**Preprints of the 18th IFAC World Congress**
**Milano (Italy) August 28 - September 2, 2011**

# Automatic Derivation of Optimal Piecewise Affine Approximations of Nonlinear Systems

**Michal Kvasnica** [*,1], **Alexander Szücs** [*], and **Miroslav Fikar** [*]

[*] *Institute of Automation, Information Engineering and Mathematics, Slovak University of Technology, 812 37 Bratislava, Slovakia*

**Abstract:** : The paper proposes a method for deriving optimal piecewise affine (PWA) approximations of nonlinear systems with known analytic form. The procedure employs nonlinear optimization to derive an approximation of maximal accuracy and given complexity. We show that under mild assumptions, the task can be transformed into a series of one-dimensional approximations. An automatic procedure for generation of such approximations is discussed as well.

*Keywords:* hybrid systems, approximation, nonlinear optimization

## 1. INTRODUCTION

Mathematical models of physical plants play a vital role in many areas, such as in rigorous simulations, analysis, or control synthesis. Typically, high model accuracy is usually desired while keeping the model complexity on an acceptable level. Traditionally, nonlinear models were preferred from simulations, while most of available control techniques are based on a local approximation around a single operating point. PWA systems (Sontag, 1981) can be viewed as a compromise solution between accuracy of the model and its complexity. They are composed of several local models accompanied with logic IF-THEN conditions which enforce switching of the local dynamics. Therefore they belong to the class of hybrid systems.

The problem which we address in this paper is the following: given a nonlinear dynamical model $x^+ = f(x,u)$ and a fixed complexity of its PWA approximation $\widetilde{f}(x,u) \approx f(x,u)$, how should one design $\widetilde{f}$ which minimizes the approximation error $\int (f(x,u) - \widetilde{f}(x,u))^2$? The answer is non-trivial even when putting optimality of the approximation aside. Traditionally, two distinct approaches for deriving PWA approximations are used. When the mathematical formulation of the original nonlinear system is known, one can design the approximation by hand. This is usually done by employing human knowledge and experience to devise several linearization points around which the original nonlinear model should be linearized. Needless to say, placement of such points has a crucial impact on the accuracy of the approximation. The HYSDEL (Hybrid Systems Description Language) tool (Torrisi and Bemporad, 2004; Kvasnica and Herceg, 2010) can be used to accelerate this line of development. Formally, HYSDEL transforms a linguistic description of a PWA system (or of a hybrid system in general) into the corresponding mathematical form. The language allows to define IF-THEN switching rules which, based on whether some logic condition is satisfied or not, enforce certain continuous dynamics. Another option is to use hybrid identification techniques (Ferrari-Trecate et al., 2001; Roll et al., 2004; Ferrari-Trecate, 2005) to construct the PWA approximation from the input-output measurements. The crucial advantage is that the model of the original nonlinear system is not required to be fully available. The downside, however, is that the approximation is only accurate in the interval captured by the identification data. Moreover, the procedure is computationally expensive and suited mainly to low-dimensional problems.

In this work we propose to use an optimization-based approach to derive PWA approximations of nonlinear systems whose vector field is an a-priori known function of multiple variables. After formally stating the problem in Section 2, we show in Section 3 that an optimal PWA approximation of generic nonlinear functions in one variable can be formulated and solved as a nonlinear programming problem. Several non-trivial illustrative cases are discussed to show that the approach is both efficient and computational tractable. Subsequently, the approach is extended to deriving PWA approximations of multivariable functions in Section 4. We show that, under a certain assumption, the problem boils down to solving a series of one-dimensional approximations. The algorithmic and software implementation of the approximation procedure are then discussed in Section 5. Specifically, we introduce a new software tool which is capable of exporting the obtained optimal PWA approximations into the HYSDEL language. This brings two crucial advantages. First, the HYSDEL compiler can be used to convert the PWA approximation into a mathematical form, which is then suitable e.g. for control design. Second, since the exported approximation is described in a human-readable format, it can be further fine-tuned by hand. Finally, in Section 6 we illustrate the procedure on a case study involving a highly non-linear chemical reactor.

[1] Corresponding author, e-mail: michal.kvasnica@stuba.sk

## 2. PROBLEM STATEMENT

We consider generic dynamic systems in discrete-time

$$x^+ = f(x,u), \tag{1}$$

where the vector field $f(\cdot,\cdot)$ is assumed to be continuous in the state variables $x \in \mathbb{R}^{n_x}$ and in the inputs $u \in \mathbb{R}^{n_u}$. System states and inputs are assumed to be constrained to connected and closed domains $\mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$, respectively.

The objective is to approximate (1) by a different dynamic system $x^+ = \widetilde{f}(x,u)$ whose vector field $\widetilde{f}(x,u)$ is a PWA function which consists of a pre-specified number $N$ of local linear dynamics:

$$\widetilde{f}(x,u) = \begin{cases} A_1x + B_1u + c_1 & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_1 \\ \quad\vdots & \quad\vdots \\ A_Nx + B_Nu + c_N & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_N. \end{cases} \tag{2}$$

Here, $A_i \in \mathbb{R}^{n_x \times n_x}$, $B_i \in \mathbb{R}^{n_x \times n_u}$, $c_i \in \mathbb{R}^{n_x}$, are the state-update matrices of the $i$-th local linear approximation, and $\mathcal{R}_i \subset \mathbb{R}^{n_x \times n_u}$ is the region of validity of the $i$-th local model satisfying $\mathcal{R}_i \neq \emptyset$, $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i \neq j$, and $\cup_i \mathcal{R}_i = \mathcal{X} \times \mathcal{U}$.

Formally, the problem which we aim at solving can be stated as follows:

*Problem 2.1.* Given a nonlinear vector field $f(x,u)$ of system (1), find the PWA approximation (2) of pre-specified complexity which minimizes the approximation error

$$e_{\text{aprx}} := \int (f(x,u) - \widetilde{f}(x,u))^2 \, dxdu, \tag{3}$$

where the integral is evaluated over the whole region of validity of (1), i.e. over $\mathcal{X} \times \mathcal{U}$.

In the sequel we show how to solve Problem 2.1 provided that the vector field $f(z)$, $z = [x,\ u]^T$ satisfies the following assumption.

*Assumption 2.2.* The function $f(z_1, \ldots, z_n)$ can be written as $\sum_{i=1}^{n} \alpha_i \Big( \prod_{j=p_i}^{q_i} f_j(z_j) \Big)$.

As an example, the function $z_1 e^{z_2}$ satisfies such an assumption, while the function $e^{z_1 z_2}$ does not. Although the assumption is somewhat restrictive, the gained advantage is that approximating any multivariable function $f(z_1, \ldots, z_n)$ boils down to solving a series of 1D problems, as evidenced in the following two sections.

*Remark 2.3.* Since the approximation procedure discussed in the sequel considers only the vector field in the right-hand-side of (1), continuous-time systems $\dot{x} = f(x,u)$ can be treated as well.

## 3. FUNCTIONS IN ONE VARIABLE

First, we consider the one-dimensional case, i.e. approximating a nonlinear function $f(z) : \mathbb{R} \to \mathbb{R}$, with domain $\mathcal{Z} \subset \mathbb{R}$, by a PWA function $\widetilde{f}(z) = a_i z + c_i$ if $z \in \mathcal{R}_i$. Since $\mathcal{Z}$ is assumed to be connected and closed, it is a line segment $[\underline{z},\ \overline{z}]$. Regions $\mathcal{R}_i$ define the partition of such a line into $N$ non-overlapping parts, i.e. $\mathcal{R}_1 = [\underline{z},\ r_1]$, $\mathcal{R}_2 = [r_1,\ r_2]$, …, $\mathcal{R}_{N-1} = [r_{N-2},\ r_{N-1}]$, $\mathcal{R}_N = [r_{N-1},\ \overline{z}]$ with $\cup_i \mathcal{R}_i = [\underline{z},\ \overline{z}]$. Solving Problem 2.1 then becomes to find the slopes $a_i$, offsets $c_i$ and breakpoints $r_i$ such that the approximation error is minimized, i.e.

$$\min_{a_i,c_i,r_i} \int_{\underline{z}}^{\overline{z}} (f(z) - \widetilde{f}(z))^2 \, dz \tag{4a}$$

$$\text{s.t. } \widetilde{f}(z) = \begin{cases} a_1 z + c_1 & \text{if } z \in [\underline{z},\ r_1] \\ \quad\vdots & \quad\vdots \\ a_N z + c_N & \text{if } z \in [r_{N-1},\ \overline{z}] \end{cases} \tag{4b}$$

$$\underline{z} \leq r_1 \leq \cdots \leq r_{N-1} \leq \overline{z}, \tag{4c}$$

$$a_i r_i + c_i = a_{i+1} r_i + c_{i+1},\ i = 1, \ldots, N-1, \tag{4d}$$

where (4d) enforces continuity of $\widetilde{f}(z)$ along the breakpoints $r_i$. The IF-THEN based nonlinear constraint (4b) can be eliminated by observing that, by definition, regions $\mathcal{R}_i$ are non-overlapping, and the integral in (4a) can hence be written as

$$\int_{\underline{z}}^{\overline{z}} \big(f(z) - \widetilde{f}(z)\big)^2 = \sum_{i=1}^{N} \Big( \int_{r_{i-1}}^{r_i} \big(f(z) - (a_i z + c_i)\big)^2 \Big), \tag{5}$$

with $r_0 = \underline{z}$ and $r_N = \overline{z}$. The NLP (4) can therefore be written as

$$\min_{a_i,c_i,r_i} \sum_{i=1}^{N} \Big( \int_{r_{i-1}}^{r_i} \big(f(z) - (a_i z + c_i)\big)^2 \, dz \Big) \tag{6a}$$

$$\text{s.t. } \underline{z} \leq r_1 \leq \cdots \leq r_{N-1} \leq \overline{z}, \tag{6b}$$

$$a_i r_i + c_i = a_{i+1} r_i + c_{i+1},\ i = 1, \ldots, N-1. \tag{6c}$$

For simple functions $f(z)$, the integral in (6a) can be expressed in an analytical form in unknowns $a_i, c_i, r_i$, along with the corresponding gradients. For more complex expressions, the integrals can be evaluated numerically, e.g. by using the trapezoidal rule. In either case, problem (6) can be solved to a local optimality e.g. by using the `fmincon` solver of MATLAB. Alternatively, one can use global optimization methods (Adjiman et al., 1996; Papamichail and Adjiman, 2004; Chachuat et al., 2006) which guarantee that an $\epsilon$-neighborhood of the global optimum can be found.

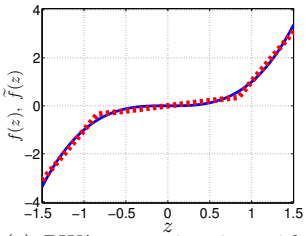
*Example 3.1.* Consider the function $f(z) = z^3$ on domain $-1.5 \leq z \leq 1.5$. The analytic form of the integral (6a) is

$$\sum_{i=1}^{N} \big( c_i^2 (r_i + r_{i-1}) + a_i c_i (r_i^2 - r_i^2) + \frac{a_i^2}{3}(r_i^3 - r_{i-1}^3) - \\ - \frac{c_i}{2}(r_i^4 - r_{i-1}^4) - \frac{2a_i}{5}(r_i^5 - r_{i-1}^5) + \frac{1}{7}(r_i^7 - r_{i-1}^7) \big),$$

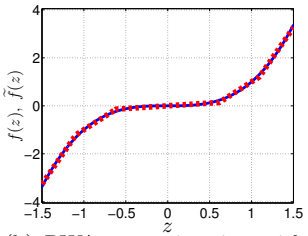
with $r_0 = -1.5$ and $r_N = 1.5$. The PWA approximation of $f(z)$ with $N = 3$ regions was obtained by solving the NLP (6) using `fmincon`, which only took 0.05 seconds on a 2.4 GHz CPU running MATLAB 2009b. The obtained PWA approximation is then given by

$$\widetilde{f}(z) = \begin{cases} 4.1797z + 3.1621 & \text{if } -1.5 \leq z \leq -0.8423 \\ 0.4257z & \text{if } -0.8423 \leq z \leq 0.8423 \\ 4.1797z - 3.1621 & \text{if } \ 0.8423 \leq z \leq 1.5 \end{cases}$$

Naturally, quality of the approximation can be improved by increasing the complexity of the PWA function, i.e. by enlarging $N$. Two PWA approximations with $N = 3$ and $N = 5$ are shown, respectively, in Figures 1(a) and 1(b).

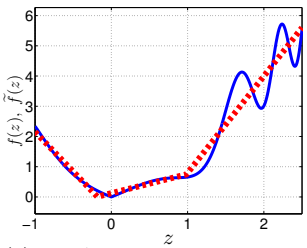



(a) PWA approximation with 3 regions.

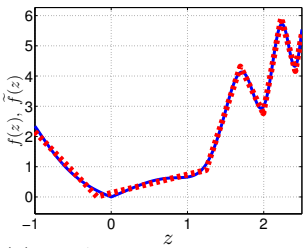



(b) PWA approximation with 5 regions.

Fig. 1. Graph of $f(z) = z^3$ (blue line) and the PWA approximations $\widetilde{f}(z)$ (red dashed lines).




(a) PWA approximation with $N = 3$ regions.




(b) PWA approximation with $N = 7$ regions.

Fig. 2. Graph of $f(z) = |z| + 0.5z^2 - \sin(z^3)$ (blue line) and the PWA approximations $\widetilde{f}(z)$ (red dashed lines).

*Example 3.2.* Consider the function $f(z) = |z| + 0.5z^2 - \sin(z^3)$ on domain $-1 \le z \le 2.5$, graph of which is shown in Figure 2(a). Since no analytic expression of the integral in (6a) could be obtained, we have opted for numeric integration of the cost while solving the NLP problem (6) by `fmincon`. The PWA approximations for $N = 3$ and $N = 7$ are shown in Figures 2(a) and 2(b).

## 4. MULTIVARIABLE FUNCTIONS

The task is to approximate a given multivariable function $f(z_1, \dots, z_n) : \mathbb{R}^n \to \mathbb{R}$ with domain $\mathcal{Z} \subset \mathbb{R}^n$ by a PWA function $\widetilde{f}(z_1, \dots, z_n)$, defined over the same domain, such that the approximation error (3) is minimized.

*Definition 4.1.* (Williams (1993)). Function $f(z_1, \dots, z_n)$ is called *separable* if it can be expressed as the sum of functions of a single variable, i.e. $f(z_1, \dots, z_n) = f_1(z_1) + \dots + f_n(z_n)$.

If $f(z_1, \dots, z_n)$ is readily separable (e.g. when $f(z_1, z_2) = e^{z_1} + \sin(z_2)$), its optimal PWA approximation can be obtained by applying the 1D scenario of Section 3 to the individual components of the function, i.e. $\widetilde{f}(z_1, \dots, z_n) = \widetilde{f}_1(z_1) + \dots + \widetilde{f}_n(z_n)$. The total number of regions over which the PWA approximation $\widetilde{f}(\cdot)$ is defined is hence given by $\sum_{j=1}^{n} N_j$, where $N_j$ is the pre-specified complexity of the $j$-th approximation $\widetilde{f}_j(z_j)$.

A surprisingly large number of non-separable functions can be converted into the separable form by applying a simple trick, elaborated in more details e.g. in Williams (1993). To introduce the procedure, consider a non-separable function $f(z_1, z_2) = z_1 z_2$ with domain $\mathcal{Z} := [\underline{z}_1,\ \overline{z}_1] \times [\underline{z}_2,\ \overline{z}_2]$. Define two new variables

$$y_1 = (z_1 + z_2), \quad y_2 = (z_1 - z_2). \tag{7}$$

Then it is easy to verify that $1/4(y_1^2 - y_2^2) = z_1 z_2$. The coordinate transformation therefore transforms the original function into a separable form, where both terms ($y_1^2$ and $y_2^2$) are now functions of a single variable. The procedure of Section 3 can thus be applied to compute PWA approximations of $f_{y_1}(y_1) := y_1^2$ and $f_{y_2}(y_2) := y_2^2$, where the function arguments relate to $z_1$ and $z_2$ via (7). Important to notice is that $f_{y_1}(\cdot)$ and $f_{y_2}(\cdot)$ have different domains, therefore their PWA approximations $\widetilde{f}_{y_1}(y_1) \approx y_1^2$ and $\widetilde{f}_{y_2}(y_2) \approx y_2^2$ will, in general, be different. Specifically, the domain of $f_{y_1}(\cdot)$ is $[\underline{y}_1,\ \overline{y}_1]$ with $\underline{y}_1 = \min\{z_1 + z_2 \mid \underline{z}_1 \le z_1 \le \overline{z}_1,\ \underline{z}_2 \le z_2 \le \overline{z}_2\}$ and $\overline{y}_1 = \max\{z_1 + z_2 \mid \underline{z}_1 \le z_1 \le \overline{z}_1,\ \underline{z}_2 \le z_2 \le \overline{z}_2\}$. Similarly, the domain of $f_{y_2}(\cdot)$ is $[\underline{y}_2,\ \overline{y}_2]$, whose boundaries can be computed by respectively minimizing and maximizing $z_1 - z_2$ subject to the constraint $[z_1,\ z_2]^T \in \mathcal{Z}$. The overall PWA approximation $\widetilde{f}(z_1, z_2) \approx z_1 z_2$ then becomes

$$\widetilde{f}(z_1, z_2) = 1/4(\widetilde{f}_{y_1}(z_1 + z_2) - \widetilde{f}_{y_2}(z_1 - z_2)). \tag{8}$$

The value of $\widetilde{f}(z_1, z_2)$ for any points $z_1, z_2$ is obtained by subtracting the value of the PWA function $\widetilde{f}_{y_2}(\cdot)$ evaluated at the point $z_1 - z_2$ from the function value of $\widetilde{f}_{y_1}(\cdot)$ evaluated at $z_1 + z_2$, followed by a linear scaling.

The procedure naturally extends to multivariable functions represented by the product of two nonlinear functions of a single variable, i.e. $f(z_1, z_2) = f_1(z_1) f_2(z_2)$. Here, the transformation (7) becomes

$$y_1 = f_1(z_1) + f_2(z_2), \quad y_2 = f_1(z_1) - f_2(z_2). \tag{9}$$

Therefore, $1/4(y_1^2 - y_2^2) = f(z_1, z_2)$ still holds. Let $f_{y_1}(y_1) := y_1^2$ and $f_{y_2}(y_2) := y_2^2$. The domain of $f_{y_1}(\cdot)$ is $[\underline{y}_1,\ \overline{y}_1]$ and dom $f_{y_2}(\cdot) = [\underline{y}_2,\ \overline{y}_2]$ with

$$\underline{y}_1 = \min\{f_1(z_1) + f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{10a}$$
$$\overline{y}_1 = \max\{f_1(z_1) + f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{10b}$$
$$\underline{y}_2 = \min\{f_1(z_1) - f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{10c}$$
$$\overline{y}_2 = \max\{f_1(z_1) - f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{10d}$$

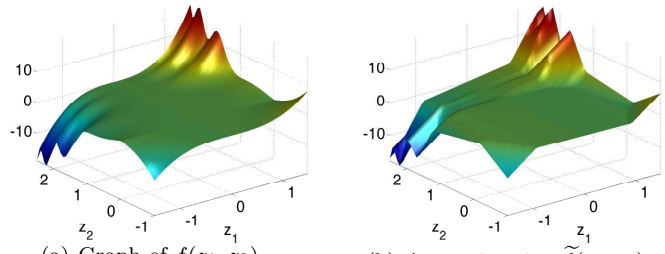
which can be computed by solving four NLP problems.

Finally, since all expressions are now functions of a single variable, the PWA approximations $\widetilde{f}_1(z_1) \approx f_1(z_1)$, $\widetilde{f}_2(z_2) \approx f_2(z_2)$, $\widetilde{f}_{y_1}(y_1) \approx f_{y_1}(y_1)$, and $\widetilde{f}_{y_2}(y_2) \approx f_{y_2}(y_2)$ can be computed by solving the NLP (6). The overall optimal PWA approximation $\widetilde{f}(z_1, z_2) \approx f(z_1, z_2)$ then becomes

$$\widetilde{f}(z_1, z_2) = 1/4\Big(\widetilde{f}_{y_1}\big(\widetilde{f}_1(z_1) + \widetilde{f}_2(z_2)\big) - \widetilde{f}_{y_2}\big(\widetilde{f}_1(z_1) - \widetilde{f}_2(z_2)\big)\Big). \tag{11}$$

The evaluation procedure is similar as above. I.e., given the arguments $z_1$ and $z_2$, one first evaluates $\widetilde{z}_1 = \widetilde{f}_1(z_1)$ and $\widetilde{z}_2 = \widetilde{f}_2(z_2)$. Subsequently, one evaluates $\widetilde{y}_1 = \widetilde{f}_{y_1}(\cdot)$ with the argument $\widetilde{z}_1 + \widetilde{z}_2$, then $\widetilde{y}_2 = \widetilde{f}_{y_2}(\cdot)$ at the point $\widetilde{z}_1 - \widetilde{z}_2$. Finally, $\widetilde{f}(z_1, z_2) = 1/4(\widetilde{y}_1 - \widetilde{y}_2)$.

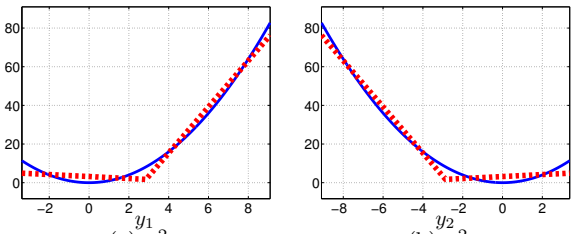
*Example 4.2.* Consider a non-separable function given as the product of the two functions discussed in Examples 3.1 and 3.2, i.e. $f(z_1, z_2) = f_1(z_1) f_2(z_2)$ with $f_1(z_1) = z_1^3$, $f_2(z_2) = |z_2| + 0.5z_2^2 - \sin(z_2)^3$ on domain $[-1.5,\ 1.5] \times$



(a) Graph of $f(z_1, z_2)$. (b) Approximation $\widetilde{f}(z_1, z_2)$

Fig. 3. Graph of $f(z_1, z_2)$ and its PWA approximation (11) in Example 4.2.

$[-1,\ 2.5]$. Graph of the function is shown in Figure 3(a). In order to convert $f(z_1, z_2)$ into a separable form, we introduce variables $y_1$ and $y_2$ as per (9). The PWA approximation $\widetilde{f}(z_1, z_2) \approx f(z_1, z_2)$ is then given by (11). Here, $\widetilde{f}_1(z_1)$ was obtained by approximating $f_1(z_1)$ by a PWA function with 3 regions as shown in Figure 1(a), while $\widetilde{f}_2(z_2) \approx f_2(z_2)$ was approximated by 7 regions as depicted in Figure 2(b). Subsequently, the domains $[\underline{y}_1,\ \overline{y}_1]$ and $[\underline{y}_2,\ \overline{y}_2]$ were computed via (10), which resulted into $\operatorname{dom} y_1 = [-3.374,\ 9.095]$ and $\operatorname{dom} y_2 = [-9.095,\ 3.374]$. Finally, the PWA approximations $\widetilde{f}_{y_1}(y_1) \approx y_1^2$ and $\widetilde{f}_{y_2}(y_2) \approx y_2^2$ were obtained by solving the NLP (6) with $N = 2$. Graphs of $y_1^2$, $y_2^2$ and their respective PWA approximations are presented in Figure 4. The overall approximation $\widetilde{f}(z_1, z_2)$ therefore consists of 14 regions. Despite a rather crude approximation of the square functions, the combined PWA function (11), shown in Figure 3(b), features only a minor average approximation error of 3% and a worst-case error of 15%. By increasing the number of linearizations for $y_1^2$ and $y_2^2$ from $N = 2$ to $N = 4$ (hence increasing the complexity of $\widetilde{f}(z_1, z_2)$ from 14 to 18 regions), the average and worst-case errors can be further reduced to 1% and 8%, respectively.



(a) $y_1^2$ (b) $y_2^2$

Fig. 4. Functions $y_i^2$ (blue) and their PWA approximation $\widetilde{f}_{y_i}(y_i)$ (red dashed lines) in Example 4.2.

Separation of multivariable functions with more than two terms can be performed in an inductive manner. Consider $f(z_1, z_2, z_3) = f_1(z_1)f_2(z_2)f_3(z_3)$. First, approximate the product $f_1(z_1)f_2(z_2)$ by a PWA function of the form of (11), which requires four PWA approximations

$$\widetilde{f}_1(\cdot) \approx f_1(\cdot),\ \widetilde{f}_2(\cdot) \approx f_2(\cdot),\ \widetilde{f}_{y_1}(\cdot) \approx y_1^2,\ \widetilde{f}_{y_2}(\cdot) \approx y_2^2,$$

with $y_1$ and $y_2$ as in (9). Let $f_a(z_1, z_2) := f_1(z_1)f_2(z_2)$. Then $f(z_1, z_2, z_3) = f_a(z_1, z_2)f_3(z_3)$, which can again be approximated as a product of two functions. Specifically, define

$$y_3 = f_a(\cdot) + f_3(z_3), \quad y_4 = f_a(\cdot) - f_3(z_3), \tag{12}$$

and hence $f_a(z_1, z_2)f_3(z_3) = {}^1\!/_4(y_3^2 - y_4^2)$. The domains over which $y_3^2$ and $y_4^2$ need to be approximated are, respectively, $[\underline{y}_3,\ \overline{y}_3]$ and $[\underline{y}_4,\ \overline{y}_4]$ with

$$\underline{y}_3 = \min\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{13a}$$
$$\overline{y}_3 = \max\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{13b}$$
$$\underline{y}_4 = \min\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{13c}$$
$$\overline{y}_4 = \max\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{13d}$$

and $z = [z_1,\ z_2,\ z_3]^T$. Subsequently, three additional PWA approximations

$$\widetilde{f}_{y_3}(y_3) \approx y_3^2,\ \widetilde{f}_{y_4}(y_4) \approx y_4^2,\ \widetilde{f}_3(z_3) \approx f_3(z_3)$$

need to be computed over the corresponding domains. The aggregated optimal PWA approximation $\widetilde{f}(z_1, z_2, z_3) \approx f(z_1)f(z_2)f(z_3)$ consists of 7 individual approximations and is given by

$$\widetilde{f}(\cdot) = {}^1\!/_4\Big(\underbrace{\widetilde{f}_{y_3}\big(\widehat{f}_a + \widetilde{f}_3(z_3)\big)}_{\widehat{y}_3} - \underbrace{\widetilde{f}_{y_4}\big(\widehat{f}_a - \widetilde{f}_4(z_3)\big)}_{\widehat{y}_4}\Big). \tag{14}$$

Here, $\widehat{f}_a$ is the function value of $\widetilde{f}_a(z_1, z_2) \approx f_1(z_1)f_2(z_2)$ at $z_1$ and $z_2$, where $\widetilde{f}_a(\cdot)$ is obtained from (11), i.e.:

$$\widehat{f}_a = {}^1\!/_4\Big(\underbrace{\widetilde{f}_{y_1}\big(\widetilde{f}_1(z_1) + \widetilde{f}_2(z_2)\big)}_{\widehat{y}_1} - \underbrace{\widetilde{f}_{y_2}\big(\widetilde{f}_1(z_1) - \widetilde{f}_2(z_2)\big)}_{\widehat{y}_2}\Big). \tag{15}$$

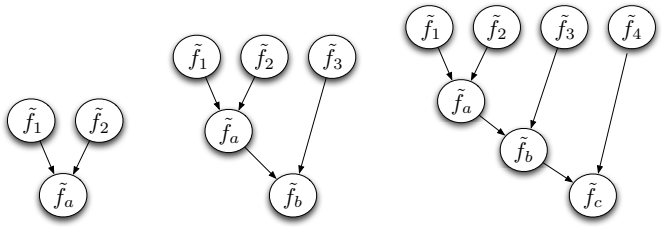
The overall PWA approximation $\widetilde{f}(z_1, z_2, z_3)$ can then be evaluated, for any $z_1,\ z_2,\ z_3 \in \mathcal{Z}$, by computing the function values of the respective approximations in the following order:

**Step 1:** $\widehat{y}_1 = \widetilde{f}_{y_1}(\widetilde{f}_1(z_1) + \widetilde{f}_2(z_2))$,
**Step 2:** $\widehat{y}_2 = \widetilde{f}_{y_2}(\widetilde{f}_1(z_1) - \widetilde{f}_2(z_2))$,
**Step 3:** $\widehat{y}_3 = \widetilde{f}_{y_3}({}^1\!/_4(\widehat{y}_1 - \widehat{y}_2) + \widetilde{f}_3(z_3))$,
**Step 4:** $\widehat{y}_4 = \widetilde{f}_{y_4}({}^1\!/_4(\widehat{y}_1 - \widehat{y}_2) - \widetilde{f}_3(z_3))$,
**Step 5:** $\widetilde{f}(z_1, z_2, z_3) = {}^1\!/_4(\widehat{y}_3 - \widehat{y}_4)$.

Such an inductive procedure can be repeated *ad-infinitum* to derive PWA approximations of any multivariable function which satisfies Assumption 2.2. In general, the PWA approximation will consists of $2p + n$ individual PWA functions, where $n$ is the number of variables in $f(z_1, \ldots, z_n)$ and $p$ is the number of products between individual subfunctions $f_j(z_j)$. As an example, for $f(\cdot) := \alpha_1 f_1(z_1)f_2(z_2)f_4(z_4) + \alpha_2 f_3(z_3)f_5(z_5)$ we have $p = 3$. We remark that inclusion of scalar multipliers $\alpha_j$ into the PWA description of the form (14)–(15) is straightforward and only requires linear scaling of the corresponding terms.

## 5. SOFTWARE IMPLEMENTATION

An algorithmic implementation of the inductive separation procedure of Section 4 is discussed next, provided that all functions are given in their symbolic representation. The procedure relies on two basic building blocks. The first one, represented by Algorithm 1, constructs the PWA approximation of a product of two functions, i.e. computes $\widetilde{f}(z_i, z_j) \approx f_i(z_i)f_j(z_j)$. Strictly speaking, the algorithm differentiates between two scenarios. If either $f_i$ or $f_j$ are




(a) Step 1. (b) Step 2. (c) Step 3.

Fig. 5. Parsing tree $\mathcal{T}$ built by Algorithm 2.

PWA functions which approximate the product of some other functions (say $f_i \approx f_p f_q$), then $\widetilde{f} \approx f_i f_j$ is computed as shown in (12)–(15). Otherwise the procedure evidenced by (7)–(11) is followed.

Algorithm 2 then utilizes this block to construct a parse tree which defines the PWA approximation of the product of multiple functions, i.e. $\prod_{i=1}^{n} f_i(z_i)$. To illustrate the procedure, consider $f(z_1, z_2, z_3, z_4) = f_1(z_1)f_2(z_2)f_3(z_3)f_4(z_4)$. First, the stack of "unexplored" functions $\mathcal{S} = \{f_4, f_3, f_2, f_1\}$ is formed. In the first pass of the `while` cycle, $f_1$ and $f_2$ are popped from the stack and the PWA approximation $\widetilde{f}_a \approx f_1 f_2$ is computed by Algorithm 1. Subsequently, $\widetilde{f}_a$ is pushed back to $\mathcal{S}$ (which then becomes $\mathcal{S} = \{f_4, f_3, \widetilde{f}_a\}$), and new nodes of the parse tree $\mathcal{T}$ are created as shown in Figure 5(a). The procedure then repeats from Step 4. I.e., $f_3$ and $\widetilde{f}_a$ are popped from $\mathcal{S}$, $\widetilde{f}_b \approx f_3 f_a$ is computed, and the parse tree is updated as illustrated in Figure 5(b). Due to Step 6, $\mathcal{S} = \{f_4, \widetilde{f}_b\}$, and the algorithm therefore performs one more pass at which $\widetilde{f}_c \approx f_4 f_b$ is created and inserted into the tree, which finally looks like in Figure 5(c). The algorithm thereupon terminates since $\mathcal{S} = \{\widetilde{f}_c\}$ contains a single element.

If the function to be approximated contains sums of products, e.g. when $f(z_1, z_2, z_3, z_4) = \alpha_1 f_1(z_1)f_2(z_2) + \alpha_2 f_3(z_3)f_4(z_4)$, separate parsing trees have to be built by Algorithm 2 for each component of the summation. We remark that treating the scaling factors $\alpha_i$ only involves scaling the bottom-most node of the corresponding tree by the respective $\alpha_i$.

**Algorithm 1** PWA approximation of $f_i(z_i)f_j(z_j)$

**INPUT:** Functions $f_i(z_i)$, $f_j(z_j)$.

**OUTPUT:** Approximation $\widetilde{f}(z_i, z_j) \approx f_i(z_i)f_j(z_j)$.

1: Obtain the PWA approximations $\widetilde{f}_i(z_i) \approx f_i(z_i)$ and $\widetilde{f}_j(z_j) \approx f_j(z_j)$ by solving two NLPs (6).
2: Get $\underline{y}_i$, $\overline{y}_i$, $\underline{y}_j$, and $\overline{y}_j$ from (10) or (13).
3: Compute the PWA approximations $\widetilde{f}_{y_i}(y_i) \approx y_i^2$ and $\widetilde{f}_{y_j}(y_j) \approx y_j^2$ on domains $[\underline{y}_i,\ \overline{y}_i]$ and $[\underline{y}_j,\ \overline{y}_j]$ by solving two NLPs (6).
4: **return** $\widetilde{f}_i(z_i)$, $\widetilde{f}_j(z_j)$, and the symbolic representation of $\widetilde{f}(z_i, z_j)$.

The parsing tree generated by Algorithm 2 can be readily used to convert the PWA approximation $\widetilde{f}(z_1, \ldots, z_n) \approx \sum_i \alpha_i \prod_j f_j(z_j)$ into a suitable mathematical model, which can subsequently be used for simulations, analysis, or control synthesis. Therefore we have created a software tool which takes a parsing tree $\mathcal{T}$ (or several such trees to accommodate for sums of products of functions), and *automatically* generates the corresponding HYSDEL representation of such a PWA approximation. It is available for free download at http://www.kirp.chtf.stuba.sk/~sw/.

**Algorithm 2** PWA approximation of $\prod_{i=1}^{n} f_i(z_i)$

**INPUT:** Functions $f_i(z_i)$.

**OUTPUT:** $\widetilde{f}(z_1, \ldots, z_n) \approx \prod_{i=1}^{n} f_i(z_i)$.

1: Create an empty last-in-first-out stack $\mathcal{S}$ and an empty tree $\mathcal{T}$.
2: Push $f_i(z_i)$, $i = n, \ldots, 1$ to the stack $\mathcal{S}$.
3: **while** $\mathcal{S}$ has more than one element **do**
4: Pop two elements $f_j(z_j)$ and $f_k(z_k)$ from $\mathcal{S}$.
5: Obtain $\widetilde{f}_j(z_j)$, $\widetilde{f}_j(z_j)$, and $\widetilde{f}(z_j, z_k) \approx f_j(z_j)f_k(z_k)$ by calling Algorithm 1.
6: Push $\widetilde{f}(z_j, z_k)$ to $\mathcal{S}$.
7: Create nodes $\widetilde{f}_j(z_j)$, $\widetilde{f}_k(z_k)$ and insert them to $\mathcal{T}$.
8: Create a node $\widetilde{f}(z_j, z_k)$ and append it as a child of nodes $\widetilde{f}_j(z_j)$ and $\widetilde{f}_k(z_k)$.
9: **end while**
10: **return** Tree $\mathcal{T}$ representing $\widetilde{f}(z_1, \ldots, z_n) \approx \prod_{i=1}^{n} f_i(z_i)$.

## 6. CASE STUDY

Consider a continuous stirred tank reactor (CSTR) where the reaction $A \rightarrow B$ takes place. The source compound is pumped into the reactor at a constant inflow with a constant concentration. The chemical reaction is exothermic and a coolant liquid is therefore pumped into the reactor's jacket to prevent overheating. The input temperature of the coolant is constant, while its flow rate $q_c$ can be manipulated and is considered an exogenous input. Concentration of the reactant $c_A$ inside of the reactor, temperature of the reactor mixture $\vartheta$, and temperature of the cooling liquid in the jacket $\vartheta_c$ are the state variables of the CSTR. The normalized material and energy balances of such a reactor are then given by

$$\begin{aligned} \dot{c}_A &= \alpha_1 - \alpha_2 c_A - \alpha_3 c_A e^{-\beta/\vartheta}, \\ \dot{\vartheta} &= \alpha_4 - \alpha_5 \alpha_2 c_A e^{-\beta/\vartheta} + \alpha_6 \vartheta + \alpha_7 \vartheta_c, \\ \dot{\vartheta}_c &= \alpha_8 q_c + \alpha_9(\vartheta - \vartheta_c) - \alpha_{10}\vartheta_c q_c, \end{aligned} \tag{16}$$

with constants $\alpha_i$ and $\beta$. The state and input variables are considered to belong to intervals $c_A \in [4,\ 4.2]$ mol $\cdot$ m$^{-3}$, $\vartheta \in [300,\ 320]$ K, $\vartheta_c \in [290,\ 310]$ K, and $q_c \in [0.002,\ 0.02]$ m$^3 \cdot$ h$^{-1}$.

The model features two nonlinearities: $\vartheta_c q_c$ and $c_A e^{-\beta/\vartheta}$, both of which satisfy Assumption 2.2. Since the first one involves a direct product of two variables, its PWA approximation $\widetilde{f}_a \approx \vartheta_c q_c$ can be obtained as in (8) by first defining $y_1 = \vartheta_c + q_c$, $y_2 = \vartheta_c - q_c$, followed by approximating the functions $y_1^2$ and $y_2^2$ by $\widetilde{f}_{y_1}(y_1)$ and $\widetilde{f}_{y_2}(y_2)$, respectively. Hence, the approximation $\widetilde{f}_1(\vartheta_c, q_c) \approx \vartheta_c q_c$ is represented by

$$\widetilde{f}_1(\vartheta_c, q_c) = 1/4\big(\widetilde{f}_{y_1}(\vartheta_c + q_c) - \widetilde{f}_{y_2}(\vartheta_c - q_c)\big). \tag{17}$$

The second nonlinearity can be approximated as in (11). First, the PWA approximation $\widetilde{g}(\vartheta) \approx e^{-\beta/\vartheta}$ is computed by solving (6). Then, $y_3 = c_A + e^{-\beta/\vartheta}$, $y_4 = c_A - e^{-\beta/\vartheta}$ are defined, followed by computing the respective PWA approximations $\widetilde{f}_{y_3}(y_3) \approx y_3^2$ and $\widetilde{f}_{y_4}(y_4) \approx y_4^2$. $\widetilde{f}_2(c_A, \vartheta) \approx c_A e^{-\beta/\vartheta}$ is thus given by

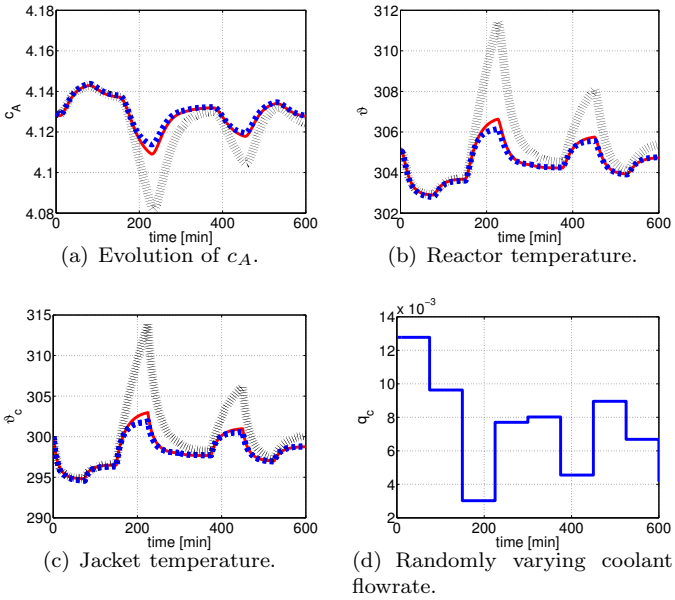
$$\widetilde{f}_2(c_A, \vartheta) = {}^1\!/_4\Big(\widetilde{f}_{y_3}\big(c_A + \widetilde{g}(\vartheta)\big) - \widetilde{f}_{y_4}\big(c_A - \widetilde{g}(\vartheta)\big)\Big) \quad (18)$$

The overall PWA approximation of the original nonlinear system $\dot{x} = f(x,u)$ with $x = [c_A,\ \vartheta,\ \vartheta_c]^T$ and $u = q_c$ is thus

$$\begin{aligned} \dot{c}_A &\approx \alpha_1 - \alpha_2 c_A - \alpha_3 \widetilde{f}_2(c_A, \vartheta)\big), \\ \dot{\vartheta} &\approx \alpha_4 - \alpha_5\alpha_2 \widetilde{f}_2(c_A, \vartheta) + \alpha_6\vartheta + \alpha_7\vartheta_c + \vartheta, \\ \dot{\vartheta}_c &\approx \alpha_8 q_c + \alpha_9(\vartheta - \vartheta_c) - \alpha_{10}\widetilde{f}_1(\vartheta_c, q_c), \end{aligned} \quad (19)$$

which can be easily converted into the general PWA form (2) as described in the previous section.

To assess approximation accuracy, we have investigated the open-loop evolution of the original nonlinear model (16) and compared it to the behavior of its PWA approximation (19). To derive the PWA model, we have chosen 3 regions for $\widetilde{f}_{y_1}(\cdot)$, $\widetilde{f}_{y_2}(\cdot)$ in (17) and $\widetilde{f}_{y_3}(\cdot)$, $\widetilde{f}_{y_4}(\cdot)$ in (18), and $N = 2$ for $\widetilde{g}(\theta) \approx e^{-\beta/\vartheta}$. The simulation results are shown in Figure 6. To better illustrate advantages of the PWA approximation, the simulation scenario also shows evolution of linearized version of (16) around the nominal steady state $c_A^s = 4.13$, $\vartheta^s = 304$, $\vartheta_c^s = 297$, and $q_c^s = 0.006$. As can be seen from the results, the PWA approximation clearly outperforms the model based on a single linearization. Specifically, the model (19) provides a 15 times more accurate tracking of the nonlinear profile compared to the linear model. Important to notice is that the PWA model consists of 14 local linear models. By increasing $N$ to 7 when approximating $\widetilde{f}_{y_1}(\cdot)$, $\widetilde{f}_{y_2}(\cdot)$ in (17) and $\widetilde{f}_{y_3}(\cdot)$, $\widetilde{f}_{y_4}(\cdot)$ in (18), the approximation accuracy is 60 times better compared to the linear model. The cost to be paid is the increased model complexity, which would then consist of 30 regions.




(a) Evolution of $c_A$. (b) Reactor temperature. (c) Jacket temperature. (d) Randomly varying coolant flowrate.

Fig. 6. Simulation results for the CSTR. Red line: nonlinear model (16), blue dashed line: PWA model (19), black dotted line: linear approximation.

## 7. CONCLUSIONS

We have shown that a large class of dynamical systems with nonlinear vector fields can be approximated by PWA systems of fixed complexity in an optimal manner. The procedure boils down to solving a series of one-dimensional problems for which efficient solution methods exist. Derivation of the approximation can be easily automated and the HYSDEL variant of the PWA approximation can be generated, hence allowing for subsequent control synthesis based on the hybrid model.

## ACKNOWLEDGMENT

The authors are pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic under the grants 1/0071/09 and 1/0095/11. This work was supported by the Slovak Research and Development Agency under the contract No. LPP-0092-07.

## REFERENCES

Adjiman, C.S., Androulakis, I.P., Maranas, C.D., and Floudas, C.A. (1996). A global optimization method, $\alpha$BB for process design. *Computers and Chemical Engineering*, 20, 419–424.

Chachuat, B., Singer, A.B., and Barton, P.I. (2006). Global methods for dynamic optimization and mixed-integer dynamic optimization. *Ind. Eng. Chem. Res.*, 45(25), 8373–8392.

Ferrari-Trecate, G. (2005). Hybrid Identification Toolbox (HIT). Available from http://www-rocq.inria.fr/who/Giancarlo. Ferrari-Trecate/HIT_toolbox.html.

Ferrari-Trecate, G., Muselli, M., Liberati, D., and Morari, M. (2001). Identification of Piecewise Affine and Hybrid Systems. In *Proc. on the American Control Conference*, 3521–3526. Arlington (VA), USA.

Kvasnica, M. and Herceg, M. (2010). HYSDEL 3.0. Available from http://kirp.chtf.stuba.sk/~kvasnica/hysdel3/.

Papamichail, I. and Adjiman, C.S. (2004). Global optimization of dynamic systems. *Computers and Chemical Engineering*, 28, 403–415.

Roll, J., Bemporad, A., and Ljung, L. (2004). Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40, 37–50.

Sontag, E.D. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Trans. on Automatic Control*, 26(2), 346–358.

Torrisi, F. and Bemporad, A. (2004). HYSDEL — A tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12, 235–249.

Williams, H. (1993). *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition.