \quad \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{13}$$

with the vector fields $\boldsymbol{f}, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_m : \mathbb{R}^n \to \mathbb{R}^n$ and the vector-valued map $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$. The output map $\boldsymbol{h}$ describes the m components of the output $\boldsymbol{y} = (y_1, \ldots, y_m)^T$ by m scalar fields $h_1, \ldots, h_m : \mathbb{R}^n \to \mathbb{R}$, i.e., $y_i = h_i(\boldsymbol{x})$ for $i = 1, \ldots, m$.

Definition 2.2 ([1,2]) *System* (13) *is said to have a (vector valued)* relative degree $(r_1, \ldots, r_m)$ *at the point* $\boldsymbol{x}^0 \in \mathbb{R}^n$ *if*

1. $L_{\boldsymbol{g}_j} L_{\boldsymbol{f}}^k h_i(\boldsymbol{x}) = 0$ *for all* $\boldsymbol{x}$ *in a neighborhood of* $\boldsymbol{x}^0$ *and all* $j = 1, \ldots, m$ *as well as* $k = 0, \ldots, r_i - 2$ *with* $i = 1, \ldots, m$, *and*

2. *the so-called* decoupling matrix

$$A(\boldsymbol{x}^0) = \begin{pmatrix} L_{\boldsymbol{g}_1} L_{\boldsymbol{f}}^{r_1-1} h_1(\boldsymbol{x}^0) & \cdots & L_{\boldsymbol{g}_m} L_{\boldsymbol{f}}^{r_1-1} h_1(\boldsymbol{x}^0) \\ \vdots & \ddots & \vdots \\ L_{\boldsymbol{g}_1} L_{\boldsymbol{f}}^{r_m-1} h_m(\boldsymbol{x}^0) & \cdots & L_{\boldsymbol{g}_m} L_{\boldsymbol{f}}^{r_m-1} h_m(\boldsymbol{x}^0) \end{pmatrix} \tag{14}$$

is regular (i.e., nonsingular).

The feedback linearization was carried out in the single-input single-output case using the last line in (7). In the multi-input multi-output case, this corresponds to

$$\begin{pmatrix} y^{(r_1)} \\ \vdots \\ y^{(r_m)} \end{pmatrix} = \underbrace{\begin{pmatrix} L_{\boldsymbol{f}}^{r_1} h_1(\boldsymbol{x}) \\ \vdots \\ L_{\boldsymbol{f}}^{r_m} h_1(\boldsymbol{x}) \end{pmatrix}}_{\boldsymbol{b}(\boldsymbol{x})} + \underbrace{\begin{pmatrix} L_{\boldsymbol{g}_1} L_{\boldsymbol{f}}^{r_1-1} h_1(\boldsymbol{x}) & \cdots & L_{\boldsymbol{g}_m} L_{\boldsymbol{f}}^{r_1-1} h_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ L_{\boldsymbol{g}_1} L_{\boldsymbol{f}}^{r_m-1} h_m(\boldsymbol{x}) & \cdots & L_{\boldsymbol{g}_m} L_{\boldsymbol{f}}^{r_m-1} h_m(\boldsymbol{x}) \end{pmatrix}}_{A(\boldsymbol{x})} \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}. \tag{15}$$

Similar as in Section 2.1 we introduce a virtual variable $\boldsymbol{v} = (v_1, \ldots, v_m)^T$, for which we want to achieve a linear input-output behavior. The feedback

$$\boldsymbol{u} = A^{-1}(\boldsymbol{x}) \left(\boldsymbol{v} - \boldsymbol{b}(\boldsymbol{x}) \right), \tag{16}$$

which is a generalization of (9), decomposes the input-output dynamics into m chains of integrators, i.e., $y^{(r_i)} = v_i$ for $i = 1, \ldots, m$. Each subsystem is linear and controllable. Furthermore, the feedback (16) also achieves a decoupling between the subsystems. These subsystems can be stabilized separately as in the single-input single-output case, see (12). If $r_1 + \cdots r_m < n$ we have a partial linearization, for $r_1 + \cdots r_m = n$ we achieve a full linearization.

Note that the vector-valued map $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ should not be confused with a vector field. As mention above, $\boldsymbol{h}$ can be seen as a collection of m scalar fields, i.e., $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))^T$. In this sense we extend the definition in (4) to the vector-valued case by

$$L_{\boldsymbol{f}} \boldsymbol{h}(\boldsymbol{x}) = \frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{x}} \, \boldsymbol{f}(x) = \begin{pmatrix} L_{\boldsymbol{f}} h_1(\boldsymbol{x}) \\ \vdots \\ L_{\boldsymbol{f}} h_m(\boldsymbol{x}) \end{pmatrix}, \tag{17}$$

which is simply the collection of the Lie derivatives $L_{\boldsymbol{f}} h_1(\boldsymbol{x}), \ldots, L_{\boldsymbol{f}} h_m(\boldsymbol{x})$ of the scalar fields $h_1, \ldots, h_m$. This notation is useful in the context of software implementations, see [12, 13].

Similarly, we can combine the input vector fields $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ into an $n \times m$ matrix

$$G(\boldsymbol{x}) = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m).$$

The Lie derivative of the vector-valued map $\boldsymbol{h}$ along the matrix G can be defined column-wise by

$$L_G \boldsymbol{h}(\boldsymbol{x}) = \left(L_{\boldsymbol{g}_1} \boldsymbol{h}(\boldsymbol{x}), \ldots, L_{\boldsymbol{g}_m} \boldsymbol{h}(\boldsymbol{x}) \right) \tag{18}$$

as a collection of the Lie derivatives $L_{\boldsymbol{g}_1} \boldsymbol{h}(\boldsymbol{x}), \ldots, L_{\boldsymbol{g}_m} \boldsymbol{h}(\boldsymbol{x})$ in the sense of (17).

Definition 2.3 ([14, 15]) *Assume system* (13) *has a well-defined relative degree* $(r_1, \ldots, r_m)$. *System* (13) *is said to have* uniform relative degree r *if* $r = r_i$ *for all* $i = 1, \ldots, m$, *i.e., if the relative degrees are the same for all outputs.*

If system (13) has a uniform relative degree r, we have $\boldsymbol{b}(\boldsymbol{x}) = L_{\boldsymbol{f}}^r \boldsymbol{h}(\boldsymbol{x})$ and $A(\boldsymbol{x}) = L_G L_{\boldsymbol{f}}^{r-1} \boldsymbol{h}(\boldsymbol{x})$. In this case, the linearizing feedback (16) can be written as

$$\boldsymbol{u} = \left(L_G L_{\boldsymbol{f}}^{r-1} \boldsymbol{h}(\boldsymbol{x}) \right)^{-1} \left(\boldsymbol{v} - L_{\boldsymbol{f}}^r \boldsymbol{h}(\boldsymbol{x}) \right).$$

2.3. Dynamic Extension

Assume that $y_1^{(r_1)}, \ldots, y_m^{(r_m)}$ are minimum order time derivatives of the outputs $y_1, \ldots, y_m$ that depend directly on some inputs u_i. The tuple $(r_1, \ldots, r_m)$ does not constitute a relative degree if the decoupling matrix (14) is singular. This possibility is illustrated by the following example.

Consider the 3-dimensional system

$$\begin{aligned} \dot{x}_1 &= \sin x_3 + u_1 \\ \dot{x}_2 &= \cos x_3 + u_1 \\ \dot{x}_3 &= u_2 \\ y_1 &= x_1 \\ y_2 &= x_2 \end{aligned} \tag{19}$$

with the two inputs u_1, u_2 and the two outputs y_1, y_2. The tuple $(r_1, r_2) = (1, 1)$ corresponds to the time derivatives

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \sin x_3 \\ \cos x_3 \end{pmatrix} + \underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}}_{A(\boldsymbol{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{20}$$

The associated decoupling matrix is singular due to the zero column in the second position. However, we can circumvent this problem with an additional integrator

$$\dot{u}_1 = u_1^*,$$

where we treat u_1 as an additional state, and u_1^* replaces the input u_1. Further differentiation of (20) yields

$$\begin{array}{rcccl} \ddot{y}_1 & = & \dot{x}_3 \cos x_3 + \dot{u}_1 & = & u_2 \cos x_3 + u_1^* \\ \ddot{y}_2 & = & -\dot{x}_3 \sin x_3 + \dot{u}_1 & = & -u_2 \sin x_3 + u_1^*, \end{array} \tag{21}$$

which can be rewritten as

$$\begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \begin{pmatrix} 1 & \cos x_3 \\ 1 & -\sin x_3 \end{pmatrix} \begin{pmatrix} u_1^* \\ u_2 \end{pmatrix}. \tag{22}$$

The augmented 4-dimensional system has the well-defined relative degree $(2, 2)$ whenever $\sin x_3 \neq \cos x_3$. The well-defined relative degree was achieved with an additional integrator. This approach is called *dynamics extension* [13, 16].

3. ALGORITHMIC DIFFERENTIATION

3.1. Methods to Compute Derivatives

For an explicitly given function, symbolic differentiation is probably he most common method to compute derivatives. This approach uses elementary differentiation rules in combination with the chain rule. For symbolic differentiation, there are several computer algebra packages available. However, symbolic differentiation has also limitations. Symbolic differentiation is usually not applicable if the function under consideration is given as an algorithm containing control structures such as loops and branches. Furthermore, the size of symbolically computed derivative expressions may increase exponentially for higher order derivatives.

Another standard method to obtain derivative values is numeric differentiation using finite difference schemes. This method can be applied to functions described by highly complicated algorithms. While this method can easily be implemented, the accuracy of the resulting derivative values is significantly lower than the accuracy of the associated function values, see [17].

An interesting alternative to the above mentioned derivative computation methods is called *algorithmic* or *automatic differentiation* [17]. This method is applicable to very general algorithmic descriptions of a given function. Similar to symbolic differentiation, elementary differentiation rules are applied in connection with the chain rule. However, the derivatives are evaluated during each step, i.e., the intermediate results are floating point numbers and not symbolic expressions. The most common approaches to implement algorithmic differentiation are source code transformation and operator overloading [18, 19]. The second approach can be used for object-oriented programming languages.

3.2. Forward Mode and Taylor Arithmetic

Consider a smooth map $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m$, which can be given as an algorithm. In addition, we consider a curve $\boldsymbol{x}$ in $\mathbb{R}^n$ given by a Taylor series

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \boldsymbol{x}_1 t + \boldsymbol{x}_2 t^2 + \cdots + x_d t^d + \mathcal{O}(t^{d+1}) \tag{23}$$

with vector-valued Taylor coefficients $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_d \in \mathbb{R}^n$. The curve $\boldsymbol{x}$ is mapped via $\boldsymbol{F}$ from the vector space $\mathbb{R}^n$ into a curve $\boldsymbol{z}$ of the vector space $\mathbb{R}^m$ given by

$$\boldsymbol{z}(t) = \boldsymbol{F}(\boldsymbol{x}(t)) = \boldsymbol{z}_0 + \boldsymbol{z}_1 t + \boldsymbol{z}_2 t^2 + \cdots + \boldsymbol{z}_d t^d + \mathcal{O}(t^{d+1}) \tag{24}$$

with Taylor coefficients $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_d \in \mathbb{R}^m$. The algorithm (in its original form) computes the Taylor coefficient $\boldsymbol{z}_0$ as the function value at the point $\boldsymbol{x}_0$, i.e.,

$$\boldsymbol{z}_0 = \boldsymbol{F}(\boldsymbol{x}_0). \tag{25}$$

The time derivative of (24) at $t = 0$ results in a formula for the next Taylor coefficient

$$\boldsymbol{z}_1 = \boldsymbol{F}'(\boldsymbol{x}_0)\,\boldsymbol{x}_1. \tag{26}$$

Assume that the algorithm under consideration is formulated in the programming language C++. The variables would usually be represented by the build-in numerical type `double`. It is easy to replace this numerical standard type by a new class (say `ddouble`) containing both the function value and the derivative value:

```
class ddouble
{
 double val;  // function value
 double der;  // derivative value
}
```

If we overload elementary functions (e.g. $\sin$, $\cos$, $\exp$) and operations (e.g. $+$, $-$, $*$, $/$), we could use the same procedure not only for the computation of the function value $\boldsymbol{z}_0$ as given in (25) but also (and simultaneously) for the computation of the directional derivative $\boldsymbol{z}_1$ in (26).

In general, the Taylor coefficients of the curve $\boldsymbol{z}$ are given by

$$\boldsymbol{z}_k = \left.\frac{d^k}{dt^k}\boldsymbol{z}(t)\right|_{t=0}. \tag{27}$$

Higher order Taylor coefficients $\boldsymbol{z}_2, \boldsymbol{z}_3, \ldots$ cannot be expressed anymore in the convenient matrix-vector notation [17]. However, these Taylor coefficients can easily be computed simply replacing the floating point type `double` by a new class (say `tdouble`) which contains not only one value but all $(d+1)$ Taylor coefficients:

```
const int d=...;  // highest degree d
class tdouble
{
 public:
 double coeff[d+1];  // array for the coefficients
}
```

Again, one has to overload elementary functions and operations. For instance, the multiplication of two (scalar) curves $x(t)$ and $y(t)$ results in the new curve

$$\begin{aligned} z(t) &= x(t)\cdot y(t) \\ &= (x_0 + x_1 t + x_2 t^2 + \cdots)\cdot(y_0 + y_1 t + y_2 t^2 + \cdots) \\ &= (x_0 + y_0) + (x_0 y_1 + x_1 y_0)t + (x_0 y_2 + x_1 y_2 + x_2 y_0)t^2 + \cdots \\ &= z_0 + z_1 t + z_2 t^2 + \cdots, \end{aligned}$$

where the Taylor coefficients z_k of the new curve z are given by the convolution

$$z_k = \sum_{i=0}^{k} x_i y_{k-i} \quad \text{for} \quad k = 0, \ldots, d.$$

Similar rules can be found for all usual operations, see [20]. These (new) operations are known as *Taylor arithmetic*.

3.3. Ordinary Differential Equations and Lie Derivatives

The Taylor arithmetic described in Section 3.2 can be used for the efficient series expansion of the solution of ordinary differential equations [21–23]. Consider the initial value problem

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{R}^n \tag{28}$$

of an unforced differential equation given by the vector field $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$. For a smooth vector field $\boldsymbol{f}$, the solution of (28) can be expanded into a Taylor series (23). The time derivative of the curve $\boldsymbol{x}$, which is the left-hand side of (28), is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{x}_1 + 2\boldsymbol{x}_2 t + 3\boldsymbol{x}_3 t^2 + 4\boldsymbol{x}_4 t^3 \cdots. \tag{29}$$

The right-hand side of (28) can be seen as a mapping from the curve $\boldsymbol{x}$ into a curve $\boldsymbol{z}$ with a Taylor expansion as in (24). For given Taylor coefficients $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$ of $\boldsymbol{x}$, the Taylor coefficients $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_k \in \mathbb{R}^n$ can be computed with Taylor arithmetic. Matching the coefficients of the curves $\dot{\boldsymbol{x}}$ in (29) and $\boldsymbol{z}$ as in (24) yields

$$\boldsymbol{x}_{k+1} = \frac{1}{k+1}\,\boldsymbol{z}_k, \tag{30}$$

which allows the recursive computation of all Taylor coefficients $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d \in \mathbb{R}^n$ starting with the initial value $\boldsymbol{x}_0$.

Now, we additionally take the output map $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ into account, which maps the curve $\boldsymbol{x}$ into a curve $\boldsymbol{y}$ given by

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) = \boldsymbol{y}_0 + \boldsymbol{y}_1 t + \boldsymbol{y}_2 t^2 + \cdots + \boldsymbol{y}_d t^d + \mathcal{O}(t^{d+1}) \tag{31}$$

with Taylor coefficients $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d \in \mathbb{R}^m$, which can directly be computed using Taylor arithmetic. Comparing the series expansion (31) with the time derivatives of the output curve discussed in Section 2 we obtain

$$L_{\boldsymbol{f}}^k \boldsymbol{h}(\boldsymbol{x}_0) = k!\,\boldsymbol{y}_k \quad \text{for} \quad k = 0, \ldots, d. \tag{32}$$

This relation allows the efficient computation of the function values of Lie derivatives using algorithmic differentiation [24–26]. The mixed Lie derivative needed for the feedback linearization can be computed based on Eq. (7). The computation of more general types of mixed Lie derivatives is discussed in [27, 28].

3.4. Packages ADOL-C and LieDrivers

ADOL-C is an algorithmic differentiation package based on the operator overloading capability of the object-oriented programming language C++ [29]. In this package, it is first necessary to generate an internal representation of the function under consideration. During this process, ADOL-C generates a sequential data structure called *tape*, which is accessed via a handle denoted as `tag`. Several types of derivatives (e.g. gradeints, Jacobians, Hessians) can be computed using appropriated driver functions, which are also provided by ADOL-C. These drivers access the internal data structure tape via its handle. More details can be found in [30].

Based on [24–26] a library for the efficient computation of different types of Lie derivatives was developed [10]. This library provides ADOL-C drivers and was therefore named LIEDRIVERS. We are currently working of the native integration of this library into ADOL-C.

As for our application, the LIEDRIVERS library provides drivers for the computation of the Lie derivatives (4)-(5) of a scalar field. The C function `Lie_scalarc` has the following arguments:

```
int Lie_scalarc(Tape_F,Tape_H,n,x0,d,res)
 short Tape_F;     // tape tag of vector field f
 short Tape_H;     // tape tag of scalar field f
 short n;          // dimension n
 double x0[n];     // vector x0
 short d;          // highest degree d
 double res[d+1]; // Lie derivatives
```

The case of multiple scalar fields defined by (17) is treated similarly. We also developed an interface to MATLAB®, where wrapper functions are provided for ADOL-C's build-in drivers and the LIEDRIVERS toolbox.

4. EXAMPLE

The Continuous-Stirred-Tank-Reactor (CSTR) model as a benchmark system in nonlinear control has been proposed in [11]. The CSTR, see Fig. 1, is continuously fed by a liquid flow containing the reactant A with the concentration c_{in} and the temperature ϑ_{in}. Within the CSTR a VAN-DER-VUSSE-reaction of the substances A, B, C and D occurs:

$$A \xrightarrow{k_1} B \xrightarrow{k_1} C, \qquad 2A \xrightarrow{k_2} D.$$

The liquid phase inside the reactor is supposed to be of constant volume and ideally mixed. The desired product of the reaction is B, while C and D are undesired by-products. The normalized flow rate u_1 and the cooling power u_2 are the available control inputs. The describing differential equations are given by

$$\begin{aligned}
\dot{c}_{\mathrm{A}} &= r_{\mathrm{A}}(c_{\mathrm{A}}, \vartheta) + (c_{\mathrm{in}} - c_{\mathrm{A}})u_1 \\
\dot{c}_{\mathrm{B}} &= r_{\mathrm{B}}(c_{\mathrm{A}}, c_{\mathrm{B}}, \vartheta) - c_{\mathrm{B}} u_1 \\
\dot{\vartheta} &= h(c_{\mathrm{A}}, c_{\mathrm{B}}, \vartheta) + \alpha(\vartheta_{\mathrm{C}} - \vartheta) + (\vartheta_{\mathrm{in}} - \vartheta)u_1 \\
\dot{\vartheta}_{\mathrm{C}} &= \beta(\vartheta - \vartheta_{\mathrm{C}}) + \gamma u_2
\end{aligned} \tag{33}$$

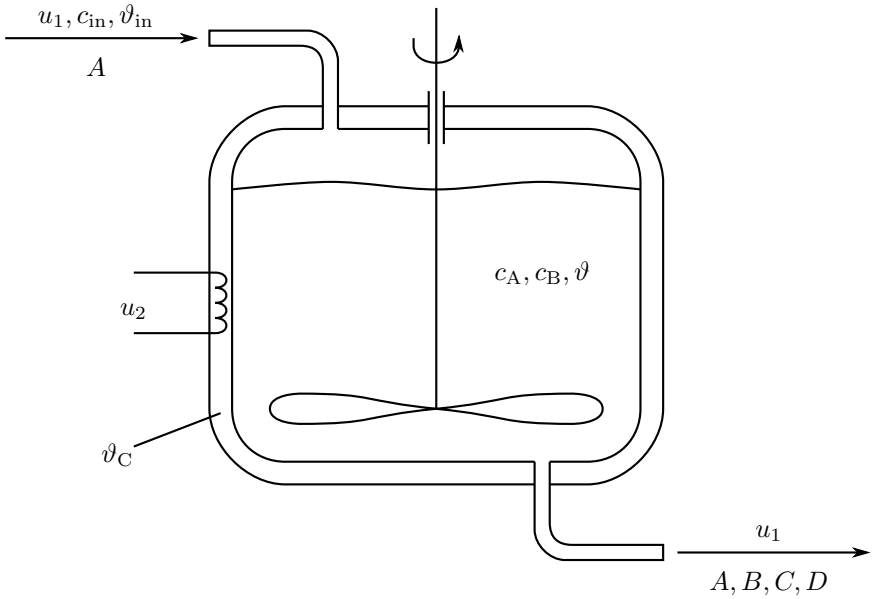


Fig. 1: Scheme of the Continuous-Stirred-Tank-Reactor

with c_{A} and c_{B}, the concentrations of A and B, ϑ and ϑ_{C}, the temperatures in the reactor and the cooling jacket, respectively. The concentrations c_{C} and c_{D} do not influence the dynamics (33) and thus they are not interesting. The reaction rates r_{A} and r_{B} and the contribution h to the reaction enthalpy are described by

$$\begin{aligned} r_{\mathrm{A}}(c_{\mathrm{A}}, \vartheta) &= -k_1(\vartheta)c_{\mathrm{A}} - k_2(\vartheta)c_{\mathrm{A}}^2 \\ r_{\mathrm{B}}(c_{\mathrm{A}}, c_{\mathrm{B}}, \vartheta) &= k_1(\vartheta)(c_{\mathrm{A}} - c_{\mathrm{B}}) \\ h(c_{\mathrm{A}}, c_{\mathrm{B}}, \vartheta) &= -\delta(k_1(\vartheta)\,(c_{\mathrm{A}}\Delta H_{\mathrm{AB}} + c_{\mathrm{B}}\Delta H_{\mathrm{BC}}) + k_2(\vartheta)c_{\mathrm{A}}^2\Delta H_{\mathrm{AD}}). \end{aligned}$$

The temperature-dependent functions k_1 and k_2 are of the ARRHENIUS-Type

$$k_i(\vartheta) = k_{i0}\exp\left(\frac{-E_i}{\vartheta/^{\circ}\mathrm{C} + 273.15}\right), \quad i = 1, 2 \tag{34}$$

with constant coefficients k_{i0}. The other symbols $\alpha, \beta, \gamma, \delta, E_1, E_2, \Delta H_{\mathrm{AB}}, \Delta H_{\mathrm{AD}}, \Delta H_{\mathrm{BC}}$ denote constant parameters adapted from [31].

With the state space vector $\boldsymbol{x} = (c_{\mathrm{A}}, c_{\mathrm{B}}, \vartheta, \vartheta_{\mathrm{C}})^T$ one can get a system of form (13) with

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} r_{\mathrm{A}}(x_1, x_3) \\ r_{\mathrm{B}}(x_1, x_2, x_3) \\ h(x_1, x_2, x_3) + \alpha(x_4 - x_3) \\ \beta(x_3 - x_4) \end{pmatrix}, \quad \boldsymbol{g}_1(\boldsymbol{x}) = \begin{pmatrix} c_{\mathrm{in}} - x_1 \\ -x_2 \\ (\vartheta_{\mathrm{in}} - x_3) \\ 0 \end{pmatrix}, \quad \boldsymbol{g}_2(\boldsymbol{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \gamma \end{pmatrix}. \tag{35}$$

In the following two controllers are suggested, applying the control schemes shown in Section 2.

4.1. Controller Design by Partial Feedback Linearization

As in [32] we first focus on the concentration part (x_1, x_2) of the system. For this subsystem a feedback linearization controller should be designed. We choose

$$y = h(x) = \frac{x_2}{c_{\mathrm{in}} - x_1} \tag{36}$$

as the output in the sense of a control variable. To specify the relative degree r we calculate time derivatives of the chosen output (36) until u_1 appears

$$\dot{y} = -\frac{k_2(x_3)x_1^2x_2 + c_{\mathrm{in}}k_1(x_3)x_2 + k_1(x_3)x_1^2 - c_{\mathrm{in}}k_1(x_3)x_1}{(x_1 - c_{\mathrm{in}})^2} = L_{\boldsymbol{f}}h(\boldsymbol{x}) \tag{37}$$

$$\ddot{y} = L_{\boldsymbol{f}}^2h(\boldsymbol{x}) + L_{\boldsymbol{g}_1}h(\boldsymbol{x})u_1. \tag{38}$$

This yields relative degree $r = 2$ for the considered subsystem. Thus, the state feedback (9) can be written as

$$u_1(\boldsymbol{x}) = \frac{1}{L_{\boldsymbol{g}_1} L_{\boldsymbol{f}} h(\boldsymbol{x})} \left(v - L_{\boldsymbol{f}}^2 h(\boldsymbol{x}) \right). \tag{39}$$

Since $r < n$, this feedback achieves only a partial linearization. To stabilize a constant output value y_{ref} we use the additional feedback

$$v = p_0(y_{\mathrm{ref}} - y) - p_1 \dot{y}. \tag{40}$$

Hence, the input-output dynamics can be described by a given characteristic polynomial

$$\rho(s) = s^2 + p_1 s + p_0. \tag{41}$$

We obtain the linearizing and stabilizing feedback law by combining (39) and (40) and expressing the time derivatives of y by Lie derivatives

$$u_1(\boldsymbol{x}) = \frac{1}{L_{\boldsymbol{g}_1} L_{\boldsymbol{f}} h(\boldsymbol{x})} \left(p_0(y_{\mathrm{ref}} - h(\boldsymbol{x})) - p_1 L_{\boldsymbol{f}} h(\boldsymbol{x}) - L_{\boldsymbol{f}}^2 h(\boldsymbol{x}) \right). \tag{42}$$

Using this control law the system output y can be kept constant for an desired y_{ref}. However, the main goal is to keep $c_{\mathrm{B}} = x_2$ constant. Therefore a steady-state dependency between y_{ref} and $x_{2,\mathrm{ref}}$ is searched. Knowing this, from an desired $x_{2,\mathrm{ref}}$ the corresponding y_{ref} can be computed. From (36) follows

$$y_{\mathrm{ref}} = \frac{x_{2,\mathrm{ref}}}{c_{\mathrm{in}} - x_{1,\mathrm{ref}}}. \tag{43}$$

To know how $x_{1,\mathrm{ref}}$ depends on $x_{2,\mathrm{ref}}$ we look at (37). For steady state one obtains

$$0 = -\left(k_2(x_3) x_{1,\mathrm{ref}}^2 x_{2,\mathrm{ref}} + c_{\mathrm{in}} k_1(x_3) x_{2,\mathrm{ref}} + k_1(x_3) x_{1,\mathrm{ref}}^2 - c_{\mathrm{in}} k_1(x_3) x_{1,\mathrm{ref}} \right), \tag{44}$$

which is quadratic w.r.t. $x_{1,\mathrm{ref}}$. The physically meaningful solution is given by

$$x_{1,\mathrm{ref}} = \frac{c_{\mathrm{in}} k_1(x_3) - \sqrt{-c_{\mathrm{in}} k_1(x_3)(4k_2(x_3) x_{2,\mathrm{ref}}^2 + 4k_1(x_3) x_{2,\mathrm{ref}} - c_{\mathrm{in}} k_1(x_3))}}{2(k_1(x_3) + k_2(x_3) x_{2,\mathrm{ref}})}. \tag{45}$$

Due to (43) with (45) we obtain

$$y_{\mathrm{ref}} = \frac{c_{\mathrm{in}}(2k_2(x_3) x_{2,\mathrm{ref}}^* + k_1(x_3)) - \sqrt{-c_{\mathrm{in}} k_1(x_3)(4k_2(x_3)(x_{2,\mathrm{ref}}^*)^2 + 4k_1(x_3) x_{2,\mathrm{ref}}^* - c_{\mathrm{in}} k_1(x_3))}}{2c_{\mathrm{in}}(c_{\mathrm{in}} k_2(x_3) + k_1(x_3))}. \tag{46}$$

We introduce the new variable $x_{2,\mathrm{ref}}^*$, such that

$$x_{2,\mathrm{ref}}^* = x_{2,\mathrm{ref}} + k_{\mathrm{i}} \int_0^t \left(x_{2,\mathrm{ref}}(\tau) - x_2(\tau) \right) \mathrm{d}\,\tau, \quad k_i > 0. \tag{47}$$

This modification by adding an I-component is made to ensure steady state accuracy also in case of disturbances or parameter variation. This modification is indispensable if c_{in} is not well known .

Now we consider the second subsystem in order to design a controller for the cooling circuit. With a given temperature ϑ_{K0}, the temperature of the cooling medium we choose the simple P-controller

$$u_2 = -\frac{\beta}{\gamma}(x_3 - \vartheta_{\mathrm{K0}}). \tag{48}$$

Therewith for x_4 the stable dynamics

$$\dot{x}_4 = -\beta(x_4 - \vartheta_{\mathrm{K0}}) \tag{49}$$

occurs.

With (42) and (48) we got controllers for both subsystems, namely for the concentration subsystem and the cooling circuit. Simulation results are shown in Section 4.3.

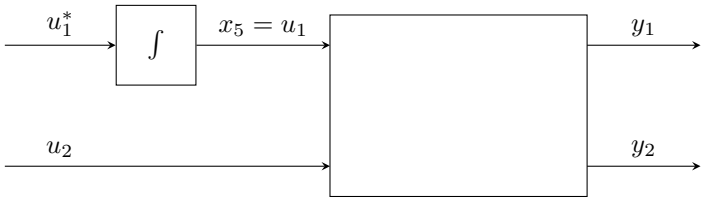


Fig. 2: Structure of CSTR system with dynamic extension

4.2. Controller Design by Full Feedback Linearization

The previous section described a feedback controller just for a partial system of the CSTR. Now we consider whether it is possible to control the whole system by some feedback controller of multi-input-multi-output type shown in Section 2.2. For system output we choose

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} h_1(\boldsymbol{x}) \\ h_2(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} x_2 \\ \hline c_{\mathrm{in}} - x_1 \\ x_3 \end{pmatrix}. \tag{50}$$

Using Definition 2.2 we determine the relative degree (r_1, r_2):

$$\begin{array}{rcll} L_{\boldsymbol{g}_1} h_1(\boldsymbol{x}) & \equiv & 0 & \\ L_{\boldsymbol{g}_2} h_1(\boldsymbol{x}) & \equiv & 0 & \\ L_{\boldsymbol{g}_1} h_2(\boldsymbol{x}) & = & \vartheta_{\mathrm{in}} - x_3 & \Rightarrow r_2 = 1? \\ L_{\boldsymbol{g}_2} h_2(\boldsymbol{x}) & \equiv & 0 & \\ \hline L_{\boldsymbol{g}_1} L_{\boldsymbol{f}} h_1(\boldsymbol{x}) & \not\equiv & 0 & \Rightarrow r_1 = 2? \\ L_{\boldsymbol{g}_2} L_{\boldsymbol{f}} h_1(\boldsymbol{x}) & \equiv & 0 & \end{array} \tag{51}$$

Checking the decoupling matrix

$$A(\boldsymbol{x}^0) = \begin{pmatrix} L_{\boldsymbol{g}_1} L_{\boldsymbol{f}} h_1(\boldsymbol{x}^0) & L_{\boldsymbol{g}_2} L_{\boldsymbol{f}} h_1(\boldsymbol{x}^0) \\ L_{\boldsymbol{g}_1} h_2(\boldsymbol{x}^0) & L_{\boldsymbol{g}_2} h_2(\boldsymbol{x}^0) \end{pmatrix} = \begin{pmatrix} * & 0 \\ * & 0 \end{pmatrix} \tag{52}$$

for regularity we found out the relative degree $r_1 = 2, r_2 = 1$ is ill-defined. As proposed in Section 2.3 we use an additional integrator

$$\dot{u}_1 = u_1^* \tag{53}$$

and treat u_1 as an additional state $x_5 = u_1$. Fig. 2 shows the structure of the augmented system. This leads to the state space representation

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}_1(\boldsymbol{x}) u_1^* + \boldsymbol{g}_2(\boldsymbol{x}) u_2 \tag{54}$$

with

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} r_{\mathrm{A}}(x_1, x_3) + (c_{\mathrm{in}} - x_1) x_5 \\ r_{\mathrm{B}}(x_1, x_2, x_3) - x_2 x_5 \\ h(x_1, x_2, x_3) + \alpha(x_4 - x_3) + (\vartheta_{\mathrm{in}} - x_3) x_5 \\ \beta(x_3 - x_4) \\ 0 \end{pmatrix}, \quad \boldsymbol{g}_1(\boldsymbol{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{g}_2(\boldsymbol{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \gamma \\ 0 \end{pmatrix}. \tag{55}$$

For this augmented 5-dimensional system also relative degree has to be determined. Doing this one recognize that y_1 has to be derived three times and y_2 two times until any input component occurs. This leads to the decoupling matrix

$$A(\boldsymbol{x}^0) = \begin{pmatrix} L_{\boldsymbol{g}_1} L_{\boldsymbol{f}}^2 h_1(\boldsymbol{x}^0) & L_{\boldsymbol{g}_2} L_{\boldsymbol{f}}^2 h_1(\boldsymbol{x}^0) \\ L_{\boldsymbol{g}_1} L_{\boldsymbol{f}} h_2(\boldsymbol{x}^0) & L_{\boldsymbol{g}_2} L_{\boldsymbol{f}} h_2(\boldsymbol{x}^0) \end{pmatrix} = \begin{pmatrix} * & * \\ * & * \end{pmatrix} \tag{56}$$

which is fully occupied and regular. Thus relative degree is $r_1 = 3, r_2 = 2$. Since $r_1 + r_2 = n$ the augmented system is full linearizable. The state feedback (16) adapted to the augmented system can be written as

$$\begin{pmatrix} u_1^* \\ u_2 \end{pmatrix} = A^{-1}(\boldsymbol{x}) \begin{pmatrix} v_1 - L_{\boldsymbol{f}}^3 h_1(\boldsymbol{x}) \\ v_2 - L_{\boldsymbol{f}}^2 h_2(\boldsymbol{x}) \end{pmatrix}. \tag{57}$$

Now we got a three dimensional and a two dimensional linear subsystem which are decoupled from each other. In order to stabilize some constant output vector $\boldsymbol{y}_{\mathrm{ref}} = (y_{1,\mathrm{ref}}, y_{2,\mathrm{ref}})^T$ we use the additional feedback

$$\begin{aligned} v_1 &= p_{1,0}(y_{1,\mathrm{ref}} - y_1) - p_{1,1}\dot{y}_1 - p_{1,2}\ddot{y}_1, \\ v_2 &= p_{2,0}(y_{2,\mathrm{ref}} - y_2) - p_{2,1}\dot{y}_2. \end{aligned} \tag{58}$$

Therewith the input-output dynamics of the two subsystems is described by the characteristic polynomials

$$\begin{aligned} \rho_1(s) &= s^3 + p_{1,2}s^2 + p_{1,1}s + p_{1,0}, \\ \rho_2(s) &= s^2 + p_{2,1}s + p_{2,0}. \end{aligned} \tag{59}$$

Combining (57) and (58) leads to the control law

$$\begin{pmatrix} u_1^* \\ u_2 \end{pmatrix} = \Delta^{-1}(\boldsymbol{x}) \cdot \begin{pmatrix} p_{1,0}(y_{1,\mathrm{ref}} - h_1(\boldsymbol{x})) - p_{1,1}L_{\boldsymbol{f}}h_1(\boldsymbol{x}) - p_{1,2}L_{\boldsymbol{f}}^2h_1(\boldsymbol{x}) - L_{\boldsymbol{f}}^3h_1(\boldsymbol{x}) \\ p_{2,0}(y_{2,\mathrm{ref}} - h_2(\boldsymbol{x})) - p_{2,1}L_{\boldsymbol{f}}h_2(\boldsymbol{x}) - L_{\boldsymbol{f}}^2h_2(\boldsymbol{x}) \end{pmatrix}. \tag{60}$$

To compute $y_{1,\mathrm{ref}}$ from $x_{2,\mathrm{ref}}$ we again use the steady state characteristic (46). Also the additional I-component (47) is used. Furthermore we use an I-component for $y_{2,\mathrm{ref}}$:

$$y_{2,\mathrm{ref}} = x_{3,\mathrm{ref}} + k_{\mathrm{i},2} \int_0^t \big(x_{3,\mathrm{ref}}(\tau) - x_3(\tau)\big)\,\mathrm{d}\,\tau, \quad k_{\mathrm{i},2} > 0, \tag{61}$$

where $x_{3,\mathrm{ref}}$ is the desired reactor temperature.

4.3. Simulation Results

Both control laws from Section 4.1 and 4.2 have been implemented using MATLAB. For the control laws, (42) and (48) respectively (57) is being evaluated. The required Lie derivatives are computed in each step by calling MATLAB-MEX functions wrapping calls to the ADOL-C using package LIEDRIVERS [10]. Take for example, the computation of Lie derivatives of the scalar field h along the vector field $\boldsymbol{f}$, required in (42) and (57), respectively. Therefore the C function `Lie_scalarc` has to be called. With

$$L_{\boldsymbol{g}}L_{\boldsymbol{f}}^{r-1}h(\boldsymbol{x}) = L_{\boldsymbol{f}+\boldsymbol{g}}^{r}h(\boldsymbol{x}) - L_{\boldsymbol{f}}^{r}h(\boldsymbol{x}), \tag{62}$$

also mixed Lie derivatives may be computed this way, see last line of Eq. (7).

The presented control laws have been parametrized by

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} 10\,000 \\ 200 \end{pmatrix}, \quad k_{\mathrm{i}} = 20 \tag{63}$$

for the controller based on partial feedback linearization and

$$\begin{pmatrix} p_{1,0} \\ p_{1,1} \\ p_{1,2} \end{pmatrix} = \begin{pmatrix} 8\,000\,000 \\ 120\,000 \\ 600 \end{pmatrix}, \quad \begin{pmatrix} p_{2,0} \\ p_{2,1} \end{pmatrix} = \begin{pmatrix} 40\,000 \\ 400 \end{pmatrix}, \quad k_{\mathrm{i},1} = k_{\mathrm{i},2} = 20 \tag{64}$$

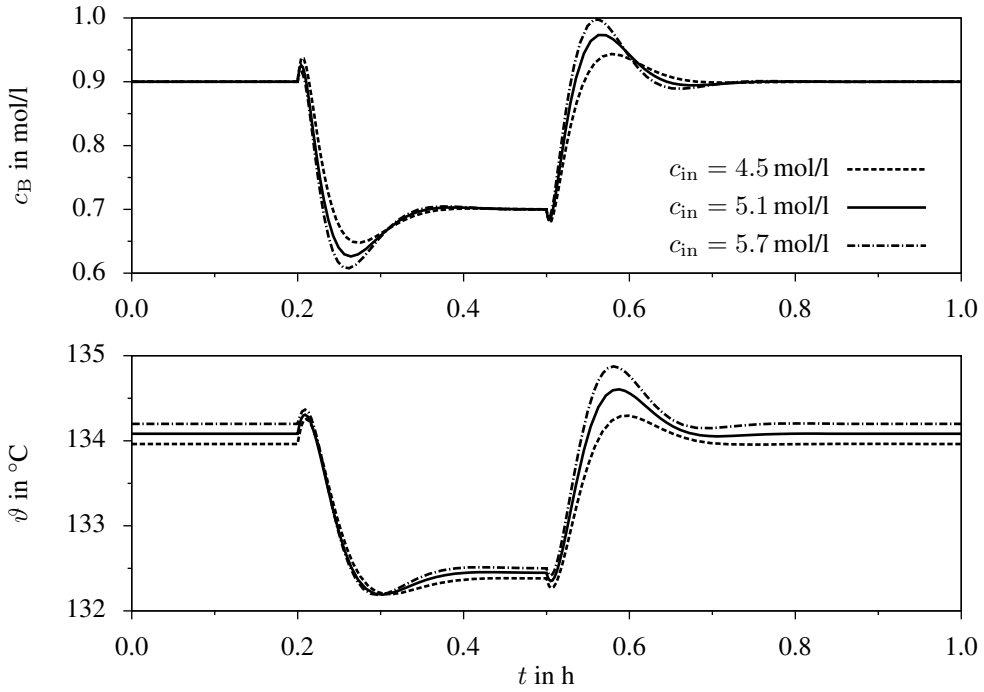
for the controller based on full feedback linearization. The control variables are said to be limited by

$$5 \le u_1 \cdot \mathrm{h} \le 35, \quad -8500 \le u_2 \cdot \mathrm{kJ}^{-1}\,\mathrm{h} \le 0. \tag{65}$$

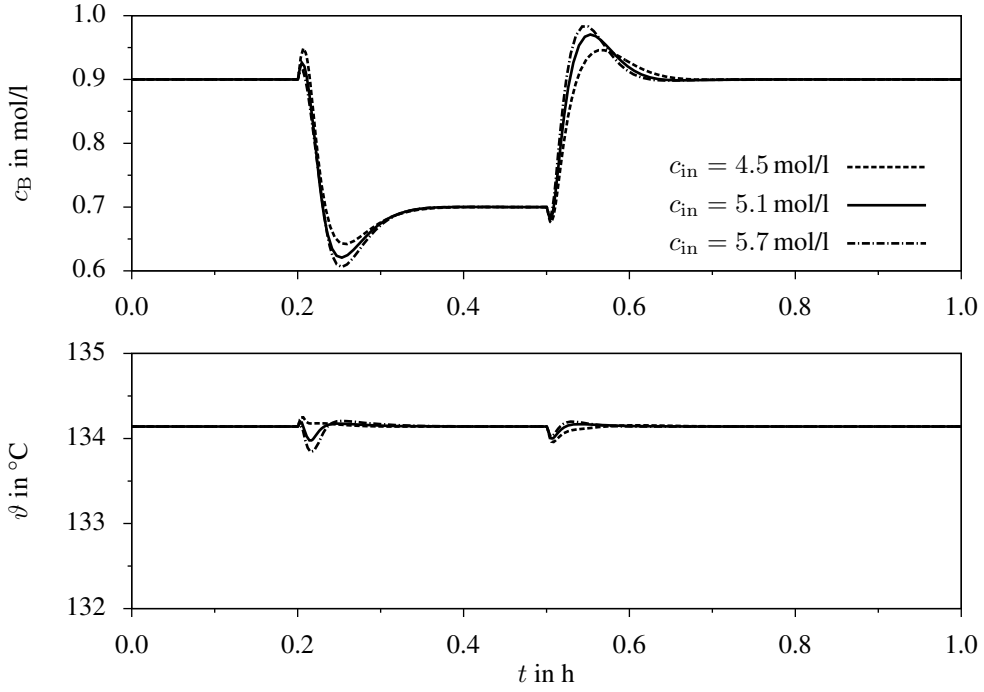
For some set point jumps simulation results are given in Fig. 3. There c_{in} is assumed to be $5.1\,\mathrm{mol\,l}^{-1}$ while it actually varies in some particular range. Therefore simulation was executed for three different values.

5. SUMMARY

During the last decades, several nonlinear control methods based on differential geometric concepts have been developed. The associated control laws are often formulated in terms of Lie derivatives. Usually, these Lie derivatives are computed symbolically, which can be difficult for highly complex systems. Alternatively, these Lie derivatives can be calculated using algorithmic differentiation. We reported the implementation of an library for the computation of Lie derivatives used in nonlinear control. This library was applied successfully to the complicated nonlinear model of a tank reactor.



(a) Partial feedback control



(b) Full feedback control

Fig. 3: Simulation results for the CSTR system with $x_{3,\mathrm{ref}} = 134.14\,°\mathrm{C}$, $\alpha = 30.828\,\mathrm{h}^{-1}$, $\beta = 86.688\,\mathrm{h}^{-1}$, $\gamma = 0.1\,\mathrm{K\,kJ}^{-1}$, $\delta = 3.522 \cdot 10^{-4}\,\mathrm{m}^3\,\mathrm{K\,kJ}^{-1}$, $c_\mathrm{in} = 5.1\,\mathrm{mol\,l}^{-1}$, $\vartheta_\mathrm{in} = 130\,°\mathrm{C}$, $\vartheta_\mathrm{K0} = 128.95\,°\mathrm{C}$, $k_{10} = 1.287 \cdot 10^{12}\,\mathrm{h}^{-1}$, $k_{20} = 9.043 \cdot 10^{6}\,\mathrm{m}^3\,(\mathrm{mol\,h})^{-1}$, $E_1 = 9758.3$, $E_2 = 8560$, $\Delta H_\mathrm{AB} = 4.2\,\mathrm{kJ\,mol}^{-1}$, $\Delta H_\mathrm{AD} = -41.85\,\mathrm{kJ\,mol}^{-1}$, $\Delta H_\mathrm{BC} = -11.0\,\mathrm{kJ\,mol}^{-1}$.

6. REFERENCES

[1] A. Isidori, *Nonlinear Control Systems: An Introduction*, Springer-Verlag, London, 3rd edition, 1995.

[2] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control systems*, Springer, New York, 1990.

[3] M. Bär, H. Fritz, and M. Zeitz, "Rechnergestützter Entwurf nichtlinearer Beobachter mit Hilfe einer symbolverarbeitenden Programmiersprache," *Automatisierungstechnik*, vol. 35, no. 5, pp. 177–183, 1987.

[4] J. C. Gómez, "NONACODE: A software package for analysis and design of adaptive control of nonlinear systems using computer algebra," in *SIAM 1993 Annual Meeting, Philadelphia, USA*, July 1993.

[5] B. de Jager, "The use of symbolic computation in nonlinear control: Is it viable?," *IEEE Trans. on Automatic Control*, vol. 40, no. 1, pp. 84–89, 1995.

[6] A. Kugi, K. Schlacher, and R. Novaki, *Symbolic Computation for the Analysis and Synthesis of Nonlinear Control Systems*, vol. 2 of *Software Studies*, pp. 255–264, WIT-Press, Southampton, 1999.

[7] K. Röbenack and K. J. Reinschke, "Reglerentwurf mit Hilfe des Automatischen Differenzierens," *Automatisierungstechnik*, vol. 48, no. 2, pp. 60–66, 2000.

[8] K. Röbenack, "Automatic differentiation and nonlinear controller design by exact linearization," *Future Generation Computer Systems*, vol. 21, no. 8, pp. 1372–1379, 2005.

[9] K. Röbenack, "Controller design for nonlinear multi-input – multi-output systems based on an algorithmic plant description," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 13, no. 2, pp. 193–209, 2007.

[10] K. Röbenack, J. Winkler, and S. Wang, "LIEDRIVERS — a toolbox for the efficient computation of Lie derivatives based on the object-oriented algorithmic differentiation package ADOL-C," in *Proc. of the 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Zurich, 2011, pp. 57–66.

[11] K.-U. Klatt and S. Engell, "Rührkesselreaktor mit Parallel- und Folgereaktion," in *Nichtlineare Regelungen: Methoden, Werkzeuge, Anwendungen*, S. Engell, Ed., vol. 1026 of *VDI-Berichte*, pp. 101–108. VDI-Verlag, Düsseldorf, 1993.

[12] V. Polyakov, R. Ghanadan, and G. L. Blankenship, "Symbolic numerical computational tools for nonlinear and adaptive control," in *Proc. IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, Tucson, Arizona, 1994, pp. 117–122.

[13] H. G. Kwatny and G. L. Blankenship, *Nonlinear Control and Analytical Mechanics: A Computational Approach*, Birkhäuser, Boston, 2000.

[14] R. Sepulchre, M. Janković, and P. Kokotović, *Constructive Nonlinear Control*, Springer, London, 1997.

[15] A. J. van der Schaft, "Complimentary modeling of hybrid systems," *IEEE Trans. on Automatic Control*, vol. 43, no. 4, pp. 483–490, Dec. 1998.

[16] J. Descusse and C. H. Moog, "Decoupling with dynamic compensation for strong invertible affine non-linear systems," *Int. J. Control*, vol. 42, no. 6, pp. 1387–1398, 1985.

[17] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2nd edition, 2008.

[18] M. Berz, Ch. Bischof, G. Corliss, and A. Griewank, Eds., *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, 1996.

[19] G. Corliss, Ch. Faure, A. Griewank, L. Hascoët, and U. Naumann, Eds., *Automatic Differentiation: From Simulation to Optimization*, Springer-Verlag, New York, 2002.

[20] C. Bendtsen and O. Stauning, "TADIFF, a flexible C++ package for automatic differentiation," Technical Report IMM-REP-1997-07, TU of Denmark, Dept. of Mathematical Modelling, Lungby, 1997.

[21] Y. F. Chang, "Automatic solution of differential equations," in *Constructive and Computational Methods for Differential and Integral Equations*, D. L. Colton and R. P. Gilbert, Eds., vol. 430 of *Lecture Notes in Mathematics*, pp. 61–94. Springer Verlag, New York, 1974.

[22] G. F. Corliss and Y. F. Chang, "Solving ordinary differential equations using Taylor series," *ACM Trans. Math. Software*, vol. 8, pp. 114–144, 1982.

[23] A. Griewank, G.F. Corliss, P. Henneberger, G. Kirlinger, F.A. Potra, and H.J. Stetter, "High-order stiff ODE solvers via automatic differentiation and rational prediction," in *Numerical Analysis and Its Applications*, vol. 1196 of *Lecture Notes in Computer Science*, pp. 114–125. Springer, 1997.

[24] K. Röbenack and K. J. Reinschke, "The computation of Lie derivatives and Lie brackets based on automatic differentiation," *Z. Angew. Math. Mech.*, vol. 84, no. 2, pp. 114–123, 2004.

[25] K. Röbenack, "Computation of Lie derivatives of tensor fields required for nonlinear controller and observer design employing automatic differentiation," *Proc. in Applied Mathematics and Mechanics*, vol. 5, no. 1, pp. 181–184, 2005.

[26] K. Röbenack, *Regler- und Beobachterentwurf für nichtlineare Systeme mit Hilfe des Automatischen Differenzierens*, Shaker Verlag, Aachen, 2005.

[27] K. Röbenack, "Computation of multiple Lie derivatives by algorithmic differentiation," *J. of Computational and Applied Mathematics*, vol. 213, no. 2, pp. 454–464, 2008.

[28] K. Röbenack, "Computation of mixed Lie derivatives in nonlinear control," *Proc. in Applied Mathematics and Mechanics*, vol. 10, no. 1, pp. 627–628, Dec. 2010.

[29] A. Griewank, D. Juedes, and J. Utke, "ADOL-C: A package for automatic differentiation of algorithms written in C/C++," *ACM Trans. Math. Software*, vol. 22, pp. 131–167, 1996.

[30] A. Walther, A. Griewank, and O. Vogel, "ADOL-C: Automatic differentiation using operator overloading in C++," *Proc. in Applied Mathematics and Mechanics*, vol. 2, no. 1, pp. 41–44, 2003.

[31] H. Chen, A. Kremling, and F. Allgöwer, "Nonlinear predictive control of a CSTR benchmark problem," in *Proc. 3rd European Control Conf. ECC'95, Roma, Italy*, 1995, pp. 3247–3252.

[32] K. Schlacher and A. Kugi, "Zustandsregler mit Beobachter für einen Rührkesselreaktor," in *Nichtlineare Regelungen: Methoden, Werkzeuge, Anwendungen*, Düsseldorf, 1993, VDI-Berichte 1026, pp. 251–265, VDI-Verlag.