

## A Computational Approach to Polynomial Conservation Laws\*

Aurélien Desoievres<sup>†</sup>, Alexandru Iosif<sup>‡</sup>, Christoph Lüders<sup>§</sup>, Ovidiu Radulescu<sup>¶</sup>,  
Hamid Rahkooy<sup>||</sup>, Matthias Seiß<sup>#</sup>, and Thomas Sturm<sup>††</sup>

**Abstract.** For polynomial ODE models, we introduce and discuss the concepts of exact and approximate conservation laws, which are the first integrals of the full and truncated sets of ODEs. For fast-slow systems, truncated ODEs describe the fast dynamics. We define compatibility classes as subsets of the state space, obtained by equating the conservation laws to constants. A set of conservation laws is complete when the corresponding compatibility classes contain a finite number of steady states. Complete sets of conservation laws can be used for model order reduction and for studying the multistationarity of the model. We provide algorithmic methods for computing linear, monomial, and polynomial conservation laws of polynomial ODE models and for testing their completeness. The resulting conservation laws and their completeness are either independent or dependent on the parameters. In the latter case, we provide parametric case distinctions. In particular, we propose a new method to compute polynomial conservation laws by comprehensive Gröbner systems and syzygies.

**Key words.** first integrals, chemical reaction networks, polynomial conservation laws, syzygies, comprehensive Gröbner systems, polynomial ODEs

**MSC codes.** 37N25, 92C45, 14-04, 34C14, 13P10

**DOI.** 10.1137/22M1544014

**1. Introduction.** Several important models in physics, chemistry, biology, ecology, economics, and engineering can be written as systems of polynomial ODEs. *Conservation laws* are first integrals of such models, i.e., quantities that are constant on any solution of the ODEs. When they exist, they can be used for model order reduction [11]. A reduced model can be more easily simulated and analyzed. For example, ODEs with  $n$  variables and  $n - 1$  independent conservation laws are solvable by quadratures [3].

\*Received by the editors December 28, 2022; accepted for publication (in revised form) by M. Chaves November 7, 2023; published electronically March 12, 2024.

<https://doi.org/10.1137/22M1544014>

**Funding:** This work was funded by the interdisciplinary bilateral project ANR-17-CE40-0036 and DFG-391322026 SYMBIONT. The fifth author received funding through Royal Society enhancement grant BKR01980 BK00.04.

<sup>†</sup>University of Montpellier and CNRS UMR5235 LPHI, Montpellier, France ([desoievres.aurelien@gmail.com](mailto:desoievres.aurelien@gmail.com)).

<sup>‡</sup>Rey Juan Carlos University, Madrid, Spain ([alejandroiosif@gmail.com](mailto:alejandroiosif@gmail.com)).

<sup>§</sup>University of Bonn, Bonn, Germany ([chris@cfs.de](mailto:chris@cfs.de)).

<sup>¶</sup>Corresponding author. University of Montpellier and CNRS UMR5235 LPHI, Montpellier, France ([ovidiu.radulescu@umontpellier.fr](mailto:ovidiu.radulescu@umontpellier.fr)).

<sup>||</sup>University of Oxford, Oxford OX1 4BH, UK ([rahkooy@maths.ox.ac.uk](mailto:rahkooy@maths.ox.ac.uk)).

<sup>#</sup>University of Kassel, Kassel, Germany ([mseiss@mathematik.uni-kassel.de](mailto:mseiss@mathematik.uni-kassel.de)).

<sup>††</sup>CNRS, INRIA, and the University of Lorraine, Nancy, France, and Max Planck Institute for Informatics, Saarbrücken, Germany ([sturm@mpi-inf.mpg.de](mailto:sturm@mpi-inf.mpg.de)).

The use of first integrals for eliminating variables is an exact model reduction method [11]. However, many model reduction methods are approximate. For instance, the quasi-steady state approximation is an approximate model reduction method based on the elimination of fast variables [21, 29, 31, 44, 56, 70]. This reduction method fails when the fast dynamics has first integrals. We introduce here the *approximate conservation laws*, a key concept for model reduction by variable pooling used when the fast dynamics has first integrals [5, 21, 57].

Conservation laws are also used for studying the steady states of ODE models, for instance in applications that deal with multistationarity [28]. In the case of linear conservation laws, the set of steady states can be partitioned into stoichiometric compatibility classes [27] that are characterized by fixed, different values of the linear conservation laws. Here we extend this definition to polynomial conservation laws. Compatibility classes are invariant sets of the ODE flow and, under some conditions that we call *completeness*, contain a finite number of steady states. The completeness of approximate conservation laws is also important for model reduction applications [21].

A lot of effort has been put into computing linear conservation laws of ODE models [59, 64]. Nonlinear conservation laws have been addressed using Darboux polynomials [16, 49, 50, 55]. Monomial first integrals were discussed in [32, 49]. For a thorough introduction to first integrals and related computational methods, refer to Zhang's book [74]. Polynomial first integrals were used for formal proofs of hybrid systems safety in [63], where they were computed using the method of undetermined coefficients.

In this paper, we propose a new method for computing polynomial conservation laws using syzygies, which are relations among the polynomial right-hand side (r.h.s.) of the ODEs. Furthermore, we use comprehensive Gröbner bases techniques for partitioning the parameter space into branches such that the syzygies (and hence, conservation laws) are invariant under evaluation in each branch. We also propose algorithmic solutions for testing the independence and the completeness of the computed conservation laws.

The structure of this paper is as follows. In the next section, we introduce the polynomial ODE models. We show that any polynomial ODE model can be written as a chemical reaction network (CRN) model. In section 3 we discuss exact and approximate conservation laws, which are important in model reduction applications. In section 4 we provide methods to test the independence and completeness of the conservation laws. Section 5 exposes extant methods for computing linear conservation laws. Section 6.1 is dedicated to monomial conservation laws. In section 6.2 we present a new method to compute polynomial conservation laws using syzygies.

**2. Models.** In this paper, we consider polynomial ODE models

$$(2.1) \quad \dot{x}_1 = f_1(\mathbf{k}, \mathbf{x}), \dots, \dot{x}_n = f_n(\mathbf{k}, \mathbf{x}),$$

whose r.h.s. are integer coefficient polynomials in the variables  $\mathbf{x}$  and parameters  $\mathbf{k}$   $f_i(\mathbf{k}, \mathbf{x}) \in \mathbb{Z}[\mathbf{k}, \mathbf{x}] = \mathbb{Z}[k_1, \dots, k_r, x_1, \dots, x_n]$ . Up to a redefinition of parameters, the functions  $f_i$  can be considered homogeneous of degree one in  $\mathbf{k}$ . We denote the vector of r.h.s. of (2.1) by

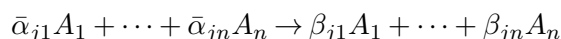
$$\mathbf{F}(\mathbf{k}, \mathbf{x}) = (f_1(\mathbf{k}, \mathbf{x}), f_2(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))^\top.$$

For certain applications we consider that the functions  $f_i$  have the particular structure of CRNs [2]:

$$(2.2) \quad f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^r S_{ij} k_j \mathbf{x}^{\alpha_j}.$$

The monomials  $\mathbf{x}^{\alpha_j} = x_1^{\alpha_{j1}} \cdots x_n^{\alpha_{jn}}$  appearing in the r.h.s. of (2.2) are defined by  $r$  multi-indices  $\alpha_j = (\alpha_{j1}, \dots, \alpha_{jn}) \in \mathbb{N}^n$  and for each monomial the parameter  $k_j$  represents a rate constant. The parameters  $\mathbf{k} = (k_1, \dots, k_r)$  take values in  $\mathbb{R}_+^*$  and the integer coefficients  $S_{ij}$  form a matrix  $\mathbf{S} = (S_{ij}) \in \mathbb{Z}^{n \times r}$ , which is called the stoichiometric matrix.

Mass action CRNs with  $n$  species  $A_1, \dots, A_n$  and concentrations  $\mathbf{x} = (x_1, \dots, x_n)$  satisfy (2.2). For a mass action reaction



we have  $S_{ij} = \beta_{ji} - \bar{\alpha}_{ji}$  and the reaction rate is  $k_j \mathbf{x}^{\bar{\alpha}_j}$ , i.e., the stoichiometric coefficients  $\bar{\alpha}_j$  and the multi-indices  $\alpha_j$  coincide. This constraint leads to algebraic properties of CRNs exploited in *chemical reaction network theory*, which was initiated by Horn, Jackson, and Feinberg [26, 27]. However, this may not be the case in general, meaning that the multi-indices  $\alpha_j$  and  $\bar{\alpha}_j$  in  $S_{ij} = \beta_{ji} - \bar{\alpha}_{ji}$  and  $k_j \mathbf{x}^{\bar{\alpha}_j}$  are not necessarily equal. As a matter of fact the CRNs with monomial rate functions used here do not reduce the generality of polynomial ODE models described by (2.1). This is shown by the following theorem.

**Theorem 2.1.** *Any system of polynomial ODEs*

$$\dot{x}_1 = f_1(\mathbf{k}, \mathbf{x}), \dots, \dot{x}_n = f_n(\mathbf{k}, \mathbf{x}),$$

where each  $f_i(\mathbf{k}, \mathbf{x}) \in \mathbb{Z}[\mathbf{k}, \mathbf{x}]$  is an integer coefficient polynomial in  $\mathbf{k}$  and  $\mathbf{x}$ , homogeneous of degree one in  $\mathbf{k} = (k_1, \dots, k_r)$ , can be written as (2.1), (2.2) for appropriate matrix  $\mathbf{S}$  and multi-indices  $\alpha_j$ .

*Proof.* A constructive proof is given by Algorithm 2.1. ■

**Remark 2.2.** In many applications, parameters are not symbolic but given by their numerical values, sometimes linearly dependent over the integers. In this case, the independent parameters and the stoichiometric matrix are hidden behind these numbers. We propose in Algorithm 2.2 a way to still obtain descriptions of the form (2.2), where the  $k_j$  are linearly independent over the integers and the stoichiometric coefficients are clearly visible. For example, the ODE system  $\dot{x}_1 = 1.5x_1x_2$ ,  $\dot{x}_2 = x_2 + 3x_1x_2$  is transformed into  $\dot{x}_1 = k_1x_1x_2$ ,  $\dot{x}_2 = k_2x_2 + 2k_1x_1x_2$  with the stoichiometric matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

**Remark 2.3.** In chemical and biochemical applications CRN variables have positive values and the ODE flow preserves the positive orthant. A sufficient condition for this property is that inputs of Algorithm 2.1 satisfy

$$\beta_{iji} > 0, \text{ whenever } z_{ij} < 0,$$

**Algorithm 2.1.** Smatrix.

**Input:** An ODE system whose r.h.s. are sums of rational monomials (meaning we also allow negative powers) in  $\mathbf{k}$  and  $\mathbf{x}$ , homogeneous of degree one in  $\mathbf{k}$ , with integer coefficients:

$$\dot{x}_i = f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^{r_i} z_{ij} k_{ij} \mathbf{x}^{\beta_{ij}},$$

where  $z_{ij} \in \mathbb{Z}$ ,  $\beta_{ij} \in \mathbb{Z}^n$ ,  $k_{ij} \in \{k_1, \dots, k_r\}$ , for  $1 \leq i \leq n$ ,  $1 \leq j \leq r_i$ .

**Output:** An integer coefficient matrix  $\mathbf{S}$  and a set of multi-indices  $\alpha_1, \dots, \alpha_r \in \mathbb{N}^n$  such that  $f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^r S_{ij} k_j \mathbf{x}^{\alpha_j}$  for all  $1 \leq i \leq n$ .

1: Compute a list of distinct monomials

$$\{M_j = k_j \mathbf{x}^{\alpha_j}, 1 \leq j \leq r\} = \bigcup_{i=1}^n \bigcup_{j=1}^{r_i} \{k_{ij} \mathbf{x}^{\beta_{ij}}\}.$$

Two monomials  $k_{ij} \mathbf{x}^{\beta_{ij}}$ ,  $k_{i'j'} \mathbf{x}^{\beta_{i'j'}}$  are distinct if  $\beta_{ij} \neq \beta_{i'j'}$  or  $k_{ij} \neq k_{i'j'}$ .

```

2: for  $i := 1$  to  $n$  do
3:   for  $j := 1$  to  $r$  do
4:      $S_{ij} = 0$ 
5:     for  $l := 1$  to  $r_i$  do
6:       if  $k_{il} \mathbf{x}^{\beta_{il}} = M_j$  then
7:          $S_{ij} = S_{ij} + z_{il}$ 
8:       end if
9:     end for
10:   end for
11: end for

```

where  $\beta_{iji}$  is the  $i$ th coordinate of the vector  $\beta_{ij}$ . This condition ensures that rates of reactions consuming a species  $x_i$  tend to zero whenever the concentration of this species tends to zero. The corresponding condition on the system (2.1) reads

$$\alpha_{ji} > 0, \text{ whenever } S_{ij} < 0.$$

However, these conditions can be lifted in other applications.

**Remark 2.4.** The advantage of the form (2.2) and the reason to use it even for nonchemical applications is, as will be shown later in sections 5 and 6.1, that this form facilitates the computation of linear and monomial conservation laws.

**3. Exact versus approximate conservation laws.** We start this section with a motivating example.

**Example 3.1.** Let us consider the irreversible Michaelis–Menten mechanism, used as a model of enzymatic reactions. We choose rate constants corresponding to the so-called quasi-equilibrium, as done by Michaelis and Menten [51]. The reaction network for this model is

---

**Algorithm 2.2.** CompareAndSplitCoefficients.

---

**Input:** An ODE system of the form  $\dot{x}_i = f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^m B_{ij} \mathbf{x}^{\alpha_j}$ , where  $\alpha_j \in \mathbb{Z}^n$ ,  $B_{ij} \in \mathbb{R}$ ,  $m$  is the number of distinct  $\alpha_j$ .

**Output:** A set of incommensurate real numbers  $\{k_1, \dots, k_r\}$  and a set of integers  $\{z_{ij}, 1 \leq i \leq n, 1 \leq j \leq r_i\}$  such that

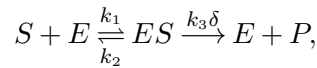
$$f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^{r_i} z_{ij} k_{ij} \mathbf{x}^{\beta_{ij}},$$

where  $k_{ij} \in \{k_1, \dots, k_r\}$ , for  $1 \leq i \leq n$ ,  $1 \leq j \leq r_i$ .

```

1: for h := 1 to m do
2:   for i := 1 to n do
3:     for j := 1 to i do
4:       if  $B_{jh} = 0$  then
5:          $z_{jh} := 1, k_{jh} := 0, z_{ih} := \text{sign}(B_{ih}), k_{ih} := B_{ih}/z_{ih}$ 
6:          $\beta_{ih} := \alpha_h, \beta_{jh} := \alpha_h$ 
7:       else if  $B_{ih}/B_{jh} \in \mathbb{Z}$  then
8:          $z_{ih} := \text{sign}(B_{ih})|B_{ih}/B_{jh}|, k_{ih} := B_{ih}/z_{ih}, z_{jh} := \text{sign}(B_{jh}), k_{jh} := B_{jh}/z_{jh}$ 
9:          $\beta_{ih} := \alpha_h, \beta_{jh} := \alpha_h$ 
10:      else if  $B_{jh}/B_{ih} \in \mathbb{Z}$  then
11:         $z_{jh} := \text{sign}(B_{jh})|B_{jh}/B_{ih}|, k_{jh} := B_{jh}/z_{jh}, z_{ih} := \text{sign}(B_{ih}), k_{ih} := B_{ih}/z_{ih}$ 
12:         $\beta_{ih} := \alpha_h, \beta_{jh} := \alpha_h$ 
13:      else
14:         $z_{jh} := \text{sign}(B_{jh}), k_{jh} := B_{jh}/z_{jh}, z_{ih} := \text{sign}(B_{ih}), k_{ih} := B_{ih}/z_{ih}$ 
15:         $\beta_{ih} := \alpha_h, \beta_{jh} := \alpha_h$ 
16:      end if
17:    end for
18:  end for
19: end for
20: for i := 1 to n do
21:    $r_i := m, j := 1$ 
22:   while  $j < r_i$  do
23:     if  $k_{ij} = 0$  then
24:       for h := j to  $r_i - 1$  do
25:         /*re-indexing*/  $z_{ij} k_{ij} \mathbf{x}^{\beta_{ij}} := z_{ij+1} k_{ij+1} \mathbf{x}^{\beta_{ij+1}}$ 
26:       end for
27:        $z_{ir_i} k_{ir_i} \mathbf{x}^{\beta_{ir_i}} := 0$ 
28:        $r_i := r_i - 1$ 
29:     end if
30:     j := j + 1
31:   end while
32: end for
```

---



where  $S$  is a substrate,  $E$  is an enzyme,  $ES$  is an enzyme-substrate complex, and  $k_1, k_2, k_3$  are rate constants of the same order of magnitude. Here  $0 < \delta < 1$  is a small positive scaling parameter, indicating that the third rate constant is small.

According to mass-action kinetics, the concentrations  $x_1 = [S]$ ,  $x_2 = [SE]$ , and  $x_3 = [E]$  satisfy the system of ODEs

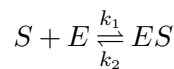
$$\begin{aligned} (3.1a) \quad & \dot{x}_1 = -k_1 x_1 x_3 + k_2 x_2, \\ (3.1b) \quad & \dot{x}_2 = k_1 x_1 x_3 - k_2 x_2 - \delta k_3 x_2, \\ (3.1c) \quad & \dot{x}_3 = -k_1 x_1 x_3 + k_2 x_2 + \delta k_3 x_2. \end{aligned}$$

We consider the truncated system of ODEs

$$\begin{aligned} (3.2a) \quad & \dot{x}_1 = -k_1 x_1 x_3 + k_2 x_2, \\ (3.2b) \quad & \dot{x}_2 = k_1 x_1 x_3 - k_2 x_2, \\ (3.2c) \quad & \dot{x}_3 = -k_1 x_1 x_3 + k_2 x_2, \end{aligned}$$

which is obtained by setting  $\delta = 0$  in (3.1). If  $k_i, 1 \leq i \leq 3$ , are of order  $O(\delta^0)$ , then the truncated system (3.2) describes the dynamics of the model on fast timescales of order  $O(\delta^0)$ .

The steady state of the fast dynamics is obtained by equating to zero the r.h.s. of (3.2). The resulting condition is called quasi-equilibrium (QE), because it means that the complex formation rate  $k_1 x_1 x_3$  is equal to the complex dissociation rate  $k_2 x_2$ . In other words, the reversible reaction



is at equilibrium. The QE condition is reached only at the end of the fast dynamics and is satisfied with a precision of order  $O(\delta)$  during the slow dynamics [33].

We introduce the linear combinations of variables  $x_4 = x_1 + x_2$  and  $x_5 = x_2 + x_3$ , corresponding to the total substrate and total enzyme concentrations, respectively. Addition of (3.1b) and (3.1c) leads to  $\dot{x}_5 = 0$ , which means that for solutions of the full system,  $x_5$  is constant for all times. We will call such a quantity an *exact conservation law*. The addition of (3.2a) and (3.2b) leads to  $\dot{x}_4 = 0$ . This means that  $x_4$  is constant for solutions of the truncated dynamics, valid at short times  $t = O(\delta^0)$ . However, addition of (3.1a) and (3.1b) leads to

$$\dot{x}_4 = -\delta k_3 x_2,$$

meaning that  $x_4$  evolves slowly under the full dynamics.  $x_4$  is not constant at large times  $t = O(\delta^{-1})$ . We call such a quantity an *approximate conservation law*. The quantity  $x_5$  is both an exact and an approximate conservation law, because it is conserved by both (3.2) and (3.1) and therefore it is constant at all times.

**Remark 3.2.** The exact and approximate conservation laws can be used as a new parameter and as a slow variable of the model, respectively. Two extant variables can then be eliminated as such  $x_2 = x_4 - x_1, x_3 = k_4 + x_1 - x_4$ , where  $k_4 = x_5$  is the new parameter. In the remaining variables  $x_1, x_4$  the full system reads

$$(3.3a) \quad \dot{x}_1 = -k_1 x_1 (k_4 + x_1 - x_4) + k_2 (x_4 - x_1),$$

$$(3.3b) \quad \dot{x}_4 = -\delta k_3 (x_4 - x_1).$$

The transformed model (3.3) is typically a slow-fast system with  $x_1$  the fast and  $x_4$  the slow variable [29, 70] and can be further reduced using the quasi-steady state approximation. In the transformed model the fast dynamics (3.3a) has a nondegenerate, hyperbolic steady state. This was not the case for the fast dynamics (3.2) in the original formulation of the problem, which has a degenerate steady state manifold. However, the foliations defined by hyperplanes on which the two conservation laws are constant intersect transversally the degenerate state manifold, resulting in nondegenerate steady states. This property, called completeness, will be detailed in section 4.

More generally, let us assume that the r.h.s. of the system in (2.1) can be decomposed into a sum of two terms, the first one dominating (being much larger than the second). This decomposition can be obtained by rescaling the model variables and parameters by powers of a small scaling parameter  $\delta$ , where  $0 < \delta < 1$ . To this aim, we define  $\bar{\mathbf{k}} = (\bar{k}_1, \dots, \bar{k}_r)$  and  $\bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_m)$  such that  $k_i = \delta^{e_i} \bar{k}_i$ ,  $x_i = \delta^{d_i} \bar{x}_i$  and obtain

$$(3.4) \quad f_i(\mathbf{k}, \mathbf{x}) = f_i^{(1)}(\mathbf{k}, \mathbf{x}) + f_i^{(2)}(\mathbf{k}, \mathbf{x}),$$

where

$$f_i^{(1)}(\mathbf{k}, \mathbf{x}) = \delta^{b_i} \bar{f}_i^{(1)}(\bar{\mathbf{k}}, \bar{\mathbf{x}}) \quad \text{and} \quad f_i^{(2)}(\mathbf{k}, \mathbf{x}) = \delta^{b'_i} \bar{f}_i^{(2)}(\bar{\mathbf{k}}, \bar{\mathbf{x}}, \delta)$$

with  $b_i < b'_i$  (for details see [21, 44]).

For a small parameter  $\delta$ , the terms  $\delta^{b'_i} \bar{f}_i^{(2)}(\bar{\mathbf{k}}, \bar{\mathbf{x}}, \delta)$  are dominated by the terms  $\delta^{b_i} \bar{f}_i^{(1)}(\bar{\mathbf{k}}, \bar{\mathbf{x}})$ . This justifies introducing the *truncated system* as the system of ODEs obtained by keeping only the lowest order, dominant terms of the system in (2.1), namely

$$(3.5) \quad \dot{x}_1 = f_1^{(1)}(\mathbf{k}, \mathbf{x}), \dots, \dot{x}_n = f_n^{(1)}(\mathbf{k}, \mathbf{x}).$$

We denote the vector of r.h.s. of the truncated system by

$$\mathbf{F}^{(1)}(\mathbf{k}, \mathbf{x}) = (f_1^{(1)}(\mathbf{k}, \mathbf{x}), \dots, f_n^{(1)}(\mathbf{k}, \mathbf{x}))^\top.$$

**Remark 3.3.** Depending on the application, the truncated system in (3.5) can consist of fewer ODEs than the system in (2.1). For instance, in the case of model reduction of fast-slow systems, only the ODEs corresponding to fast variables are included in the truncated system. In this case, the truncated system describes the fast timescale dynamics.



**Definition 3.4.**

- (a) A function  $\phi(\mathbf{k}, \mathbf{x})$  is a parametric exact conservation law if there exists a nonempty semialgebraic set  $V_{\mathbf{k}} \subseteq \mathbb{R}_+^r$  such that for all  $\mathbf{k} \in V_{\mathbf{k}}$ ,  $\phi(\mathbf{k}, \mathbf{x})$  is a first integral of the full system in (2.1), i.e., if

$$\forall \mathbf{k} \in V_{\mathbf{k}} \quad \forall \mathbf{x} \in \mathbb{R}_+^n \quad \left( \sum_{i=1}^n \frac{\partial \phi}{\partial x_i}(\mathbf{k}, \mathbf{x}) f_i(\mathbf{k}, \mathbf{x}) = 0 \right).$$

- (b) A function  $\phi(\mathbf{k}, \mathbf{x})$  is a parametric approximate conservation law if there exists a nonempty semialgebraic set  $V_{\mathbf{k}} \subseteq \mathbb{R}_+^r$  such that for all  $\mathbf{k} \in V_{\mathbf{k}}$ ,  $\phi(\mathbf{k}, \mathbf{x})$  is a first integral of the truncated system in (3.5), i.e., if

$$\forall \mathbf{k} \in V_{\mathbf{k}} \quad \forall \mathbf{x} \in \mathbb{R}_+^n \quad \left( \sum_{i=1}^n \frac{\partial \phi}{\partial x_i}(\mathbf{k}, \mathbf{x}) f_i^{(1)}(\mathbf{k}, \mathbf{x}) = 0 \right).$$

- (c) A parametric exact (approximate) conservation law of the form

$$\phi(\mathbf{k}, \mathbf{x}) = c_1(\mathbf{k})x_1 + \cdots + c_n(\mathbf{k})x_n$$

with  $c_i(\mathbf{k}) \in \mathbb{R}[\mathbf{k}]$  is called an exact (approximate) linear conservation law. If  $c_i \geq 0$  for  $1 \leq i \leq n$ , the linear conservation law is called semipositive.

- (d) A parametric exact (approximate) conservation law of the form  $\phi(\mathbf{k}, \mathbf{x}) = a(\mathbf{k})x_1^{m_1} \cdots x_n^{m_n}$  with  $m_i \in \mathbb{Z}$  and  $a(\mathbf{k}) \in \mathbb{R}[\mathbf{k}]$  is called an exact (approximate) rational monomial conservation law. If  $m_i \in \mathbb{Z}_+$  for  $1 \leq i \leq n$ , the conservation law is called monomial. For simplicity, in this paper, we will call both types monomial.
- (e) A parametric exact (approximate) conservation law of the form

$$\sum_{i=1}^s a_i(\mathbf{k})x_1^{m_{1i}} \cdots x_n^{m_{ni}}$$

with  $m_{ji} \in \mathbb{Z}_+$  and  $a_i(\mathbf{k}) \in \mathbb{R}[\mathbf{k}]$  is called a parametric exact (approximate) polynomial conservation law.

**Remark 3.5.** In some applications we are interested in conservation laws  $\phi(\mathbf{x})$  that do not depend on  $\mathbf{k}$ . For instance, in model reduction of a fast-slow system, approximate conservation laws that are not exact represent variables of the reduced model [5, 21, 33, 56, 57]. When these conservation laws depend only on  $\mathbf{x}$ , one can show that these approximate conservation laws are always slower than the variables which compose them, a prerequisite for fast-slow reduction [21]. Moreover this ensures, for CRN models, that the reduced model is obtained by graph rewriting operations, i.e., by species and reactions pooling and pruning (see [56]).

**Remark 3.6.** Following Remark 3.5, in this paper, we use the following convention. An unconditional conservation law is a function  $\phi(\mathbf{x})$  that does not depend on  $\mathbf{k}$  and satisfies the conditions of Definition 3.4, while a parametric conservation law is a function  $\phi(\mathbf{k}, \mathbf{x})$  that does depend on  $\mathbf{k}$ .



**4. Independence and completeness.** Some variables  $x_i$  may not appear in a conservation law  $\phi(\mathbf{k}, \mathbf{x})$ . This means, in the case of a linear conservation law, that some coefficients  $c_i$  may be zero or, in the case of a nonlinear conservation law, that some partial derivatives  $\frac{\partial \phi}{\partial x_i}(\mathbf{k}, \mathbf{x})$  may vanish. If  $r$  is the number of all nonzero quantities  $\frac{\partial \phi}{\partial x_i}(\mathbf{k}, \mathbf{x})$ , then we say that the conservation law *depends on  $r$  variables*.

**Definition 4.1.** An exact or approximate conservation law depending on  $r$  variables is called simple if it can't be split into the sum or the product of two conservation laws such that at least one of them depends on a number of variables  $r'$  with  $1 \leq r' < r$ .

**Definition 4.2.** For  $\mathbf{k} \in \mathbb{R}_+^r$  a steady state  $\mathbf{x}$  is a positive solution of  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = 0$ . We denote the steady state variety by  $\mathcal{S}_{\mathbf{k}}$ . A steady state is degenerate or nondegenerate if the Jacobian  $D_{\mathbf{x}}\mathbf{F}(\mathbf{k}, \mathbf{x})$  is singular or regular, respectively.

Degeneracy of steady states implies that  $\mathcal{S}_{\mathbf{k}}$  is not discrete. Reciprocally, we have the following result.

**Theorem 4.3.** Assume that for  $\mathbf{k} \in \mathbb{R}_+^r$ ,  $\mathcal{S}_{\mathbf{k}}$  is a manifold. If the local dimension at a point  $\mathbf{x}_0 \in \mathcal{S}_{\mathbf{k}}$  is strictly positive, then  $\mathbf{x}_0$  is degenerate.

*Proof.* There is a neighborhood  $U \subset \mathbb{R}$  of zero and a smooth function  $U \rightarrow \mathcal{S}_{\mathbf{k}}$ ,  $\alpha \mapsto \mathbf{x}(\alpha)$  with  $\mathbf{x}(0) = \mathbf{x}_0$ . By differentiating  $\mathbf{F}(\mathbf{k}, \mathbf{x}(\alpha)) = 0$  with respect to  $\alpha$  at 0 we get  $D_{\mathbf{x}}\mathbf{F}(\mathbf{k}, \mathbf{x}_0) \frac{d\mathbf{x}(0)}{d\alpha} = 0$ , and thus  $D_{\mathbf{x}}\mathbf{F}(\mathbf{k}, \mathbf{x}_0)$  is singular. ■

**Definition 4.4.** A set of  $s$  exact conservation laws defined by the vector

$$\Phi(\mathbf{k}, \mathbf{x}) = (\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x}))^T$$

is called complete if the Jacobian matrix

$$\mathbf{J}_{\mathbf{F}, \Phi}(\mathbf{k}, \mathbf{x}) = D_{\mathbf{x}}(\mathbf{F}(\mathbf{k}, \mathbf{x}), \Phi(\mathbf{k}, \mathbf{x}))^T$$

has rank  $n$  for any  $\mathbf{k} \in \mathbb{R}_+^r, \mathbf{x} \in \mathbb{R}_+^n$ . The set is called independent if the Jacobian matrix of  $\Phi(\mathbf{k}, \mathbf{x})^T$  with respect to  $\mathbf{x}$  has rank  $s$  for any  $\mathbf{k} \in \mathbb{R}_+^r, \mathbf{x} \in \mathbb{R}_+^n$ . In the case of a set of approximate conservation laws defined by a vector  $\Phi(\mathbf{k}, \mathbf{x})$ , completeness is defined with  $\mathbf{F}(\mathbf{k}, \mathbf{x})$  replaced by  $\mathbf{F}^{(1)}(\mathbf{k}, \mathbf{x})$ .

**Proposition 4.5.** If a system has a complete set of conservation laws, then the intersection of  $\mathcal{S}_{\mathbf{k}}$  with  $\{\mathbf{x} \mid \Phi(\mathbf{k}, \mathbf{x}) = \mathbf{c}_0\} \cap \mathbb{R}_+^n$ , where  $\mathbf{c}_0 \in \mathbb{R}^s$ , is finite.

*Proof.* Suppose that the intersection is nonempty and contains  $\mathbf{x}$ . Because the rank of  $\mathbf{J}_{\mathbf{F}, \Phi}(\mathbf{k}, \mathbf{x})$  in  $\mathbf{x}$  is  $n$ , the implicit function theorem implies that, for  $\mathbf{k}$  and  $\mathbf{c}_0$ ,  $\mathbf{x}$  is isolated from other solutions of  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = 0$ ,  $\Phi(\mathbf{k}, \mathbf{x}) = \mathbf{c}_0$ . As  $\mathbf{F}(\mathbf{k}, \mathbf{x})$  is polynomial, the intersection is finite. ■

**Remark 4.6.** When  $\Phi(\mathbf{k}, \mathbf{x})$  is linear in  $\mathbf{x}$  and results from a stoichiometric matrix, the set

$$\mathcal{C}_{\mathbf{c}_0} = \{\mathbf{x} \mid \Phi(\mathbf{k}, \mathbf{x}) = \mathbf{c}_0\} \cap \mathbb{R}_+^n$$

is called the *stoichiometric compatibility class* or *reaction simplex* [27, 72]. Stoichiometric compatibility classes of systems with complete sets of linear conservation laws contain a finite

number of steady states. We note that some authors call completeness of linear conservation laws *nondegeneracy* [28]. Here, we extend these definitions and call *compatibility classes* all sets of the type  $\mathcal{C}_{\mathbf{c}_0}$ , even if  $\Phi(\mathbf{k}, \mathbf{x})$  is a nonlinear conservation law. Compatibility classes of polynomial conservation laws are semialgebraic sets.

Compatibility classes and the completeness of conservation laws have applications to the study of multistationarity of models [28] and to model order reduction [21] (see also section 3 and Remark 3.2).

**Remark 4.7.** Since our concern is the number of positive solutions in  $\mathbf{x}$  of  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = 0$ ,  $\Phi(\mathbf{k}, \mathbf{x}) = \mathbf{c}_0$ , it would be more natural to consider in Definition 4.4 the rank of  $\mathbf{J}_{\mathbf{F}, \phi}(\mathbf{k}, \mathbf{x})$  on

$$\mathcal{S}_{\mathbf{k}} \cap \{\mathbf{x} \mid \Phi(\mathbf{k}, \mathbf{x}) = \mathbf{c}_0\} \cap \mathbb{R}_+^n.$$

In fact, as this rank does not depend on  $\mathbf{c}_0$ , it is simpler and equivalent to impose its value on  $\mathcal{S}_{\mathbf{k}} \cap \mathbb{R}_+^n$ .

**Remark 4.8.** The independent linear conservation laws

$$\Phi(\mathbf{k}, \mathbf{x}) = (x_1 + x_2, x_2 + x_3)$$

of the truncated system in Example 3.1 are complete. More precisely, the Jacobian of

$$(\mathbf{F}(\mathbf{k}, \mathbf{x}), \Phi(\mathbf{k}, \mathbf{x}))^\top,$$

where  $\mathbf{F}(\mathbf{k}, \mathbf{x})$  is the vector of the r.h.s. of (3.2), has a  $3 \times 3$  minor

$$\det(D_{\mathbf{x}}(-k_1x_1x_3 + k_2x_2, \Phi(\mathbf{k}, \mathbf{x}))^\top) = -k_2 - k_1x_1 - k_1x_3.$$

This minor cannot be zero for positive  $\mathbf{x}$ ,  $\mathbf{k}$  on the steady state variety which is defined by the single equation  $-k_1x_1x_3 + k_2x_2 = 0$ . Therefore the rank of  $\mathbf{J}_{\mathbf{F}, \phi}(\mathbf{x}, \mathbf{k})$  is 3.

Furthermore, all stoichiometric compatibility classes defined by  $x_1 + x_2 = c_{01}$ ,  $x_2 + x_3 = c_{02}$ ,  $\mathbf{x} > 0$ , contain a unique positive steady state

$$\begin{aligned} x_1 &= (k_1(c_{01} - c_{02}) - k_2 + \sqrt{\Delta})/(2k_1), \\ x_2 &= (k_1(c_{01} + c_{02}) + k_2 - \sqrt{\Delta})/(2k_1), \\ x_3 &= (k_1(c_{02} - c_{01}) - k_2 + \sqrt{\Delta})/(2k_1), \end{aligned}$$

where  $\Delta = (c_{01} - c_{02})^2k_1^2 + k_2^2 + 2k_1k_2(c_{01} + c_{02})$ . This nondegenerate steady state is precisely the one used for the quasi-steady reduction of the transformed model (see also Remark 3.2).

The notions of completeness and independence in Definition 4.4 are effective. Indeed, Algorithms 4.2 and 4.3 below check for completeness and independence, respectively. Both algorithms use a parametric rank computation (see in both cases line 1 with the call “ParametricRank”), which we are going to present first.

Let  $L, N, M \in \mathbb{N}$  and let  $\mathbb{R}[\mathbf{v}]$  be a polynomial ring in  $P$  indeterminates  $\mathbf{v} = (v_1, \dots, v_L)$ . Furthermore, let  $A \in \mathbb{R}[\mathbf{v}]^{M \times N}$  be a matrix whose entries are polynomials in  $\mathbf{v}$ . We are interested in the rank of  $A$  in dependence of different values  $\bar{\mathbf{v}} \in \mathbb{R}^L$  for the variables  $\mathbf{v}$ .

Algorithm 4.1 (ParametricRank) provides a decomposition of the affine space  $\mathbb{R}^L$  into disjoint semialgebraic subsets

$$S_1, \dots, S_I$$

such that for each subset  $S_i$  among the  $S_1, \dots, S_I$  the rank of  $A(\bar{v})$  is constant for all points  $\bar{v}$  of  $S_i$ . The output of the algorithm is organized as a set of pairs

$$R = \{(\Gamma_1, r_1), \dots, (\Gamma_I, r_I)\}.$$

For each  $i = 1, \dots, I$ ,  $\Gamma_i$  is a conjunction of polynomial constraints in  $\mathbf{v}$ , i.e., polynomial equations and inequations in  $\mathbf{v}$  with coefficients in  $\mathbb{R}$ , and  $r_i$  is an integer. The semialgebraic sets  $S_1, \dots, S_I$  from above are defined by  $\Gamma_1, \dots, \Gamma_I$  and  $r_1, \dots, r_I$  are the ranks of  $A$  on these sets.

In order to determine the different ranks, the algorithm computes row echelon forms by introducing case distinctions. Roughly speaking the algorithm works in the following way. For a semialgebraic set  $S$  defined by a conjunction of polynomial constraints  $\Gamma$  and for a matrix  $A$  which is in row echelon form up to the  $p$ th row, the algorithm checks if there is an entry  $A_{ij}$  in  $A$ , where  $p+1 \leq i \leq M$  and  $p+1 \leq j \leq N$ , which does not vanish for any point of  $S$ . If  $A_{mn}$  is such an entry, then one swaps rows  $p+1$  with  $m$  and columns  $p+1$  with  $n$ . One then uses the entry  $A_{p+1,p+1}$  in the new matrix as a pivot element to delete all subsequent entries and introduces a new case consisting of the same set  $S$ , i.e., the same  $\Gamma$ , but now with the matrix in row echelon form up to row  $p+1$ . If there is not such an entry, the algorithm checks if there is an entry  $A_{ij}$ , where  $i, j$  runs as above, which does not vanish for all points of  $S$ . If  $A_{mn}$  is such an entry, the algorithm introduces two new cases. Both cases consist of the old matrix  $A$  in row echelon form up to row  $p$ . The difference is now that one case consists of the semialgebraic set defined by  $\Gamma \wedge A_{mn} \neq 0$  and the other one consists of the set defined by  $\Gamma \wedge A_{mn} = 0$ . The new conditions guarantee the nonvanishing and the vanishing of the entry  $A_{mn}$  in the respective case. If there is not an entry  $A_{mn}$  as described above, on  $S$  the matrix  $A$  is in row echelon form and has rank  $p$ . The algorithm returns  $\Gamma$  and  $p$ . The algorithm processes a stack of cases in the way described above starting with the case which consists of the semialgebraic set  $\mathbb{R}^L$  defined by  $\Gamma = \text{"true"}$  and the matrix  $A$  in row echelon form up to row  $p=0$ .

Technically, the algorithm uses a logical deduction procedure  $\vdash$  to heuristically derive from  $\Gamma$  whether or not relevant entries vanish in  $\mathbb{R}$ . The correctness of the algorithm requires only two very natural assumptions on  $\vdash$ :

- (i) If  $\Gamma \vdash \gamma$ , then  $\Gamma$  entails  $\gamma$  in  $\mathbb{R}$ , i.e.,  $\vdash$  is sound.
- (ii)  $\gamma \wedge \Gamma \vdash \gamma$ , i.e.,  $\vdash$  can derive constraints  $\gamma$  that literally occur on the l.h.s.

Of course, our notation in (ii) should be read modulo associativity and commutativity of the logical conjunction operator. Notice that (ii) is easy to implement, and implementing only (ii) is certainly sound.

*Example 4.9.* We consider the  $3 \times 3$ -matrix

$$A^{(0)} = \begin{pmatrix} 0 & v_1 & 1 \\ v_1 & v_3^2 - 1 + v_1 v_2 & v_3 \\ v_3 - 1 & v_3 + 1 + v_1 v_3^3 & v_3^3 \end{pmatrix}$$

with polynomial entries in  $\mathbb{R}[\mathbf{v}]$  with  $\mathbf{v} = (v_1, v_2, v_3)$ . Starting with  $(\text{true}, A^{(0)}, 0)$  we find in  $A^{(0)}$  the entry  $A_{1,3}^{(0)} = 1$  which clearly does not vanish on  $\mathbb{R}^3$ . We swap the first and third columns and use 1 as a pivot to delete the entries  $v_3$  and  $v_3^3$ . We obtain

$$A^{(1)} = \begin{pmatrix} 1 & v_1 & 0 \\ 0 & v_3^2 - 1 & v_1 \\ 0 & v_3 + 1 & v_3 - 1 \end{pmatrix}$$

and introduce  $(\text{true}, A^{(1)}, 1)$ . For each entry  $A_{i,j}^{(1)}$  with  $2 \leq i, j \leq 3$  there is at least one point in  $\mathbb{R}^3$  such that  $A_{i,j}^{(1)}$  vanishes. On the other hand, for each entry  $A_{i,j}^{(1)}$  with  $2 \leq i, j \leq 3$  there is also at least one point in  $\mathbb{R}^3$  such that  $A_{i,j}^{(1)}$  does not vanish. We choose  $A_{2,2}^{(1)} = v_3^2 - 1$  and introduce the cases  $(v_3^2 - 1 \neq 0, A^{(1)}, 1)$  and  $(v_3^2 - 1 = 0, A^{(1)}, 1)$ . Continuing with the first case, we trivially find that the entry  $A_{2,2}^{(1)} = v_3^2 - 1$  does not vanish on the semialgebraic set  $\{\bar{\mathbf{v}} \in \mathbb{R}^3 \mid \bar{v}_3^2 - 1 \neq 0\}$  and we use it as a pivot to delete the entry  $v_3 + 1$ . We obtain

$$A^{(2)} = \begin{pmatrix} 1 & v_1 & 0 \\ 0 & v_3^2 - 1 & v_1 \\ 0 & 0 & (v_3 - 1)^2 - v_1 \end{pmatrix}$$

and introduce  $(v_3^2 - 1 \neq 0, A^{(2)}, 2)$ . Since the set  $\{\bar{\mathbf{v}} \in \mathbb{R}^3 \mid \bar{v}_3^2 - 1 \neq 0\}$  contains points where  $A_{3,3}^{(2)}$  vanishes and points where  $A_{3,3}^{(2)}$  does not vanish, we introduce the two cases

$$\begin{aligned} & (v_3^2 - 1 \neq 0 \wedge (v_3 - 1)^2 \neq v_1, A^{(2)}, 2), \\ & (v_3^2 - 1 \neq 0 \wedge (v_3 - 1)^2 = v_1, A^{(2)}, 2). \end{aligned}$$

In the first case, the entry  $A_{3,3}^{(2)}$  clearly does not vanish on the semialgebraic set

$$\{\bar{\mathbf{v}} \in \mathbb{R}^3 \mid \bar{v}_3^2 - 1 \neq 0 \wedge (\bar{v}_3 - 1)^2 \neq \bar{v}_1\}.$$

Since there is no entry left to delete with the pivot  $A_{3,3}^{(2)}$ , we simply introduce the new case

$$(v_3^2 - 1 \neq 0 \wedge (v_3 - 1)^2 \neq v_1, A^{(2)}, 3),$$

which yields the output  $(v_3^2 - 1 \neq 0 \wedge (v_3 - 1)^2 \neq v_1, 3)$ . In the second case we have that the entry  $A_{3,3}^{(2)}$  is zero on the constructed semialgebraic set and so the algorithm returns  $(v_3^2 - 1 \neq 0 \wedge (v_3 - 1)^2 = v_1, 2)$ . Finally, we are coming back to the case  $(v_3^2 - 1 = 0, A^{(1)}, 1)$ . On the semialgebraic set  $\{\bar{\mathbf{v}} \in \mathbb{R}^3 \mid \bar{v}_3^2 - 1 = 0\}$  the entries  $\bar{v}_3^2 - 1$ ,  $\bar{v}_3 + 1$ , and  $\bar{v}_3 - 1$  are zero while there are points in this set where the entry  $A_{2,3}^{(1)} = v_1$  vanishes and points where it does not vanish. We introduce the cases  $(v_3^2 - 1 = 0 \wedge v_1 \neq 0, A^{(1)}, 1)$  and  $(v_3^2 - 1 = 0 \wedge v_1 = 0, A^{(1)}, 1)$ . In the first case the entry  $A_{2,3}^{(1)} = v_1$  is nonzero on the semialgebraic set

$$\{\bar{\mathbf{v}} \in \mathbb{R}^3 \mid \bar{v}_3^2 - 1 = 0 \wedge \bar{v}_1 \neq 0\}$$

and so we obtain  $(v_3^2 - 1 = 0 \wedge v_1 \neq 0, A^{(1)}, 2)$ , which leads to the output  $(v_3^2 - 1 = 0 \wedge v_1 \neq 0, 2)$ . In the second case all entries  $A_{i,j}^{(1)}$  with  $2 \leq i, j \leq 3$  are zero on the corresponding semialgebraic set and therefore the algorithm returns  $(v_3^2 - 1 = 0 \wedge v_1 = 0, 1)$ .

**Algorithm 4.1.** ParametricRank.**Input:** An  $M \times N$ -matrix  $A(\mathbf{v})$  with polynomial entries in parameters  $\mathbf{v}$ .**Output:** A set of pairs  $\{(\Gamma_1(\mathbf{v}), r_1), \dots, (\Gamma_I(\mathbf{v}), r_I)\}$ , where each  $\Gamma_i(\mathbf{v})$  is a conjunction of polynomial equations and inequations and  $r_i \in \{1, \dots, N\}$ . For any real choice  $\bar{\mathbf{v}}$  of parameters  $\mathbf{v}$  there is one and only one  $i \in \{1, \dots, I\}$  such that  $\Gamma_i(\bar{\mathbf{v}})$  holds in  $\mathbb{R}$ . For this  $i$  we have  $\text{rank } A(\bar{\mathbf{v}}) = r_i$ .

---

```

1:  $I := 0$ 
2: create an empty stack
3: push (true,  $A$ , 0)
4: while stack is not empty do
5:    $(\Gamma, A, p) := \text{pop}$ 
6:   if  $\Gamma \not\equiv \text{false}$  then
7:     if there is  $m \in \{p+1, \dots, M\}$ ,  $n \in \{p+1, \dots, N\}$  s.t.  $\Gamma \vdash A_{mn} \neq 0$  then
8:       in  $A$ , swap rows  $p+1$  with  $m$  and columns  $p+1$  with  $n$ 
9:       in  $A$ , use row  $p+1$  to obtain  $A_{p+2,p+1} = \dots = A_{M,p+1} = 0$ 
10:      push  $(\Gamma, A, p+1)$ 
11:     else if there is  $m \in \{p+1, \dots, M\}$ ,  $n \in \{p+1, \dots, N\}$  s.t.  $\Gamma \not\vdash A_{mn} = 0$  then
12:        $a := A_{mn}$ 
13:       push  $(\Gamma \wedge a \neq 0, A, p)$ 
14:       in  $A$ , set  $A_{mn} := 0$  optional optimization
15:       push  $(\Gamma \wedge a = 0, A, p)$ 
16:     else A is in row echelon form modulo  $\Gamma$ 
17:        $I := I + 1$ 
18:        $(\Gamma_I, r_I) := (\Gamma, p)$ 
19:     end if
20:   end if
21: end while
22: return  $\{(\Gamma_1, r_1), \dots, (\Gamma_I, r_I)\}$ 

```

---

*Remark 4.10.* Similar approaches [62] do not necessarily yield disjoint parameter ranges  $S_i$  (see [6, section 5.3]).

Let us get back to checking for completeness and independence. We start with Algorithm 4.2, which handles completeness.

On the grounds of Algorithm 4.1, we construct in lines 2–3 in Algorithm 4.2 an equivalent logic condition  $\varrho_n$  for  $\text{rank } \mathbf{J}_{\mathbf{F},\phi}(\mathbf{k}, \mathbf{x}) = n$ . In line 4 we construct  $\gamma$  as a direct formalization of the definition of completeness and in line 5 we finally test validity of  $\gamma$  over the reals. Technically we use a combination of various effective quantifier elimination procedures for the theory of real closed fields (see [24, 67, 68] and the references there) combined with heuristic simplification techniques [23] in the Redlog system [22, 42, 60]. Alternatively, one could use satisfiability modulo theories solving over the logic QF\_NRA [1, 7, 54]. Algorithm 4.3 proceeds

---

**Algorithm 4.2.** IsComplete.

---

**Input:** 1.  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))$ ; 2.  $\Phi(\mathbf{k}, \mathbf{x}) = (\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x}))$ ;  
3.  $\mathbf{k} = (k_1, \dots, k_r)$ ; 4.  $\mathbf{x} = (x_1, \dots, x_n)$

**Output:** “yes” if  $\Phi$  is complete according to Definition 4.4, “no” otherwise

```

1:  $R := \text{ParametricRank}(\mathbf{J}_{\mathbf{F}, \phi}(\mathbf{k}, \mathbf{x}))$ 
2:  $R_n := \{(\Gamma, r) \in R \mid r = n\}$ 
3:  $\varrho_n := \bigvee_{(\Gamma, n) \in R_n} \Gamma$ 
4:  $\gamma := \forall \mathbf{k} \forall \mathbf{x} (\mathbf{k} > 0 \wedge \mathbf{x} > 0 \wedge \mathbf{F}(\mathbf{k}, \mathbf{x}) = 0 \longrightarrow \varrho_n)$ 
5: if  $\mathbb{R} \models \gamma$  then
6:   return “yes”
7: else
8:   return “no”
9: end if

```

---



---

**Algorithm 4.3.** IsIndependent.

---

**Input:** 1.  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))$ ; 2.  $\Phi(\mathbf{k}, \mathbf{x}) = (\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x}))$ ;  
3.  $\mathbf{k} = (k_1, \dots, k_r)$ ; 4.  $\mathbf{x} = (x_1, \dots, x_n)$

**Output:** “yes” if  $\Phi$  is independent according to Definition 4.4, “no” otherwise

```

1:  $R := \text{ParametricRank}(D_{\mathbf{x}}\Phi(\mathbf{k}, \mathbf{x})^T)$ 
2:  $R_s := \{(\Gamma, r) \in R \mid r = s\}$ 
3:  $\varrho_s := \bigvee_{(\Gamma, s) \in R_s} \Gamma$ 
4:  $\iota := \forall \mathbf{k} \forall \mathbf{x} (\mathbf{k} > 0 \wedge \mathbf{x} > 0 \wedge \mathbf{F}(\mathbf{k}, \mathbf{x}) = 0 \longrightarrow \varrho_s)$ 
5: if  $\mathbb{R} \models \iota$  then
6:   return “yes”
7: else
8:   return “no”
9: end if

```

---

analogously to Algorithm 4.2 and tests for independence.

*Example 4.11.* We automatically process Example 3.1 with Algorithm 4.2, IsComplete. Our input is

$$\begin{aligned}\mathbf{F}(\mathbf{k}, \mathbf{x}) &= (-k_1x_1x_3 + k_2x_2, k_1x_1x_3 - k_2x_2, -k_1x_1x_3 + k_2x_2), \\ \Phi(\mathbf{k}, \mathbf{x}) &= (x_1 + x_2, x_2 + x_3)\end{aligned}$$

and  $\mathbf{k} = (k_1, k_2)$ ,  $\mathbf{x} = (x_1, x_2, x_3)$ . We obtain the parametric Jacobian

$$\mathbf{J}_{\mathbf{F}, \phi}(\mathbf{k}, \mathbf{x}) = \begin{pmatrix} -k_1x_3 & k_2 & -k_1x_1 \\ k_1x_3 & -k_2 & k_1x_1 \\ -k_1x_3 & k_2 & -k_2x_1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

and compute in line 1 the parametric rank

$$R = \{(k_1x_1 + k_1x_3 + k_2 \neq 0, 3), (k_1x_1 + k_1x_3 + k_2 = 0, 2)\},$$

from which we select  $R_3 = \{(k_1x_1 + k_1x_3 + k_2 \neq 0, 3)\}$  in line 2 and

$$\varrho_3 = (k_1x_1 + k_1x_3 + k_2 \neq 0)$$

in line 3. Completeness is straightforwardly formalized in line 4 by

$$\gamma = \forall \mathbf{k} \forall \mathbf{x} (\mathbf{k} > 0 \wedge \mathbf{x} > 0 \wedge k_1x_1x_3 - k_2x_2 = 0 \longrightarrow k_1x_1 + k_1x_3 + k_2 \neq 0),$$

where some redundant equations are automatically removed from  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = 0$  via the standard simplifier described in [23]. Real quantifier elimination on  $\gamma$  in line 5 equivalently yields “true,” which confirms completeness, and we return “yes” in line 6.

Similarly, Algorithm 4.3, *IsIndependent*, computes

$$D_{\mathbf{x}}\Phi(\mathbf{k}, \mathbf{x})^{\top} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

$R = \{(\text{true}, 2)\}$ ,  $R_2 = R$ , and  $\varrho_2 = \text{true}$ . This yields

$$\iota = \forall \mathbf{k} \forall \mathbf{x} (\mathbf{k} > 0 \wedge \mathbf{x} > 0 \wedge k_1x_1x_3 - k_2x_2 = 0 \longrightarrow \text{true}),$$

which by quantifier elimination is equivalent to “true,” and we return “yes.” Recall that any implication with “true” on the r.h.s. holds already due to Boolean logic [61]. In our framework here, this corresponds to the observation that completeness and independence hold whenever we encounter full rank of the corresponding Jacobian for all choices of parameters. The automatic computations described here take less than 0.01 sec altogether.

The next example shows that there are models with degenerate steady states and no complete set of conservation laws.

*Example 4.12.* One checks easily that the system

$$\dot{x}_1 = 1 - x_1 - x_2, \quad \dot{x}_2 = x_1 + x_2 - 1$$

has the linear conservation law  $\Phi(\mathbf{k}, \mathbf{x}) = x_1 + x_2$  and that the Jacobian of  $(\mathbf{F}(\mathbf{k}, \mathbf{x}), \Phi(\mathbf{k}, \mathbf{x}))^{\top}$  is constant and has rank 1. Trivially the Jacobian has everywhere rank 1 and so  $\Phi(\mathbf{k}, \mathbf{x})$  is not complete. The explicit solutions of the ODE system are

$$\begin{aligned} x_1(t) &= (1 - c_0)t + c_1, \\ x_2(t) &= (c_0 - 1)t + c_0 - c_1. \end{aligned}$$

Thus, all invariant curves are of the form  $x_1 + x_2 = c_0$ . We conclude that there are no further first integrals and so the system has no complete set of conservation laws. We can also note that the intersection of the steady state variety defined by  $x_1 + x_2 = 1$  with a stoichiometric compatibility class  $\{x_1 + x_2 = c_0, x_1 > 0, x_2 > 0\}$  is either empty or a line segment.



In the absence of the parameters  $\mathbf{k}$  Algorithm 4.2, IsComplete, computes the parametric rank  $R = \{(\text{true}, 1)\}$ . However, we have  $n = 2$ , which yields  $R_2 = \emptyset$  and  $\varrho_2 = \text{false}$ . This gives us

$$\gamma = \forall \mathbf{k} \forall \mathbf{x} (\text{true} \wedge \mathbf{x} > 0 \wedge x_1 + x_2 - 1 = 0 \longrightarrow \text{false}),$$

which quantifier elimination identifies to be “false,” and the algorithm returns “no.” Notice that  $\gamma$  is equivalent to

$$\forall \mathbf{x} (\mathbf{x} > 0 \longrightarrow x_1 + x_2 - 1 \neq 0),$$

which illustrates that with deficient rank, completeness can hold formally only when there is no steady state in the positive orthant.

**5. Computing linear conservation laws.** Throughout this section we consider that the polynomial ODE system (2.1) has the structure given by (2.2). If the model is a CRN this is automatically true; otherwise we can always bring the system to this form by using Algorithms 2.1 and 2.2.

By definition a vector  $\mathbf{c} = (c_1, \dots, c_n) \in \mathbb{R}^n$  defines an exact linear conservation law

$$\Phi(\mathbf{k}, \mathbf{x}) = \sum_{i=1}^n c_i x_i$$

of the system in (2.1) with velocity field satisfying (2.2) if

$$\sum_{i=1}^n c_i f_i(\mathbf{k}, \mathbf{x}) = \sum_{j=1}^r (\mathbf{cS})_j k_j \mathbf{x}^{\alpha_j} = 0$$

for all  $\mathbf{k}, \mathbf{x}$ . Exact linear conservation laws of the system in (2.1) can be obtained from the matrix  $\mathbf{S}$ , as the following theorem shows.

**Theorem 5.1.** *Assume that all monomial reaction rates with the same multi-index have different rate constants, i.e.,  $\alpha_j = \alpha_{j'}$  with  $j \neq j'$  implies  $k_j \neq k_{j'}$ . This condition is always fulfilled in our setting (see the Remark 2.2). Let  $r_S$  be the rank of the matrix  $\mathbf{S}$ . Then there is a full rank matrix  $\mathbf{C}$  with  $n - r_S$  rows such that  $\mathbf{CS} = 0$ . Furthermore,  $\mathbf{C}$  can be chosen such that its rows form a set of independent, simple, exact linear conservation laws unconditional on  $\mathbf{k}$ . Finally, all  $\mathbf{c} = (c_1, \dots, c_n)$  defining unconditional, exact linear conservation laws are linear combinations of the rows of  $\mathbf{C}$ .*

**Proof.** As a direct consequence of the rank-nullity theorem, there is a matrix  $\mathbf{C}$  with  $n - r_S$  independent rows such that  $\mathbf{CS} = 0$ . If  $\mathbf{c} = (c_1, \dots, c_n)$  defines an exact, unconditional linear conservation law, we then have that

$$\sum_{j=1}^r (\mathbf{cS})_j k_j \mathbf{x}^{\alpha_j} = 0$$

for all  $\mathbf{k}, \mathbf{x}$ . If  $i \neq j$  one has  $\alpha_i \neq \alpha_j$  or  $k_i \neq k_j$ . It follows that  $k_j \mathbf{x}^{\alpha_j}$  are linearly independent in  $\mathbb{R}[\mathbf{k}, \mathbf{x}]$  and  $\mathbf{cS} = 0$ . Therefore,  $\mathbf{c}$  is a linear combination of rows of  $\mathbf{C}$ . Those rows of  $\mathbf{C}$  that

are not simple can be decomposed as a sum of simple linear conservation laws. The resulting conservation laws form a generating set for the vector space of linear conservation laws and therefore this set contains a basis. The  $n - r_S$  elements of this basis can be chosen as the simple rows of  $\mathbf{C}$ . Here, the unconditionality of  $\mathbf{c}$  results from the fact that  $\mathbf{S}$  and therefore  $\mathbf{c}$  do not depend on  $\mathbf{k}$ . ■

**Remark 5.2.** Since  $S_{ij} \in \mathbb{Z}$ , the coefficients  $c_i$  can be chosen to be integers. For conservative CRNs linear conservation laws can be chosen semipositive, that is, all  $c_i \geq 0$  (see [59]). Semipositive linear conservation laws are important, because their existence implies that concentrations of some species are bounded at all times. Also, semipositive conservation laws allow easy interpretation in terms of pools of chemical species. An algorithm to compute semipositive linear conservation laws can be found in [59]. We must emphasize that it is not always possible to transform complete systems of linear conservation laws into semipositive linear conservation laws. As an example, one may consider the formation of a heterodimer described by  $\dot{x}_1 = -k_1 x_1 x_2 + k_2 x_3$ ,  $\dot{x}_2 = -k_1 x_1 x_2 + k_2 x_3$ ,  $\dot{x}_3 = k_1 x_1 x_2 - k_2 x_3$ . This system has the complete system of linear conservation laws  $\Phi = (x_1 - x_2, x_1 + x_2 - 2x_3)$  that cannot be transformed into a semipositive system. Of course, semipositiveness is not relevant for nonchemical application, except for population dynamics models used in ecology.

**Remark 5.3.** In Example 4.12 we have seen that  $n - r_S$  independent linear conservation laws may not form a complete system. The system in this example satisfies (2.2), although it is not a mass action network.

**Theorem 5.4.** Consider the Jacobian matrix  $\mathbf{J}(\mathbf{k}, \mathbf{x}) = D_{\mathbf{x}} \mathbf{F}(\mathbf{k}, \mathbf{x})$  and the matrix  $\mathbf{C}$  introduced in Theorem 5.1. The rows of  $\mathbf{C}$  provide a complete set of linear conservation laws if for all  $\mathbf{k} \in \mathbb{R}_+^r$ ,  $\mathbf{x} \in \mathbb{R}_+^n$  with  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = \mathbf{0}$  the following conditions hold:

- (i)  $rk(\mathbf{J}(\mathbf{k}, \mathbf{x})) = rk(\mathbf{S}) = r_S$ ,
- (ii) no row of the product  $\mathbf{J}(\mathbf{k}, \mathbf{x})\mathbf{S}$  is zero.

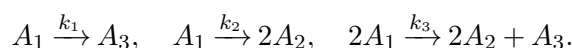
**Proof.** We check that the assumptions imply the conditions in Definition 4.4. From (i) it follows that  $\mathbf{J}(\mathbf{k}, \mathbf{x})$  has  $r_S$  independent rows. Moreover, from (ii) we obtain that these rows are not in the left kernel of  $\mathbf{S}$ , which is spanned by the  $n - r_S$  rows of  $\mathbf{C}$ . Thus the matrix  $(\mathbf{J}(\mathbf{k}, \mathbf{x}), \mathbf{C})^T$  has rank  $n$ . ■

The CRN in Example 4.12 has no complete set of linear conservation laws. This model fails to satisfy condition (ii) of the theorem, but it fulfills (i). The network is not a mass action CRN, so one may ask whether mass action CRNs automatically satisfy both (i) and (ii). The answer to this question is no, as shown by the following example.

**Example 5.5.** The CRN

$$\dot{x}_1 = -k_1 x_1 - k_2 x_1 - 2k_3 x_1^2, \quad \dot{x}_2 = 2k_3 x_1^2 + 2k_2 x_1, \quad \dot{x}_3 = k_3 x_1^2 + k_1 x_1$$

is a mass action network described by the reactions



Its stoichiometric matrix is

$$\mathbf{S} = \begin{pmatrix} -1 & -1 & -2 \\ 0 & 2 & 2 \\ 1 & 0 & 1 \end{pmatrix}$$

and one easily checks that it has rank two. There is a single linear conservation law, namely  $\phi(\mathbf{k}, \mathbf{x}) = 2x_1 + x_2 + 2x_3$ . The Jacobian matrix of the system is

$$J(\mathbf{k}, \mathbf{x}) = \begin{pmatrix} -k_1 - k_2 - 4k_3x_1 & 0 & 0 \\ 2k_2 + 4k_3x_1 & 0 & 0 \\ k_1 + 2k_3x_1 & 0 & 0 \end{pmatrix},$$

which clearly has rank one. Therefore, this model does not satisfy condition (i) of Theorem 5.4, whereas (ii) is fulfilled. Furthermore, the equations  $\mathbf{F}(\mathbf{k}, \mathbf{x}) = 0$ ,  $\phi(\mathbf{k}, \mathbf{x}) = c_0$ , which define the intersection of  $\mathcal{S}_{\mathbf{k}}$  with the stoichiometric compatibility class, have degenerate solutions  $x_1 = 0$ ,  $x_2 = c_0 - 2x_3$  and so the conservation law is not complete.

In [28], sufficient conditions were found for mass action CRNs to be “injective”, meaning to have a unique steady state in any stoichiometric compatibility class (see Theorems 5.6 and 9.1 of [28]). The same conditions also imply completeness of any  $n - r_{\mathbf{S}}$  independent, linear conservation laws.

**6. Computing nonlinear conservation laws.** Nonlinear conservation laws, i.e., first integrals, of polynomial systems of ODEs have been approached by means of a variety of methods. For example, simple conservation laws, i.e., expressions composed of polynomials, logarithms, and exponentials, can be obtained by *Darboux polynomials* [16, 55]. The bottleneck of this method is the computation of all Darboux polynomials of the ODE system. This problem has been approached in computational algebra by using Gröbner bases [50]. For particular models, for instance, Lotka–Volterra models, Darboux polynomials and conservation laws of a specific kind can be obtained with the methods described in [49].

In this section we restrict ourselves to polynomial conservation laws. We show that polynomial conservation laws are obtained from syzygies, which are easier to compute than Darboux polynomials. Our methods are more general and apply to models not covered in [49]. However, we do not cope with parametric exponential conservation laws existing in Lotka–Volterra models.

We also discuss monomial and rational monomial conservation laws that can be effectively computed without syzygies or Darboux polynomials. We show that the computation of monomial conservation laws can be reduced to the computation of linear conservation laws. Finally we use the databases *BioModels* [15] and *ODEbase* [47] to study the scalability of our methods.

**6.1. Computing monomial conservation laws.** The following proposition establishes a link between linear and monomial conservation laws (see Theorem 1 of [32] for a similar result).

**Proposition 6.1.** *Let  $E$  be a system of ODEs given by*

$$\dot{x}_1 = f_1(\mathbf{k}, \mathbf{x}), \dots, \dot{x}_n = f_n(\mathbf{k}, \mathbf{x})$$

and let  $E'$  be the system of ODEs given by

$$\dot{x}_1 = \frac{f_1(\mathbf{k}, \mathbf{x})}{x_1}, \dots, \dot{x}_n = \frac{f_n(\mathbf{k}, \mathbf{x})}{x_n}.$$

Then  $E$  admits a monomial conservation law if and only if  $E'$  admits a linear conservation law with integer coefficients. Moreover, if the linear conservation law for  $E'$  is  $m_1x_1 + \dots + m_nx_n$ , then the monomial conservation law for  $E$  is  $x_1^{m_1} \dots x_n^{m_n}$ . The linear conservation laws of  $E'$  are simple if and only if the corresponding monomial conservation laws of  $E$  are simple.

*Proof.* Let  $M = x_1^{m_1} \dots x_n^{m_n}$  be a monomial. Then the derivative of  $M$  is

$$\dot{M} = \left( \sum_{i=1}^n m_i \frac{\dot{x}_i}{x_i} \right) M.$$

The equivalences

$$\dot{M} = 0 \iff \sum_{i=1}^n m_i \frac{\dot{x}_i}{x_i} = 0 \iff \sum_{i=1}^n m_i \frac{f_i(\mathbf{k}, \mathbf{x})}{x_i} = 0$$

prove the first part of the proposition.

Let  $\phi = m_1x_1 + \dots + m_nx_n$  be the linear conservation law of  $E'$  corresponding to the monomial conservation law  $M$  of  $E$ . Assume that  $M$  is not simple, say,  $M = M_1M_2$  with  $M_k = x_1^{m_{k1}} \dots x_n^{m_{kn}}$  and  $\dot{M}_k = 0$  for  $k \in \{1, 2\}$ . Then by the first part

$$\dot{M}_k = 0 \iff \sum_{i=1}^n m_{ki} \frac{\dot{x}_i}{x_i} = 0$$

for  $k \in \{1, 2\}$ . Thus,  $\phi = \phi_1 + \phi_2$ , where  $\phi_k = m_{k1}x_1 + \dots + m_{kn}x_n$  are linear conservation laws of  $E'$  for  $k \in \{1, 2\}$ . This means that the linear conservation for  $E'$  is not simple. The reverse follows from the same reasoning. ■

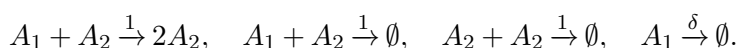
We consider that  $E'$  has the structure given by (2.2) with a stoichiometric matrix  $\mathbf{S}'$ . This is always possible, eventually after using the Algorithm 2.1.

**Remark 6.2.** Proposition 6.1 holds even when the r.h.s. of  $E'$  are sums of rational monomials, which can be the case when  $E$  is a system as in (2.1) and some of the polynomials  $f_i(\mathbf{k}, \mathbf{x})$  are not a multiple of  $x_i$ . In this case, a stoichiometric matrix  $\mathbf{S}'$  can still be associated to  $E'$  using Algorithm 2.1 and the linear conservation laws of  $E'$  can be computed from the matrix  $\mathbf{S}'$ . This procedure is summarized in Algorithm 6.1.

**Example 6.3.** The model

$$\begin{aligned} \dot{x}_1 &= x_1(x_2 - x_1) - \delta x_1, \\ \dot{x}_2 &= x_2(x_1 - x_2) \end{aligned}$$

is a mass action network and is described by the reactions



---

**Algorithm 6.1.** StoichiometricMonomialConservation.

---

**Input:** Polynomial ODE system  $E$  written as in (2.1).

**Output:** Integer coefficient matrix  $C$  whose rows define monomial conservation laws  $\mathbf{x}^{C_i}$  for  $E$ .

- 1: Compute  $E'$  by dividing each ODE in  $E$  by  $x_i$ .
  - 2:  $S' = \text{Smatrix}(E')$ .
  - 3: Find  $C$  such that  $CS' = 0$  and  $C$  has  $n - r_S$  simple, independent rows, where  $r_S = \text{rk}(S')$ .
- 

The truncated system

$$\begin{aligned}\dot{x}_1 &= x_1(x_2 - x_1), \\ \dot{x}_2 &= x_2(x_1 - x_2)\end{aligned}$$

has the steady state variety  $x_1 = x_2$  and the Jacobian matrix is singular on it. After truncation and elimination of the factors  $x_1, x_2$  we obtain the system

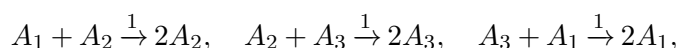
$$\begin{aligned}\dot{x}_1 &= x_2 - x_1, \\ \dot{x}_2 &= x_1 - x_2,\end{aligned}$$

which has the linear simple conservation law  $\phi_l(x_1, x_2) = x_1 + x_2$ . We conclude that the model has the monomial simple approximate conservation law  $\phi_m(x_1, x_2) = x_1 x_2$ . The monomial conservation law is complete, since the  $2 \times 2$  minor

$$\det(D_{\mathbf{x}}(x_1(x_2 - x_1), x_1 x_2)^T) = -2x_1^2$$

of the Jacobian matrix does not vanish for  $x_1 > 0$ . For  $c_0 \in \mathbb{R}_+$  the intersection of the steady state variety with the compatibility class defined by  $\phi_m(x_1, x_2) = c_0$  and  $x_1, x_2 > 0$  is the point  $x_1 = x_2 = \sqrt{c_0}$ .

*Example 6.4.* A model by Volpert (see [49]) is described by the mass action network



and the ODEs

$$\begin{aligned}\dot{x}_1 &= x_1(x_3 - x_2), \\ \dot{x}_2 &= x_2(x_1 - x_3), \\ \dot{x}_3 &= x_3(x_2 - x_1)\end{aligned}$$

have the steady state variety  $x_1 = x_2 = x_3$ . Moreover, the stoichiometric matrix

$$S = \begin{pmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

has rank  $r_S = 2$  and its left kernel is generated by  $\mathbf{c} = (1, 1, 1)$ . The stoichiometric matrix of the system obtained after elimination of the factors  $x_1, x_2, x_3$  is

$$S' = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

and has rank  $r_{S'} = 2$ . Moreover, its left kernel is also generated by  $\mathbf{c} = (1, 1, 1)$ . Thus the model has two approximate conservation laws, one linear  $\phi_l(\mathbf{x}) = x_1 + x_2 + x_3$  and one monomial  $\phi_m(\mathbf{x}) = x_1 x_2 x_3$ . Both conservation laws are complete. Indeed, the Jacobian  $D_{\mathbf{x}}(F_1, F_2, F_3, \phi_m)^T$  has a nonvanishing  $3 \times 3$  minor

$$\det(D_{\mathbf{x}}(x_1(x_3 - x_2), x_2(x_1 - x_3), x_1 x_2 x_3)^T) = x_1^2 x_2 x_3 + x_1 x_2^2 x_3 + x_1 x_2 x_3^2.$$

Moreover, the Jacobian  $D_{\mathbf{x}}(F_1, F_2, F_3, \phi_l)^T$  has a  $3 \times 3$  minor

$$\begin{aligned} M &= \det(D_{\mathbf{x}}(x_1(x_3 - x_2), x_2(x_1 - x_3), x_1 + x_2 + x_3)^T) \\ &= -(x_1 - x_2)^2 + 2x_1 x_3 + 2x_2 x_3 - x_3^2. \end{aligned}$$

On  $\mathcal{S}_{\mathbf{k}} \cap \mathbb{R}_+^3$  we have  $x_1 = x_2 = x_3$  and so  $M = 3x_1^2$  does not vanish there.

However, the two conservation laws are not independent on  $\mathcal{S}_{\mathbf{k}} \cap \mathbb{R}_+^3$ . Indeed, for  $x_1 = x_2 = x_3$  we have

$$D_{\mathbf{x}}(\phi_l, \phi_m)^T = \begin{pmatrix} 1 & 1 & 1 \\ x_1^2 & x_1^2 & x_1^2 \end{pmatrix}.$$

We want to point out that in this example we have used a special choice of the rate constants. Parameter dependent monomial conservation laws can be computed with methods presented in [32]. In particular, monomial conservation laws with parameter dependent exponents are reported in [49].

**6.2. Computing polynomial conservation laws from syzygies.** In this section, we are going to compute parametric polynomial conservation laws. To this end, we will use results and techniques from computational algebra, that is, from Gröbner bases theory. Our algorithm will be based on the computation of parametric syzygies for the polynomial r.h.s. in  $\mathbb{Q}[\mathbf{x}, \mathbf{k}]$  of (2.1). Since these r.h.s. depend on the parameters  $\mathbf{k}$ , we will compute a comprehensive Gröbner system (CGS) and compute for each branch the parametric syzygies. The CGS will give us a decomposition of the parameter space into semialgebraic sets such that evaluating the parameters  $\mathbf{k}$  at every point of such a semialgebraic set, the syzygies of the respective Gröbner basis will still be syzygies but with entries in  $\mathbb{Q}[\mathbf{x}]$ . Vector calculus will then tell us which syzygies are the gradients of polynomial functions in  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$  defining parametric conservation laws.

In the first subsection, we introduce the concept of syzygies for a polynomial system, and in the second subsection, we explain CGSs. The next subsection is devoted to establishing a connection between syzygies, conservative vector fields, and conservation laws. We also explain there why we need to consider the set of syzygies as a  $\mathbb{Q}$ -vector space instead of a  $\mathbb{Q}[\mathbf{x}, \mathbf{k}]$ -module. Having this connection, in the next subsection, we develop our algorithm. Finally, in the last subsection, we present the results and discuss our benchmark.

**6.2.1. The  $\mathbb{Q}$ -vector space of syzygies.** In this section  $\mathbb{Q}[\mathbf{x}]$  denotes a polynomial ring in the indeterminates  $\mathbf{x} = (x_1, \dots, x_n)$  over the field of rational numbers  $\mathbb{Q}$ . For  $m$  polynomials

$$F = (f_1, \dots, f_m)$$

of  $\mathbb{Q}[\mathbf{x}]$  we consider the ideal  $I$  of  $\mathbb{Q}[\mathbf{x}]$  generated by the polynomials in  $F$ .

**Definition 6.5.** A syzygy of  $I$  with respect to the basis  $F = (f_1, \dots, f_m)$  is an element  $\mathbf{g} = (g_1, \dots, g_m) \in \mathbb{Q}[\mathbf{x}]^m$  such that

$$g_1 f_1 + \dots + g_m f_m = 0.$$

A syzygy of the form

$$\mathbf{g} = (0, \dots, 0, f_j, 0, \dots, 0, -f_i, 0, \dots, 0),$$

where  $f_j$  is the  $i$ th and  $-f_i$  is the  $j$ th entry of  $\mathbf{g}$ , is called a trivial syzygy. The set of all syzygies for  $F$  is denoted by  $\text{Syz}(F)$ .

In the definition of a syzygy, some authors require  $F = (f_1, \dots, f_m)$  to be a minimal generating set of the ideal  $I$  (for example, [25]). However, for the purpose of this paper, we do not require such a condition on the generators. Moreover, it is well-known that  $\text{Syz}(F)$  form a  $\mathbb{Q}[\mathbf{x}]$ -submodule of  $\mathbb{Q}[\mathbf{x}]^m$ .

**Remark 6.6.** In this section, for computational purposes, we consider only polynomials over the rational numbers, that is, polynomials in  $\mathbb{Q}[\mathbf{x}]$ . Note that since  $\mathbb{Q} \subset \mathbb{R}$  all computations are also valid over  $\mathbb{R}$  and so all results can be considered as polynomials in  $\mathbb{R}[\mathbf{x}]$  making them applicable to CRNs.

**Example 6.7.** Consider the ideal  $I = \langle x_1, x_2 \rangle \subseteq \mathbb{Q}[x_1, x_2]$ . Since

$$(-x_2)x_1 + (x_1)x_2 = 0,$$

the vector  $\mathbf{g} = (-x_2, x_1) \in \mathbb{Q}[x_1, x_2]^2$  is a trivial syzygy of the ideal  $I$  with respect to  $F = \{x_1, x_2\}$ . One can easily see that the  $\mathbb{Q}[x_1, x_2]$ -module  $\text{Syz}(F)$  is generated by  $\mathbf{g}$ , that is, every syzygy is a multiplication of  $\mathbf{g}$  by a polynomial in  $\mathbb{Q}[x_1, x_2]$ .

Syzygies have been extensively studied in commutative algebra and algebraic geometry. Buchberger's algorithm for computing Gröbner bases [10, 14] leads to an algorithm which computes a basis of a syzygy module (see Schreyer's results in [9, 58]). One can also obtain a basis of the syzygy module of  $I$  with respect to a generating set  $F$  as a by-product of computing a Gröbner basis of  $I$  via signature-based algorithms [30, 36, 71]. Many computer algebra systems such as Singular, CoCoA, etc., include implementations for syzygy computations. For more on syzygies and in particular on algorithms, we refer to the books of Becker, Weispfenning, and Kredel [8] and Cox, Little, and O'Shea [17, Chapter 5.3].

Above we mentioned that  $\text{Syz}(F)$  is a  $\mathbb{Q}[\mathbf{x}]$ -module. It is clear that  $\text{Syz}(F)$  is a  $\mathbb{Q}$ -vector space. We will need to consider here the structure of  $\text{Syz}(F)$  as a vector space instead of a module since for the computation of conservation laws we need to restrict to syzygies with a specific property which is only respected by the  $\mathbb{Q}$ -vector space structure and not by the  $\mathbb{Q}[\mathbf{x}]$ -module structure, as we will see in subsections 6.2.3 and 6.2.4.



**Lemma 6.8.** Let  $G$  be a Gröbner basis for the  $\mathbb{Q}[\mathbf{x}]$ -module  $\text{Syz}(F)$  and denote the set of monomials in  $\mathbb{Q}[\mathbf{x}]$  by  $[\mathbf{x}]$ . Consider  $\text{Syz}(F)$  as a vector space over  $\mathbb{Q}$ . Then

$$\text{Gen}(\text{Syz}(F)) := \{u\mathbf{g} \mid \mathbf{g} \in G, u \in [\mathbf{x}]\}$$

is a generating set for the  $\mathbb{Q}$ -vector space  $\text{Syz}(F)$ .

*Proof.* Let  $\mathbf{s} \in \text{Syz}(F)$  be a syzygy. Since  $G = \{\mathbf{g}_1, \dots, \mathbf{g}_r\}$  is a Gröbner basis for the  $\mathbb{Q}[\mathbf{x}]$ -module  $\text{Syz}(F)$ , there are  $h_1, \dots, h_r \in \mathbb{Q}[\mathbf{x}]$  such that

$$\mathbf{s} = \sum_{i=1}^r h_i \mathbf{g}_i.$$

For each  $h_i$  there are monomials  $u_{ij}$  with  $j = 1, \dots, r_i$  with some  $r_i \in \mathbb{N}$  and constants  $c_{ij} \in \mathbb{Q}$  such that

$$h_i = \sum_{j=1}^{r_i} c_{ij} u_{ij}.$$

We conclude that

$$\mathbf{s} = \sum_{i=1}^r h_i \mathbf{g}_i = \sum_{i=1}^r \sum_{j=1}^{r_i} c_{ij} u_{ij} \mathbf{g}_i$$

is a  $\mathbb{Q}$ -linear combination of elements of  $\{u\mathbf{g} \mid \mathbf{g} \in G, u \in [\mathbf{x}]\}$ . ■

**Remark 6.9.** If we choose a nonnegative integer  $d \in \mathbb{N}$ , then the syzygies

$$\text{Syz}(F)_d := \{\mathbf{g} = (g_1, \dots, g_m) \in \text{Syz}(F) \mid \deg(g_i) \leq d, 1 \leq i \leq m\}$$

of degree at most  $d$  form a finite dimensional  $\mathbb{Q}$ -vector subspace of  $\text{Syz}(F)$ . By the proof of Proposition 4, Chapter 9, section 3 in [18], if  $G$  is a Gröbner basis with respect to a degree ordering, then the set

$$\text{Gen}(\text{Syz}(F)_d) := \{u\mathbf{g} \in \text{Gen}(\text{Syz}(F)) \mid \deg(u\mathbf{g}_i) \leq d, 1 \leq i \leq m\}$$

is a finite generating set for  $\text{Syz}(F)_d$ . Furthermore, those elements of  $\text{Gen}(\text{Syz}(F)_d)$  whose leading terms are different form a basis for the vector space  $\text{Syz}(F)_d$ . Alternatively, one can use Gaussian elimination to determine a basis of  $\text{Syz}(F)_d$  from  $\text{Gen}(\text{Syz}(F)_d)$ .

**6.2.2. Comprehensive Gröbner systems.** To compute parametric conservation laws from syzygies for CRNs we need to handle polynomial equations whose coefficients are again polynomials in the parameters, that is, polynomials

$$F = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_m(\mathbf{k}, \mathbf{x}))$$

in the ring  $\mathbb{Q}[\mathbf{k}][\mathbf{x}]$  in the indeterminates  $\mathbf{x} = (x_1, \dots, x_n)$  over the coefficient ring  $\mathbb{Q}[\mathbf{k}]$  in the indeterminates  $\mathbf{k} = (k_1, \dots, k_r)$ . The respective tool from computational algebra is the notion of a CGS.

A specialization of  $\mathbb{Q}[\mathbf{k}]$  is a homomorphism  $\sigma: \mathbb{Q}[\mathbf{k}] \rightarrow \mathbb{Q}$ . Every element  $\alpha \in \mathbb{Q}^r$  defines a specialization  $\sigma_\alpha$  by evaluating  $\mathbf{k}$  at  $\alpha$  and it can be canonically extended to a specialization

$$\sigma_\alpha: \mathbb{Q}[\mathbf{k}][\mathbf{x}] \rightarrow \mathbb{Q}[\mathbf{x}].$$

**Definition 6.10.** Let  $F$  and  $G_1, \dots, G_l$  be finite subsets of  $\mathbb{Q}[\mathbf{k}][\mathbf{x}]$ . Further let  $A_1, \dots, A_l$  be semialgebraic sets of  $\mathbb{Q}^r$  and let  $S \subseteq \mathbb{Q}^r$  such that  $S \subset A_1 \cup \dots \cup A_l$ . A finite set

$$\mathcal{G} = \{(A_1, G_1), \dots, (A_l, G_l)\}$$

is called a CGS on  $S$  for  $F$  if for all  $i = 1, \dots, l$  the set  $\sigma_{\alpha}(G_i)$  is a Gröbner basis for the ideal  $\langle \sigma_{\alpha}(F) \rangle \subset \mathbb{Q}[\mathbf{x}]$  for every  $\alpha \in A_i$ . Each  $(A_i, G_i)$  is called a branch of  $\mathcal{G}$ . The generic branch corresponds to the semialgebraic set that assumes all leading coefficients are nonzero and ignores the other cases. If  $S = \mathbb{Q}^r$ , then  $\mathcal{G}$  is called a CGS for  $F$ .

CGSs were introduced by Weispfenning in 1992 in [73]. Independent of Weispfenning, Kapur introduced the same concept in [39], calling it parametric Gröbner bases. Today in the literature there are many publications dealing with the theory and algorithmics of CGSs, for example, [19, 35, 40, 41, 45, 52, 69], only mentioning a few. For a survey paper on this topic see [46]. Moreover, several computer algebra systems provide implementations of algorithms to compute CGSs, such as *Reduce* [37, 38, 66] and *Singular* [20].

**Lemma 6.11** (cf. Lemma 3.7, Chapter 5.3 in [17]). Let  $\{(A_1, G_1), \dots, (A_l, G_l)\}$  be a CGS of the ideal in  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$  generated by

$$F = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x})).$$

For every  $i$ ,  $1 \leq i \leq l$ , there exists a linear map that converts a basis of the syzygy modules  $\text{Syz}(F)$  to a basis of the syzygy module  $\text{Syz}(G_i)$ . Also, there exists a map that converts a basis of the syzygy module  $\text{Syz}(G_i)$  into a basis of the parametric syzygies for  $F$  only valid on the semialgebraic sets  $A_i$ .

**Proof.** Consider a branch  $(A_i, G_i)$  of the given CGS. In the following, we mean by  $G_i$  the vector of elements of the Gröbner basis  $G_i$  instead of the set  $G_i$ . One can use the algorithms described in [17, Chapter 5.3] to compute a basis of the  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -module  $\text{Syz}(G_i)$  of the ideal generated by the elements of  $G_i$ . If  $\mathbf{g} \in \text{Syz}(G_i)$ , then for every  $\alpha \in A_i$ , the specialization  $\sigma_{\alpha}(\mathbf{g})$  will be a syzygy of the ideal generated by  $\sigma_{\alpha}(G_i)$ . Following [17, Chapter 5.3, Lemma 3.7], there exist matrices  $B^{1,i}$  and  $B^{2,i}$  such that

$$G_i B^{1,i} = F^T \quad \text{and} \quad F B^{2,i} = G_i^T.$$

We obtain that for every  $\mathbf{g} \in \text{Syz}(G_i)$  the product  $(B^{2,i} \mathbf{g}^T)^T$  is a syzygy of  $F$ . On the other hand for every  $\mathbf{f} \in \text{Syz}(F)$  we have  $(B^{1,i} \mathbf{f}^T)^T \in \text{Syz}(G_i)$ . The linear map  $B^{1,i}$  between the syzygy modules  $\text{Syz}(F)$  and  $\text{Syz}(G_i)$  is not necessarily one-to-one, and one may not obtain a basis for  $\text{Syz}(F)$  by converting a basis of  $\text{Syz}(G_i)$  via  $B^{2,i}$ . However, the kernel of  $B^{1,i}$  will give us the missing elements of a basis of  $\text{Syz}(F)$ . Moreover, we have

$$G_i \mathbf{g}^T = (B^{2,i})^T F^T \mathbf{g}^T = (B^{2,i})^T F^T B^{1,i} \mathbf{f}^T = F^T \mathbf{f}$$

since  $F^T = G_i B^{1,i} = (B^{2,i})^T F^T B^{1,i}$ . Applying  $\sigma_{\alpha}$  to the equation we obtain

$$0 = \sigma_{\alpha}(G_i) \sigma_{\alpha}(\mathbf{g}) = \sigma_{\alpha}(F) \sigma_{\alpha}(\mathbf{f}),$$

and therefore,  $\sigma_\alpha(\mathbf{f})$  is a syzygy of  $\sigma_\alpha(F)$  for every  $\alpha \in A_i$ . Hence, having computed a CGS of the ideal generated by  $F$  with branches  $(A_i, G_i)$  and bases of the syzygy modules  $\text{Syz}(G_i)$ , we obtain bases of the parametric syzygies for  $F$  only valid on the semialgebraic sets  $A_i$  by applying the conversion matrices described above to the bases elements of  $\text{Syz}(G_i)$ . ■

Similar to the above lemma, one can use [17, Proposition 3.8, Chapter 5.3] in order to construct a matrix that converts Gröbner bases for  $\text{Syz}(G_i)$  into a Gröbner basis for  $\text{Syz}(F)$ , respecting the relevant specializations.

*Example 6.12.* We consider here the ideal generated by the polynomials

$$F(\mathbf{k}, \mathbf{x}) = (-k_1x_1, -k_2x_2, (k_1 + k_2)x_1x_2).$$

Computing a CGS of the ideal generated by  $F(\mathbf{k}, \mathbf{x})$  we obtain the four branches

$$\begin{aligned} A_1 &= V(\langle 0 \rangle) \setminus V(\langle k_1k_2 \rangle), \quad G_1 = (k_2x_2, k_1x_1), \\ A_2 &= V(\langle k_2 \rangle) \setminus V(\langle k_1 \rangle), \quad G_2 = (k_1x_1), \\ A_3 &= \{V(\langle k_2, k_1 \rangle) \setminus V(\langle 1 \rangle)\}, \quad G_3 = (0), \\ A_4 &= V(\langle k_1 \rangle) \setminus V(\langle k_2 \rangle), \quad G_4 = (k_2x_2). \end{aligned}$$

We only look at the first branch, which is the generic branch. We compute the conversion matrices

$$B^{1,1} = \begin{pmatrix} 0 & -1 & x_1(k_1 + k_2)/k_2 \\ -1 & 0 & 0 \end{pmatrix} \text{ and } B^{2,1} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \\ 0 & 0 \end{pmatrix},$$

which satisfy  $G_1B^{1,1} = F^T$  and  $FB^{2,1} = G_1^T$ . The Syzygy module of  $G_1$  is generated by  $\mathbf{g} = (-k_1x_1, k_2x_2)$ . Computing  $(B^{2,1}\mathbf{g}^T)^T$ , one obtains the syzygy  $\mathbf{f}_1 = (-k_2x_2, k_1x_1, 0)$  of  $F$ . The kernel of the matrix  $B^{1,1}$  gives us the additional syzygy  $\mathbf{f}_2 = (0, k_1x_1 + k_2x_1, k_2)$  of  $F$ . The syzygy module of  $F$  is generated by  $\mathbf{f}_1$  and  $\mathbf{f}_2$ . Alternatively, one can find the whole generating set of  $\text{Syz}(F)$  using [17, Proposition 3.8]

$$\text{Syz}(F) = \langle As_{ij}, \mathbf{g}_1, \dots, \mathbf{g}_m \rangle.$$

*Example 6.13.* We consider the system

$$F(\mathbf{k}, \mathbf{x}) = (f_1, f_2, f_3)(x_2 + x_1, x_2 + x_1 + 1, k_1x_2).$$

This example shows how the system and its syzygies change in the different branches. We have here two branches. In the second branch, we have  $k_1 = 0$  and so we only find the trivial syzygy  $\mathbf{f} = (-f_2, f_1, 0)$ . In the generic branch, the parameter has to satisfy  $k_1 \neq 0$ . Here we find more and nontrivial syzygies, namely

$$\mathbf{f}_1 = (-k_1x_1 - k_1, k_1x_1, 1) \text{ and } \mathbf{f}_2 = (-k_1x_2, k_1x_2, -1).$$

*Example 6.14.* We consider the system

$$F(\mathbf{k}, \mathbf{x}) = (x_1 + k_1x_2 + k_2, k_1x_1 + x_2 + k_2).$$

As in the example above we want to see how the system and its syzygies change in the different branches. In the first branch, that is, the generic one, the parameters have to satisfy  $k_1 \neq 0$ ,  $k_2 \neq 0$  and we find that the syzygy module is generated by the trivial syzygy  $\mathbf{g} = (k_1x_1 + x_2 + k_2, -x_1 - k_1x_2 - k_2)$ . In the second branch the parameters are  $k_1 = -1$ ,  $k_2 \neq 0$  and so the system reduces to

$$F(\mathbf{k}, \mathbf{x}) = (x_1 - x_2 + k_2, -x_1 + x_2 + k_2).$$

Again we only find the nonzero trivial syzygy here. In the third branch, the parameters have values  $k_1 = 1$ ,  $k_2 = 0$ , which reduces our system to

$$F(\mathbf{k}, \mathbf{x}) = (x_1 + x_2, x_1 + x_2).$$

Here the syzygy module will be generated by  $\mathbf{f} = (1, -1)$ .

**6.2.3. The connection between syzygies and conservation laws.** In this section, we consider system (2.1), that is, the system of ODEs of the form

$$\dot{x}_1 = f_1(\mathbf{k}, \mathbf{x}), \dots, \dot{x}_n = f_n(\mathbf{k}, \mathbf{x}) \in \mathbb{Z}[\mathbf{k}, \mathbf{x}].$$

We will explain how a parametric conservation law of this ODE system leads to a syzygy of the ideal generated by  $F = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))$  and how one can obtain parametric conservation laws from specific elements of  $\text{Syz}(F)$ .

We recall that the curl of the vector field  $\mathbf{g} = (g_1(\mathbf{k}, \mathbf{x}), \dots, g_n(\mathbf{k}, \mathbf{x}))$  is defined as the antisymmetric tensor field  $\nabla \times \mathbf{g}$  with entries

$$\frac{\partial g_i(\mathbf{k}, \mathbf{x})}{\partial x_j} - \frac{\partial g_j(\mathbf{k}, \mathbf{x})}{\partial x_i} \quad \text{with } 1 \leq i, j \leq n.$$

**Definition 6.15.**

1. For the dynamical system 2.1 and its r.h.s.  $F$  we define the set of conservative syzygies as

$$\text{Syz}_{\nabla}(F) := \{\mathbf{g} \in \text{Syz}(F) \mid \nabla \times \mathbf{g} = 0\}.$$

2. For a positive integer  $d \in \mathbb{N}$  we define the set of conservative syzygies of degree at most  $d$  as

$$\text{Syz}_{\nabla}(F)_d := \{\mathbf{g} \in \text{Syz}(F)_d \mid \nabla \times \mathbf{g} = 0\}.$$

3. We define  $\mathcal{C}(F) \subset \mathbb{Q}[\mathbf{k}, \mathbf{x}]$  to be the set of polynomial conservation laws of  $F$ .

**Proposition 6.16.** There is a one-to-one correspondence between conservation laws and conservative syzygies:

$$\varphi_1 : \mathcal{C}(F) \rightarrow \text{Syz}_{\nabla}(F), \quad \phi(\mathbf{k}, \mathbf{x}) \mapsto \nabla \phi(\mathbf{k}, \mathbf{x}) \quad \text{and} \quad \varphi_2 : \text{Syz}_{\nabla}(F) \rightarrow \mathcal{C}(F), \quad \mathbf{g} \mapsto \int \mathbf{g},$$

with  $\varphi_1 \circ \varphi_2 = \text{id}_{\text{Syz}_{\nabla}(F)}$  and  $\varphi_2 \circ \varphi_1 = \text{id}_{\mathcal{C}(F)}$ , where  $\int \mathbf{g}$  denotes the potential of a conservative vector field  $\mathbf{g} \in \text{Syz}_{\nabla}(F)$ .

*Proof.* First, we show that  $\varphi_1$  is well-defined. Let  $\phi(\mathbf{k}, \mathbf{x}) \in \mathcal{C}(F)$ , that is, let  $\phi(\mathbf{k}, \mathbf{x}) \in \mathbb{Q}[\mathbf{k}, \mathbf{x}]$  be a parametric exact polynomial conservation law for (2.1) with semialgebraic set  $V_{\mathbf{k}}$  for the parameters  $\mathbf{k}$ . We fix this semialgebraic set throughout the proof and ignore  $k$ . By Definition 3.4, we have

$$\sum_{i=1}^n \frac{\partial \phi}{\partial x_i}(\mathbf{k}, \mathbf{x}) f_i(\mathbf{k}, \mathbf{x}) = 0 \quad \forall \mathbf{k} \in V_{\mathbf{k}}, \forall \mathbf{x} \in \mathbb{R}^n,$$

which implies that

$$\left( \frac{\partial \phi}{\partial x_1}(\mathbf{k}, \mathbf{x}), \dots, \frac{\partial \phi}{\partial x_n}(\mathbf{k}, \mathbf{x}) \right) \in \text{Syz}(F).$$

In other words, this means that the gradient

$$\nabla \phi(\mathbf{k}, \mathbf{x}) = \left( \frac{\partial \phi}{\partial x_1}(\mathbf{k}, \mathbf{x}), \dots, \frac{\partial \phi}{\partial x_n}(\mathbf{k}, \mathbf{x}) \right)$$

of a parametric polynomial conservation law  $\phi(\mathbf{k}, \mathbf{x})$  is a syzygy of the ideal generated by  $F$  when  $\mathbf{k} \in V_{\mathbf{k}}$ . Moreover, by well-known identities of gradient and curl, we have that

$$\nabla \times (\nabla \phi) = 0$$

and so  $\nabla \phi \in \text{Syz}_{\nabla}(F)$ . This shows that  $\varphi_1$  is well-defined.

Second, we show that  $\varphi_2$  is well-defined. Assume that  $\mathbf{g} = (g_1, \dots, g_n) \in \text{Syz}_{\nabla}(F)$  is a parametric syzygy of the ideal generated by  $F$  with semialgebraic set  $V_{\mathbf{k}}$  obtained from the CGS described at the end of subsection 6.2.2. Therefore,

$$f_1(\mathbf{k}, \mathbf{x})g_1(\mathbf{k}, \mathbf{x}) + \dots + f_n(\mathbf{k}, \mathbf{x})g_n(\mathbf{k}, \mathbf{x}) = 0$$

for points  $\mathbf{k} \in V_{\mathbf{k}}$ .

It is well-known in vector calculus that a potential function  $\phi$  for  $\mathbf{g}$  as a vector field is well-defined in a simply connected domain  $\mathcal{D} \subset \mathbb{R}^n$  if and only if  $\nabla \times \mathbf{g} = 0$  everywhere in  $\mathcal{D}$ , i.e., only when  $\mathbf{g}$  defines a conservative (or irrotational) vector field in  $\mathcal{D}$  (see [65]). In this case,  $\phi$  satisfies

$$\nabla \phi(\mathbf{k}, \mathbf{x}) = \mathbf{g}.$$

By the well-known vector calculus method, one can integrate  $\mathbf{g}$  along a suitable path to obtain the corresponding potential function. For the sake of self-containedness, we explain one possible integration below. Consider the differential form  $d\mathbf{g} = g_1 dx_1 + \dots + g_n dx_n$  and take a piecewise smooth curve connecting the origin with  $\mathbf{x}$ , for instance,

$$\begin{aligned} \mathcal{C} = & \{(sx_1, 0, \dots, 0) \mid s \in [0, 1]\} \\ & \cup \{(x_1, sx_2, 0, \dots, 0) \mid s \in [0, 1]\} \cup \dots \cup \{(x_1, \dots, x_{n-1}, sx_n) \mid s \in [0, 1]\}. \end{aligned}$$

Integrating the differential form  $d\mathbf{g}$  along the curve  $\mathcal{C}$  yields

$$(6.1) \quad \int_0^{x_1} g_1(\cdot, 0, \dots, 0) dx_1 + \int_0^{x_2} g_2(x_1, \cdot, 0, \dots, 0) dx_2 + \dots + \int_0^{x_n} g_n(x_1, \dots, x_{n-1}, \cdot) dx_n,$$

where the constant of the integration is assumed to be zero. We denote this integration (with constant being zero) by  $\int \mathbf{g}$ . Denote that if  $\nabla \times \mathbf{g} \neq 0$ , the result of the integration depends on the path  $\mathcal{C}$  and there is no function  $\phi(\mathbf{k}, \mathbf{x})$  satisfying  $\nabla \phi(\mathbf{k}, \mathbf{x}) = \mathbf{g}$  for all  $\mathbf{x} \in \mathbb{R}^n$ .

Therefore, if  $\mathbf{g} \in \text{Syz}_{\nabla}(F)$ , then integration implies that  $\int \mathbf{g}$  is an element of  $\mathcal{C}(F)$  as we showed above. This shows that the definition of the map  $\varphi_2$  is correct.

Having shown that both  $\varphi_1$  and  $\varphi_2$  are well-defined, we just need to show that  $\varphi_1$  and  $\varphi_2$  are inverse to each other. The identity  $\varphi_2 \circ \varphi_1 = \text{id}_{\mathcal{C}(F)}$  follows from  $\int \nabla \phi = \phi$ . Moreover, since  $\nabla \int \mathbf{g} = \mathbf{g}$ , we have  $\varphi_1 \circ \varphi_2 = \text{id}_{\text{Syz}_{\nabla}(F)}$ . ■

**Remark 6.17.** In vector calculus, there are other standard methods to compute from a conservative syzygy  $\mathbf{g} = (g_1(\mathbf{k}, \mathbf{x}), \dots, g_n(\mathbf{k}, \mathbf{x}))$ , the corresponding conservation law  $\phi(\mathbf{k}, \mathbf{x})$ . One can first start with integrating  $g_n$  with respect to  $x_n$ , obtaining

$$\phi(\mathbf{k}, \mathbf{x}) = \int_{x_n} g_n(\mathbf{k}, \mathbf{x}) dx_n + c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1}),$$

where the integration constant  $c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})$  is in  $\mathbb{Q}[\mathbf{k}, x_1, \dots, x_{n-1}]$ . Let  $p_n := \int_{x_n} g_n(\mathbf{k}, \mathbf{x}) dx_n$  where we choose the integration constant to be zero. Then in order to compute  $c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})$ , one can use the fact that  $\frac{\partial \phi(\mathbf{k}, \mathbf{x})}{\partial x_{n-1}} = g_{n-1}(\mathbf{k}, \mathbf{x})$  and obtain an equation for  $\frac{\partial c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})}{\partial x_{n-1}}$ , namely

$$\frac{\partial \phi(\mathbf{k}, \mathbf{x})}{\partial x_{n-1}} = \frac{\partial p_n(\mathbf{k}, \mathbf{x})}{\partial x_{n-1}} + \frac{\partial c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})}{\partial x_{n-1}}.$$

Replacing  $\frac{\partial \phi(\mathbf{k}, \mathbf{x})}{\partial x_{n-1}}$  with  $g_{n-1}(\mathbf{k}, \mathbf{x})$  one obtains

$$g_{n-1}(\mathbf{k}, \mathbf{x}) = \frac{\partial p_n(\mathbf{k}, \mathbf{x})}{\partial x_{n-1}} + \frac{\partial c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})}{\partial x_{n-1}}.$$

As we know the polynomials  $g_{n-1}(\mathbf{k}, \mathbf{x})$  and  $\frac{\partial p_n}{\partial x_{n-1}}$ , one can obtain equations for the coefficients of  $\frac{\partial c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})}{\partial x_{n-1}}$ . Continuing the above procedure by taking partial derivative of  $\phi(\mathbf{k}, \mathbf{x})$  with respect to the variable  $x_{n-2}$ , replacing it with  $g_{n-2}(\mathbf{k}, \mathbf{x})$ , one can obtain more equations for the coefficients of the integration constant  $c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})$ . As  $\mathbf{g}(\mathbf{k}, \mathbf{x})$  is a conservative syzygy, the existence of a conservation law  $\phi(\mathbf{k}, \mathbf{x})$  such that  $\nabla \phi(\mathbf{k}, \mathbf{x}) = \mathbf{g}(\mathbf{k}, \mathbf{x})$  is guaranteed, which implies that the equations obtained for the coefficients of  $c_{n-1}(\mathbf{k}, x_1, \dots, x_{n-1})$  yields a unique solution.

An alternative method for integration is considering the integration

$$(6.2) \quad \phi = \int g_1 dx_1 + \dots \int g_n dx_n + C,$$

and then removing the duplicated terms that show up in the integrals.

**Example 6.18.** Consider the trivial syzygy  $\mathbf{g} = (-x_2, x_1)$  for the ideal in  $\mathbb{Q}[x_1, x_2]$  generated by  $F = (x_1, x_2)$ . As the vector field arising from this particular syzygy is not irrotational, i.e.,  $\nabla \times \mathbf{g} \neq 0$ , one cannot obtain a parametric polynomial conservation law from this syzygy. But there are generators of the same ideal with a trivial syzygy which is the gradient of a parametric

conservation law. For example, consider the generators  $F = (x_1, -x_2)$  with the trivial syzygy  $\mathbf{g} = (x_2, x_1)$ . Then  $\nabla \times \mathbf{g} = 0$ , which means that the trivial syzygy is irrotational and so it is the gradient of a parametric conservation law, which is  $x_1x_2 - c$ , where  $c$  is a constant.

**Example 6.19 (linear conservation laws).** Assume that  $F$  admits a degree zero syzygy  $\mathbf{g}$ . As the entries of a degree zero syzygy are constants, we have  $\nabla \times \mathbf{g} = 0$  and so a degree zero syzygy is conservative. This means that  $\text{Syz}_{\nabla}(F)_0 = \text{Syz}(F)_0$ . Conservation laws corresponding to degree zero syzygies are the well-known linear conservation laws.

**Example 6.20.** A syzygy of the form  $\mathbf{g} = (x_1^{\alpha_1}, \dots, x_n^{\alpha_n})$  is obviously conservative and leads to the conservation law

$$\phi(\mathbf{x}) = \alpha_1^{-1} x_1^{\alpha_1+1} + \dots + \alpha_n^{-1} x_n^{\alpha_n+1}.$$

**Remark 6.21.**

1. First we want to point out that  $\mathcal{C}(F)$  is not necessarily an ideal in the ring  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . Indeed, let  $\phi(\mathbf{k}, \mathbf{x}) \in \mathcal{C}(F)$ ,  $\phi(\mathbf{k}, \mathbf{x}) \neq 0$ , and  $h \in \mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . Then by the chain rule, we have

$$\nabla(h\phi) = h\nabla\phi(\mathbf{k}, \mathbf{x}) + \phi(\mathbf{k}, \mathbf{x})\nabla h.$$

Since  $\phi(\mathbf{k}, \mathbf{x})$  is a conservation law, that is,  $\nabla\phi(\mathbf{k}, \mathbf{x}) \cdot F = 0$ , we obtain that

$$\nabla(h\phi(\mathbf{k}, \mathbf{x})) \cdot F = (h\nabla\phi + \phi(\mathbf{k}, \mathbf{x})\nabla h) \cdot F = \phi(\mathbf{k}, \mathbf{x})(\nabla h \cdot F)$$

and  $\nabla h \cdot F$  is in general not zero as the following example shows. In Example 6.18, for the generating set  $F = (x_1, -x_2)$ , we found the conservation law  $\phi(\mathbf{x}) = x_1x_2$ . One easily checks that the product  $x_1\phi(\mathbf{x})$  is not a conservation law.

2. The set  $\text{Syz}_{\nabla}(F)$  is not necessarily a  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -submodule of the  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -module  $\text{Syz}(F)$ . Indeed, for  $h \in \mathbb{Q}[\mathbf{k}, \mathbf{x}]$  and  $\mathbf{g} \in \text{Syz}_{\nabla}(F)$  we obtain that

$$\nabla \times (h\mathbf{g}) = h(\nabla \times \mathbf{g}) + \nabla h \times \mathbf{g},$$

where we mean by  $\mathbf{u} \times \mathbf{v}$  for two vectors  $\mathbf{u} = (u_1, \dots, u_n)$  and  $\mathbf{v} = (v_1, \dots, v_n)$  the vector with entries

$$u_i v_j - u_j v_i \quad \text{with } 1 \leq i, j \leq n, \quad i \neq j.$$

As  $\mathbf{g} \in \text{Syz}_{\nabla}(F)$ , we have that  $\nabla \times \mathbf{g} = 0$ , however, not necessarily  $\nabla h \times \mathbf{g} = 0$  as the example at the end shows. Therefore,  $\text{Syz}_{\nabla}(F)$  is not necessarily closed under multiplication by the elements of  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ , which implies that  $\text{Syz}_{\nabla}(F)$  is not necessarily a  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -submodule of  $\text{Syz}(F)$ .

As a counterexample, consider again the generating set  $F = (x_1, -x_2)$  in Example 6.18. It yields the conservative syzygy  $\mathbf{g} = (x_2, x_1)$ . However, one easily checks that the product  $x_1\mathbf{g}$  is not a conservative syzygy, that is,  $\nabla \times (x_1\mathbf{g}) \neq 0$ .

**Remark 6.22.** We want to point out that by Remark 6.21, remark 2, it is not sufficient to check only the elements of a basis of the syzygies module  $\text{Syz}(F)$  to be conservative to obtain from the ones which are conservative all of  $\text{Syz}_{\nabla}(F)$ , since  $\text{Syz}_{\nabla}(F)$  does not have the structure of a  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -module.



**Proposition 6.23.** *The set  $\text{Syz}_{\nabla}(F)$  is a  $\mathbb{Q}$ -vector space.*

*Proof.* It is easy to check that the operator  $\nabla \times (\cdot)$  is  $\mathbb{Q}$ -linear, that is, for  $c \in \mathbb{Q}$  and  $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{Q}[\mathbf{k}, \mathbf{x}]^n$  we have

$$\nabla \times (c\mathbf{g}_1 + \mathbf{g}_2) = c\nabla \times \mathbf{g}_1 + \nabla \times \mathbf{g}_2.$$

Thus, if  $\mathbf{g}_1, \mathbf{g}_2 \in \text{Syz}_{\nabla}(F)$ , then  $\nabla \times (c\mathbf{g}_1 + \mathbf{g}_2) = 0$  and so  $\text{Syz}_{\nabla}(F)$  is a  $\mathbb{Q}$ -vector space. ■

We will use the  $\mathbb{Q}$ -vector space structure of  $\text{Syz}_{\nabla}(F)$  for computing conservation laws. Apart from this, the conservation laws form an algebra. Although this is well-known in the literature on first integrals (e.g., see Arnold's book [4, section 10.3]), we present it here and detail the proof, not only for the sake of self-containedness, but also because we will refer to the algebra structure of conservation laws and discuss its generators when we present our benchmark results.

**Proposition 6.24.** *The set  $\mathcal{C}(F)$  is a  $\mathbb{Q}$ -subalgebra of  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ .*

*Proof.* Let  $\phi_1(\mathbf{k}, \mathbf{x}), \phi_2(\mathbf{k}, \mathbf{x}) \in \mathcal{C}(F)$  be two conservation laws. The chain rule implies that

$$\nabla(\phi_1(\mathbf{k}, \mathbf{x})\phi_2(\mathbf{k}, \mathbf{x})) = \phi_2(\mathbf{k}, \mathbf{x})\nabla\phi_1(\mathbf{k}, \mathbf{x}) + \phi_1(\mathbf{k}, \mathbf{x})\nabla\phi_2(\mathbf{k}, \mathbf{x}).$$

Moreover, since  $\phi_1(\mathbf{k}, \mathbf{x}), \phi_2(\mathbf{k}, \mathbf{x})$  are conservation laws, we have that  $\nabla\phi_1(\mathbf{k}, \mathbf{x}) \cdot F = \nabla\phi_2(\mathbf{k}, \mathbf{x}) \cdot F = 0$  and so we obtain

$$\begin{aligned} \nabla(\phi_1(\mathbf{k}, \mathbf{x})\phi_2(\mathbf{k}, \mathbf{x})) \cdot F &= (\phi_2\nabla\phi_1 + \phi_1\nabla\phi_2) \cdot F \\ &= (\phi_2(\mathbf{k}, \mathbf{x})\nabla\phi_1(\mathbf{k}, \mathbf{x}) \cdot F) + (\phi_1(\mathbf{k}, \mathbf{x})\nabla\phi_2(\mathbf{k}, \mathbf{x}) \cdot F) \\ &= 0 + 0. \end{aligned}$$

So the product of two conservation laws is again a conservation law. Since  $\nabla$  is additive,  $\phi_1(\mathbf{k}, \mathbf{x}) + \phi_2(\mathbf{k}, \mathbf{x})$  is again a conservation law. Finally, for  $c \in \mathbb{Q}$  we have

$$\nabla(c\phi_1(\mathbf{k}, \mathbf{x})) \cdot F = c(\nabla\phi_1(\mathbf{k}, \mathbf{x}) \cdot F) = 0,$$

that is,  $c\phi_1(\mathbf{k}, \mathbf{x})$  is again a conservation law. This implies that  $\mathcal{C}(F)$  is a  $\mathbb{Q}$ -subalgebra of  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . ■

**6.2.4. Computing conservation laws from syzygies.** As pointed out in Remark 6.22, conservative syzygies do not form a  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ -module, and hence, conservative syzygies cannot be generated by the conservative elements of a basis for the syzygy module. However, we have seen in Proposition 6.23 that conservative syzygies form a  $\mathbb{Q}$ -vector space. Restricting to conservative syzygies up to degree  $d$ , one can compute a finite basis for the vector space  $\text{Syz}_{\nabla}(F)_d$  via a basis of the syzygy module  $\text{Syz}(F)_d$  by the  $\mathbb{Q}$ -linearity of  $\nabla \times (\cdot)$ .

**Proposition 6.25.**

1. *The set*

$$V_{\nabla}(F) = \{\nabla \times \mathbf{f} \mid \mathbf{f} \in \text{Syz}(F)\}$$

*is a  $\mathbb{Q}$ -vector space.*

2. The curl induces a surjective  $\mathbb{Q}$ -linear map

$$\nabla \times (\cdot) : \text{Syz}(F) \rightarrow V_{\nabla}(F), \quad \mathbf{f} \mapsto \nabla \times \mathbf{f}$$

with

$$\ker(\nabla \times (\cdot)) = \text{Syz}_{\nabla}(F).$$

*Proof.*

1. Since  $\text{Syz}(F)$  is a  $\mathbb{Q}$ -vector space and  $\nabla \times (\cdot)$  is a  $\mathbb{Q}$ -linear operator, its image is a  $\mathbb{Q}$ -vector space.
2. The map  $\nabla \times (\cdot)$  is clearly  $\mathbb{Q}$ -linear and surjective. Obviously every element of  $\ker(\nabla \times (\cdot))$  is an element of  $\text{Syz}_{\nabla}(F)$  and every syzygy  $\mathbf{g}$  with  $\nabla \times \mathbf{g} = 0$  lies in the kernel. ■

We recall that a basis for the  $\mathbb{Q}$ -vector space of syzygies up to a given degree bound  $d \in \mathbb{N}$  can be computed as in Remark 6.9. Then one can apply the curl operator and compute its kernel in order to obtain a basis for the conservative syzygies up to degree  $d$ .

We now combine our results into Algorithm 6.2, which computes for the r.h.s.  $F(\mathbf{k}, \mathbf{x})$  of an ODE system a basis of the  $\mathbb{Q}$ -vector space of parametric conservation laws up to a given degree  $d$ .

**Remark 6.26.** In line 14 of Algorithm 6.2 one needs to determine the intersection  $A_i \cap \mathbb{Q}_+^r$ , where  $A_i$  is the semialgebraic set computed via comprehensive Gröbner basis in line 3. The set  $A_i$  is described by equations and inequalities in  $\mathbb{Q}[\mathbf{k}]$ . By adding to them the inequalities  $k_1 > 0, \dots, k_r > 0$ , which represent the positive orthant, one obtains a set of equations and inequalities describing  $V_{\mathbf{k},i}$ . By applying the cylindrical algebraic decomposition, implemented in, e.g., QEPCAD [12], to the latter set of equations and inequations one may obtain another (though not necessarily better) representation of  $V_{\mathbf{k},i}$ . The program SLFQ (*Simplifying Large Formulas with QEPCAD*) [13] can be used once QEPCAD is applied to obtain such a simplified version.

**Remark 6.27.** Let  $\{\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x})\} \subseteq \mathcal{C}(F)_i$  be a set of  $s$  parametric polynomial conservation laws with respect to the semialgebraic set  $V_{\mathbf{k},i}$  returned by Algorithm 6.2. Then their Jacobian matrix

$$\begin{pmatrix} \frac{\partial \phi_1(\mathbf{k}, \mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \phi_1(\mathbf{k}, \mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \phi_s(\mathbf{k}, \mathbf{x})}{\partial x_1} & \cdots & \frac{\partial \phi_s(\mathbf{k}, \mathbf{x})}{\partial x_n} \end{pmatrix}$$

has rank  $s$ . Indeed, the conservation laws stem from  $\mathbb{Q}$ -linearly independent syzygies, that is, there are  $\mathbb{Q}$ -linearly independent syzygies  $\mathbf{g}_1, \dots, \mathbf{g}_s$  such that  $\nabla \phi_i = \mathbf{g}_i$  for all  $1 \leq i \leq s$ . Thus the Jacobian has rank  $s$ . Note that the conservation laws  $\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x})$  are not necessarily algebraically independent over  $\mathbb{Q}$ . One can check  $\phi_1(\mathbf{k}, \mathbf{x}), \dots, \phi_s(\mathbf{k}, \mathbf{x})$  for algebraic independence using Gröbner bases [18, Chapter 9, section 5] or linear algebra.

**Remark 6.28.** By Definition 3.4 parametric conservation laws depend on the rate constants  $\mathbf{k}$ , whereas unconditional conservation laws are independent of  $\mathbf{k}$ . If  $\phi$  is an unconditional

---

**Algorithm 6.2.** ParametricPolynomialConservationLawsViaSyzygies.

---

**Input:** A polynomial ODE system  $F(\mathbf{k}, \mathbf{x}) = (f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))$  whose r.h.s.  $f_i(\mathbf{k}, \mathbf{x})$  are polynomials in  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$  and a degree bound  $d$ .

**Output:** A set of pairs

$$\mathcal{S} = \{(V_{\mathbf{k},1}, \mathcal{C}(F)_1), \dots, (V_{\mathbf{k},l}, \mathcal{C}(F)_l)\},$$

where  $V_{\mathbf{k},i} \subseteq \mathbb{Q}_+^r$  is a semialgebraic set and  $\mathcal{C}(F)_i$  is a basis of the vector space of parametric polynomial conservation laws of degree at most  $d$  valid for all  $\mathbf{k} \in V_{\mathbf{k},i}$ .

- 1:  $\mathcal{S} := \emptyset$
- 2:  $I := \text{Ideal}(f_1(\mathbf{k}, \mathbf{x}), \dots, f_n(\mathbf{k}, \mathbf{x}))$  in  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$
- 3: Compute a CGS  $\mathcal{G} := \{(A_1, G_1), \dots, (A_l, G_l)\}$  of  $I$
- 4: **for**  $i = 1, \dots, l$  **do**
- 5:    $\mathcal{C}(F)_i := \emptyset$
- 6:   Compute a basis  $B(\text{Syz}(G_i))$  of the syzygy module  $\text{Syz}(G_i)$  of  $G_i$ .
- 7:   Convert  $B(\text{Syz}(G_i))$  into a basis  $B(\text{Syz}(F))$  of the parametric syzygies  $\text{Syz}(F)$  of  $F$  valid for  $A_i$ .
- 8:   Compute a basis  $B(\text{Syz}(F)_d)$  of the generating set

$$\text{Gen}(\text{Syz}(F)_d) = \{u\mathbf{g} \mid u \in [\mathbf{k}, \mathbf{x}], \mathbf{g} \in B(\text{Syz}(F)), \deg(u\mathbf{g}_i) \leq d\}$$

of the  $\mathbb{Q}$ -vector space  $\text{Syz}(F)_d$ .

- 9: Use  $B(\text{Syz}(F)_d)$  to compute a basis  $B$  of the kernel of the restriction of  $\nabla \times (\cdot)$  to  $\text{Syz}(F)_d$ .
  - 10:   **for**  $\mathbf{b} = (b_1, \dots, b_n) \in B$  **do**
  - 11:      $\phi(\mathbf{k}, \mathbf{x}) := \int \mathbf{b}$
  - 12:      $\mathcal{C}(F)_i := \mathcal{C}(F)_i \cup \{\phi(\mathbf{k}, \mathbf{x})\}$
  - 13:   **end for**
  - 14:   Compute  $V_{\mathbf{k},i} := A_i \cap \mathbb{Q}_+^r$
  - 15:    $\mathcal{S} := \mathcal{S} \cup \{(V_{\mathbf{k},i}, \mathcal{C}(F)_i)\}$
  - 16: **end for**
  - 17: **return**  $\mathcal{S}$ .
- 

conservation law, then it can be seen as a parametric conservation law as well, hence,  $\phi$  must be obtained from a syzygy that does not contain variables  $\mathbf{k}$ . Therefore, in order to compute unconditional conservation laws, we need to find those syzygies of the ideal generated by  $F(\mathbf{k}, \mathbf{x})$  that only depend on the variables  $\mathbf{x}$  and not on  $\mathbf{k}$ . To this end, we compute directly a basis of the syzygy module of  $F(\mathbf{k}, \mathbf{x})$  without the detour of the CGS using the elimination property of Gröbner bases. We fix a block term order on the variables  $\mathbf{k}, \mathbf{x}$  with  $\mathbf{k} > \mathbf{x}$ . If  $G$  is a Gröbner basis for  $\text{Syz}(F)$  with respect to the block order, then  $G \cap \mathbb{Q}[\mathbf{x}]^m$  is a Gröbner basis for  $\text{Syz}(F) \cap \mathbb{Q}[\mathbf{x}]^m$ ; see [43, Theorem 3.4.5]. This means that in order to compute unconditional conservation laws, one can modify Algorithm 6.2 by adding a computation of  $G \cap \mathbb{Q}[\mathbf{x}]^m$ . Also in order to compute a generating set for  $\text{Syz}(F)_d \cap \mathbb{Q}[\mathbf{x}]^m$  one needs to

multiply the elements of the Gröbner basis  $G \cap \mathbb{Q}[\mathbf{x}]^m$  with the monomials only in  $\mathbf{x}$ .

**Remark 6.29.** In the context of CRNs, some authors work with the ring  $\mathbb{Q}(\mathbf{k})[\mathbf{x}]$  without specializing  $\mathbf{k}$ , while in Algorithm 6.2, we work with the ring  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ , considering case distinctions done for CGS computation, as  $\mathbf{k}$  can be specialized in  $\mathbb{Q}_+^r$ . One can see that a Gröbner basis  $G$  of the ideal generated by  $F(\mathbf{k}, \mathbf{x})$  in  $\mathbb{Q}(\mathbf{k})[\mathbf{x}]$  corresponds to the Gröbner basis in the generic branch of a CGS in the ring  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . This fact about the generic branch has been considered in [53], and a new algorithm for computing CGS has been presented based on this. Details on how to obtain the semialgebraic set of the generic branch from  $G$  can be found in [53].

We implemented Algorithm 6.2 in a Singular library called `polconslaw.lib`. Our implementation uses the `grobcov` library for computing CGSs. We also implemented the approach in Remark 6.29 for computing polynomial conservation laws corresponding to the generic branch of a CGS in a Singular library, called `genpolconslaw.lib`, by just computing the conservation laws considering  $\mathbb{Q}(\mathbf{k})[\mathbf{x}]$ . Therefore, this approach is much faster than computing the whole CGS and picking up the generic branch. Our Singular libraries `polconslaw.lib` and `genpolconslaw.lib` are available from Zenodo<sup>1</sup> and GitHub.<sup>2</sup>

We present a series of examples clarifying our computational methods.

**Example 6.30.** For the system

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = x_1, \quad \dot{x}_3 = x_1 x_2,$$

the syzygy module has the basis

$$\{(0, x_2, -1), (x_1, -x_2, 0)\}.$$

Multiplying these two syzygies by all monomials in the variables  $x_1, x_2, x_3$  up to degree three including also the monomial 1, that is by the elements of the set

$$\{1, x_1, x_2, x_3, x_1^2, \dots, x_3^2 x_1, x_3^2 x_2, x_1 x_3 x_2\},$$

we obtain 40  $\mathbb{Q}$ -linearly independent syzygies. We compute for these syzygies the images under the map  $\nabla \times (\cdot)$  and then the  $\mathbb{Q}$ -linear dependencies between the results. We obtain that the kernel of the linear map  $\nabla \times (\cdot)$  has a basis formed by the elements

$$\begin{aligned} \mathbf{f}_1 &= (0, x_2, -1), \quad \mathbf{f}_2 = (0, x_2^3 - 2x_2 x_3, -x_2^2 + 2x_3), \quad \mathbf{f}_3 = (x_1, -x_2, 0), \\ \mathbf{f}_4 &= (x_1^3 - 2x_1 x_3, 0, -x_1^2 + 2x_3), \quad \mathbf{f}_5 = (x_2^2 x_1 - 2x_1 x_3, x_1^2 x_2 - x_2^3, -x_1^2 + 2x_3), \end{aligned}$$

that is,  $\mathbf{f}_1, \dots, \mathbf{f}_5$  span the  $\mathbb{Q}$ -vector space of all syzygies up to degree three with curl zero. Integration leads to the five unconditional conservation laws

$$\begin{aligned} \phi_1 &= \frac{x_2^2}{2} - x_3, \quad \phi_2 = \frac{1}{4}x_2^4 - x_2^2 x_3 + x_3^2, \quad \phi_3 = \frac{x_1^2}{2} - \frac{x_2^2}{2}, \quad \phi_4 = \frac{1}{4}x_1^4 - x_1^2 x_3 + x_3^2, \\ \phi_5 &= \frac{1}{2}x_1^2 x_2^2 - \frac{1}{4}x_2^4 - x_1^2 x_3 + x_3^2. \end{aligned}$$

<sup>1</sup><https://zenodo.org/record/7474523>.

<sup>2</sup><https://github.com/oradules/polynomial-conservation-laws>.

Note that  $\mathbf{f}_1$  and  $\mathbf{f}_3$  are the generators of the syzygy module, and they are both conservative, hence, they are integrated to the conservation laws  $\phi_1$  and  $\phi_2$ . One can see that the three other conservation laws obtained above are actually in the algebra generated by  $\phi_1$  and  $\phi_2$ . The syzygy  $\mathbf{f}_2$  is the product  $(x_2^2 - 2x_3)\mathbf{f}_1 = 2\phi_1\mathbf{f}_2$ ,  $\mathbf{f}_4$  is the product  $(x_1^2 - 2x_3)(\mathbf{f}_1 + \mathbf{f}_3) = 2(\phi_1 + \phi_3)(\mathbf{f}_1 + \mathbf{f}_3)$ , and the syzygy  $\mathbf{f}_5$  is the product  $(x_1^2 - 2x_3)\mathbf{f}_1 + (x_2^2 - 2x_3)\mathbf{f}_3 = 2(\phi_1 + \phi_3)\mathbf{f}_1 = 2\phi_1\mathbf{f}_3$ .

The computations for this example can be performed automatically using our Singular library `genpolconslaw.lib`, in the corresponding files `example6_29.sg`, in our GitHub and Zenodo repositories.

*Example 6.31.* We apply Algorithm 6.2 to the system

$$\begin{aligned}\dot{x}_1 &= -k_1x_1, \\ \dot{x}_2 &= -k_2x_2, \\ \dot{x}_3 &= (k_1 + k_2)x_1x_2\end{aligned}$$

only considering the generic branch of a comprehensive Gröbner basis, which is given by the semialgebraic set defined by  $k_1 \neq 0$  and  $k_2 \neq 0$ . For the generic branch a basis of the syzygy module is

$$\{(x_2, x_1, 1), (-k_2x_2, k_1x_1, 0)\}.$$

From the above generating set, one obtains 20 syzygies that form a basis for the  $\mathbb{Q}$ -vector space of syzygies up to degree three, out of which only two elements are conservative. One conservative syzygy is  $(x_2, x_1, 1)$ , which is integrated to the unconditional conservation law

$$\phi_1 = x_1x_2 + x_3.$$

The other conservative syzygy

$$(x_1x_2^2 + x_2x_3, x_1^2x_2 + x_1x_3, x_1x_2 + x_3)$$

leads to the degree three conservation law

$$\phi_2 = \frac{1}{2}x_1^2x_2^2 + x_1x_2x_3 + \frac{1}{2}x_3^2.$$

One can easily see that  $\phi_2 = \frac{1}{2}\phi_1^2$ , hence the two conservation laws computed are algebraically dependent. One can check that the second element of the module basis is not conservative, because

$$\nabla \times (-k_2x_2, k_1x_1, 0) = \left( \frac{\partial}{\partial x_2}(-k_2x_2) - \frac{\partial}{\partial x_1}(k_1x_1), 0, 0 \right) \neq 0.$$

Note that

$$\{((k_1 + k_2)x_2, 0, k_1), (-k_2x_2, k_1x_1, 0)\}$$

is another basis of the syzygy module of the generic branch, in which both elements have nonzero curl, but the linear combination of the basis elements

$$((k_1 + k_2)x_2, 0, k_1) + (-k_2x_2, k_1x_1, 0) = k_1(x_2, x_1, 1)$$

has zero curl and it is the only one of degree two and up to multiplication. Computations for all branches of the dynamical system in this example were carried out using our Singular library `polconslaw.lib`. The reader can use `example6_30.sg`, available in our GitHub and Zenodo repositories, in order to carry on the computations automatically.

*Example 6.32* (BIOMD0000000629 [48]). The ODEs corresponding to biomodel 629 in the ODEbase repository [48] are the following:

$$\begin{aligned}\dot{x}_1 &= -k_2x_1x_3 + k_3x_2, \\ \dot{x}_2 &= k_2x_1x_3 - k_3x_2 - k_4x_2x_4 + k_5x_5, \\ \dot{x}_3 &= -k_2x_1x_3 + k_3x_2, \\ \dot{x}_4 &= -k_4x_2x_4 + k_5x_5, \\ \dot{x}_5 &= k_4x_2x_4 - k_5x_5.\end{aligned}$$

Computations for generic branch using our Singular library `polconslaw.lib` can be found in the file `gbm629.txt`, and computations for all branches using the Singular library `genpolconslaw.lib` can be carried on automatically via the file `bm629.sg`, which is available in our GitHub and Zenodo repositories. A CGS for the steady state ideal consists of 10 branches. We consider first the generic branch and after that one nongeneric branch. One easily checks that the semialgebraic set of the generic branch of the comprehensive Gröbner basis consists of  $\{k_2 \neq 0, k_4 \neq 0\}$ , and the corresponding polynomial set consists of the five polynomials in the r.h.s. of the ODE system. The syzygy module has a basis

$$\begin{aligned}\{\mathbf{f}_1 &= (-1, 0, 1, 0, 0), \mathbf{f}_2 = (0, -1, -1, 1, 0), \mathbf{f}_3 = (0, 0, 0, 1, 1), \\ \mathbf{f}_4 &= (k_2x_1x_3 - k_4x_2x_4 - k_3x_2 + k_5x_5, k_2x_1x_3 - k_3x_2, 0, 0, 0)\}.\end{aligned}$$

Obviously,  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$  are conservative, and integrating them leads to the linearly independent unconditional conservation laws

$$\phi_1(\mathbf{k}, \mathbf{x}) = -x_1 + x_3, \phi_2(\mathbf{k}, \mathbf{x}) = -x_2 - x_3 + x_4, \phi_3(\mathbf{k}, \mathbf{x}) = x_4 + x_5.$$

The syzygy  $\mathbf{f}_4(\mathbf{k}, \mathbf{x})$  is not conservative as  $\nabla \times \mathbf{f}_4 \neq 0$ . Multiplying  $\mathbf{f}_1, \dots, \mathbf{f}_4$  with all monomials such that the degree of the products is at most two, we obtain an 18-dimensional  $\mathbb{Q}$ -vectors space of syzygies and find nine conservation laws, where the six newly found degree two conservation laws can be expressed as linear combinations of  $\phi_1(\mathbf{k}, \mathbf{x})$ ,  $\phi_2(\mathbf{k}, \mathbf{x})$  and  $\phi_3(\mathbf{k}, \mathbf{x})$  with coefficients in  $\mathbb{Q}[\mathbf{x}, \mathbf{k}]$ . Enlarging the degree by one gives us a 64-dimensional  $\mathbb{Q}$ -vector space of syzygies. Our algorithm finds here 10 new conservation laws of degree three. Again these conservation laws are in the algebra generated by  $\phi_1(\mathbf{k}, \mathbf{x})$ ,  $\phi_2(\mathbf{k}, \mathbf{x})$ , and  $\phi_3(\mathbf{k}, \mathbf{x})$ . One may assume that it is enough to check the generators of the syzygy module to be conservative and that all further conservation laws can be obtained as a linear combination

of the generators that are conservative over  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . But the last paragraph of Example 6.31 shows that this is not true.

As discussed earlier in Remark 6.28 one can compute unconditional conservation laws from the syzygy module by an additional Gröbner basis computation. On the other hand, unconditional conservation laws up to a given degree bound can be computed via ansatz, i.e., guessing the conservation laws with undetermined coefficients, and then obtaining coefficients by imposing the definition of first integrability. This method has been discussed in [34] for computing polynomial first integrals of dynamical systems and it has been implemented in the automated prover tool Pegasus [63].

Computations based on an ansatz avoid Gröbner basis (and syzygy) computations in the ring  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$ . However, computing polynomial conservation laws via ansatz would not necessarily lead to a basis in  $\mathbb{Q}[\mathbf{k}, \mathbf{x}]$  for the syzygy module, but rather to a basis for the  $\mathbb{R}$ -vector space of conservative syzygies up to a given degree. Moreover, the ansatz method cannot be used to compute branches of parametric conservation laws. Finally, using syzygies brings an algebraic structure to the problem of first integral computation. Since this paper is the first step in this particular research direction, we are confident that there exists ample room for exploration regarding the algebraic characteristics of the conservation laws. In particular, we could anticipate the possibility of finding directly a basis for the algebra of conservation laws, rather than computing a  $\mathbb{Q}$ -vector space basis for syzygies and subsequently filtering them via curl operator.

**6.2.5. Testing syzygy computation on a biochemical models database.** We have systematically tested Algorithm 6.2 using the models of the ODEbase repository [48]. Selected results and some interpretations based on the benchmark table are presented below, whereas the comprehensive report of the results is provided as supplementary materials (M154401\_01.pdf [local/web 364KB]).

In Table 1, we present the result of our computations up to degree four for four biomodels from the ODEbase, including model 629 considered in Example 6.32. Running times are in seconds, with a limit of 4 hours. Columns  $n$  and  $r$  represent the number of variables and parameters in the model, respectively. While computations for model 629 have finished in quite a short time, for models 710 and 1053, we were only able to finish the computations for a few branches, and for model 1054, computations for none of the branches finished in four hours. This is due to several reasons. First of all, comprehensive Gröbner bases are known to be hard to compute, due to the huge number of branches that they may produce. Also syzygy computations can be hard as their computation is equivalent to computing Gröbner bases. Moreover, computing a vector space basis for the syzygies has exponential complexity.

An interesting point in this table is that in several rows, the numbers of linear and polynomial conservation laws are related. This is because most of the polynomial conservation laws computed are in the algebra generated by the linear conservation laws. As noted in Proposition 6.24, conservation laws form an algebra, hence, for algebraically independent conservation laws  $\phi_1, \dots, \phi_m$ , every polynomial of the form  $\phi_1^{n_1} \phi_2^{n_2} \dots \phi_m^{n_m}$ , where  $n_1, \dots, n_m \in \mathbb{N}$  is a conservation law. There are  $\binom{m+d}{d} - 1$  many such polynomials of degree at most  $d$ . For  $d = 4$  and  $m = 1, 2, 3, 4, 5, 6$ , this number is 4, 14, 34, 69, 125, 209, respectively. We obtain exactly those numbers (in the column “# cons. laws” that presents the total number of conservation laws)



**Table 1**  
*Conservation laws in branches of CGS.*

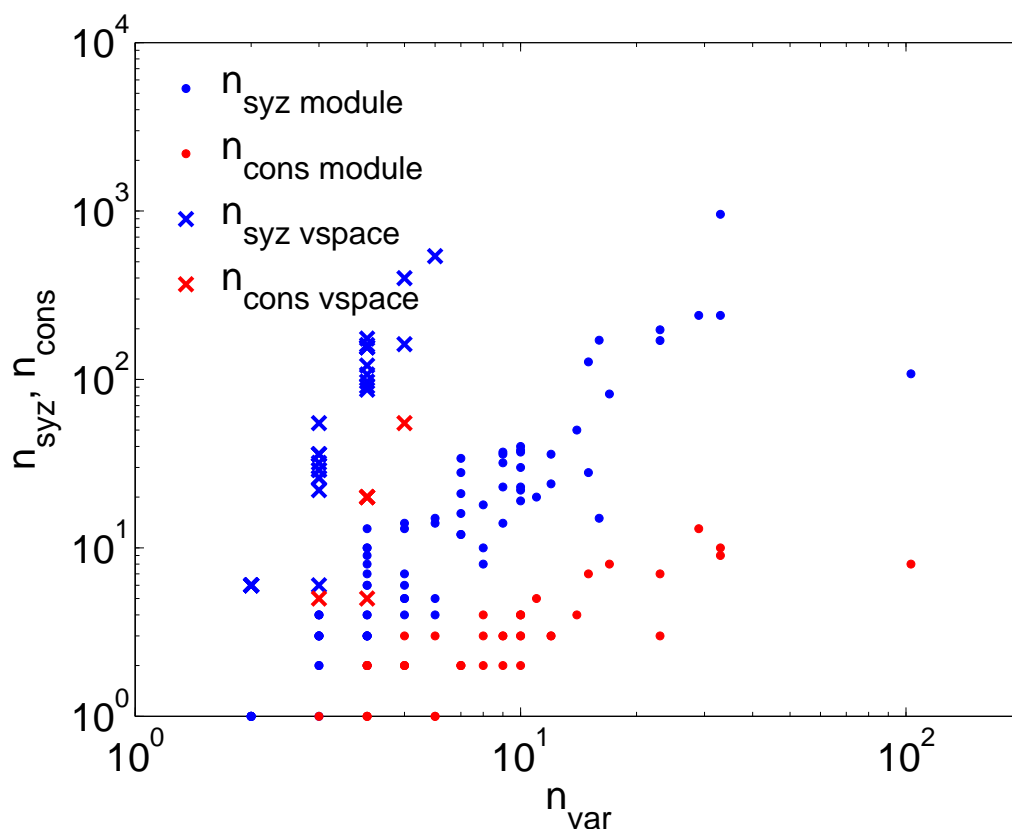
Biomodel	$n$	$r$	Running time	# branches	Branch number	# cons. laws	# lin. cons. laws	# pol. cons. laws
629	5	8	872	10	1	34	3	31
					2	34	3	31
					3	69	4	65
					4	69	4	65
					5	125	5	120
					6	34	3	31
					7	69	4	65
					8	34	3	31
					9	69	4	65
					10	34	3	31
710	7	18	14401	5891	3	4	1	3
					4	4	1	3
					5	14	2	12
					6	14	2	12
					7	34	3	31
					8	34	3	31
					9	34	3	31
					10	69	4	65
					11	125	5	120
					12	125	5	120
1053	6	7	14401	28	3	4	1	3
					4	69	4	65
					5	125	5	120
					6	209	6	203
					7	209	6	203
					8	4	1	3
					9	4	1	3
					10	21	2	19
					11	34	3	31
					12	69	4	65
					13	69	4	65
					15	4	1	3
					19	4	1	3
					21	17	2	15
					22	38	3	35
1054	7	12	14401	1379				

in all rows, except for the gray rows that include more polynomial conservation laws. Those are polynomial conservation laws that are algebraically independent of the linear conservation laws.

Similarly, in Examples 6.30 and 6.31 one can see that the polynomial conservation laws are generated by the lowest degree (not necessarily linear) conservation laws. This suggests that the  $\mathbb{Q}$ -algebra of polynomial conservation laws is finitely generated and that the generators include curl-zero elements of a minimal basis of the syzygy module. Furthermore, the generators other than those coming from a basis of the syzygy module are rare. This subject

**Table 2**  
*Computations for Syzygy module basis (generic branch).*

Biomodel	$n$	$r$	Running time	# syzygy mod. basis	# cons. laws	# lin. cons. laws	# pol. cons. laws
629	5	8	1	4	3	3	0
710	7	18	1	28	0	0	0
1053	6	7	1	15	0	0	0
1054	7	12	1	21	2	2	0



**Figure 1.** *Computation of polynomial conservation laws arising from the generic branch of CGS for CRN models from the BioModel database by two methods: from the generators of the syzygy module and from the basis of  $\mathbb{Q}$ -vector space of syzygies.  $n_{\text{var}}$  is the number of variables and ODEs in the CRN model. The number of syzygies and polynomial conservation laws increases with the number of variables. When the computation of the  $\mathbb{Q}$ -vector space basis succeeds, the number of computed polynomial conservation laws is sensitively larger.*

will be further developed elsewhere.

The discussion above leads us to consider computing only conservation laws that arise from conservative generators of a Gröbner basis of the syzygy module, as a heuristic. Such computations would naturally be much faster than computing a basis for syzygies as a vector space. Moreover, considering only the generic branch and using the trick mentioned in Remark 6.29, one avoids computing a comprehensive Gröbner basis, which further speeds up the

computations. The latter heuristic has been considered in Table 2, carrying on computations for the same biomodels as in Table 1. The computations took only one second per model, and we found two linear conservation laws for model 1054, for which we had run out of time during previous computations.

Figure 1 illustrates how the number of computed syzygies and conservation laws increases with respect to the number of variables of the model. As expected, the  $Q$ -vector space basis calculation fails more often, but when successful it leads to much more polynomial conservation laws, while syzygy module basis calculation produces algebraically independent conservation laws much faster.

**Availability.** ODEbase is available at <https://www.odebase.org/>. The Singular libraries `polconslaw.lib` and `genpolconslaw.lib` are available at Zenodo under the record <https://zenodo.org/record/7474523> and through GitHub at <https://github.com/oradules/polynomial-conservation-laws>.

**Acknowledgments.** We thank the referees for their thoughtful comments that helped us improve our manuscript. OR thanks Werner Seiler for very insightful discussions and for his hospitality at the University of Kassel, where the ideas about approximate conservation laws were initiated. HR would like to thank Deepak Kapur and Amir Hashemi for the discussions about comprehensive Gröbner bases and syzygy computations. The project SYMBIONT owes a lot to Andreas Weber, who sadly left us in 2020, but who is still present in our memories.

## REFERENCES

- [1] E. ÁBRAHÁM, J. ABBOTT, B. BECKER, A. M. BIGATTI, M. BRAIN, B. BUCHBERGER, A. CIMATTI, J. H. DAVENPORT, M. ENGLAND, P. FONTAINE, S. FORREST, A. GRIGGIO, D. KROENING, W. M. SEILER, AND T. STURM, *SC<sup>2</sup>: Satisfiability checking meets symbolic computation*, in Proceedings of C1CM 2016, M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, eds., Lecture Notes in Comput. Sci. 9791, Springer, New York, 2016, pp. 28–43.
- [2] R. ARIS AND R. MAH, *Independence of chemical reactions*, Ind. Eng. Chem. Fundam., 2 (1963), pp. 90–94.
- [3] V. ARNOLD, *Mathematical Methods of Classical Mechanics*, Springer, New York, 1978.
- [4] V. I. ARNOLD, *Ordinary Differential Equations*, Russian ed., Springer-Verlag, Berlin, 1992.
- [5] P. AUGER, R. B. DE LA PARRA, J.-C. POGGIALE, E. SÁNCHEZ, AND L. SANZ, *Aggregation methods in dynamical systems and applications in population and community dynamics*, Phys. Life Rev., 5 (2008), pp. 79–105.
- [6] C. BALLARIN AND M. KAUERS, *Solving parametric linear systems: An experiment with constraint algebraic programming*, ACM SIGSAM Bull., 38 (2004), pp. 33–46.
- [7] C. BARRETT, P. FONTAINE, AND C. TINELLI, *The SMT-LIB Standard: Version 2.6*, Technical report, Department of Computer Science, University of Iowa, 2017.
- [8] T. BECKER, V. WEISPFENNING, AND H. KREDEL, *Gröbner Bases, a Computational Approach to Commutative Algebra*, Grad. Texts in Math. 141, Springer, New York, 1993.
- [9] C. BERKESCH AND F.-O. SCHREYER, *Syzygies, finite length modules, and random curves*, Commut. Algebra Noncommut. Algebraic Geom., 1 (2015), pp. 25–52.
- [10] N. K. BOSE, *Gröbner bases: An algorithmic method in polynomial ideal theory*, in Multidimensional Systems Theory and Applications, Springer Netherlands, Dordrecht, 1995, pp. 89–127, <https://doi.org/10.1007/978-94-017-0275-1>.
- [11] F. BOULIER, M. LEFRANC, F. LEMAIRE, AND P.-E. MORANT, *Model reduction of chemical reaction systems using elimination*, Math. Comput. Sci., 5 (2011), pp. 289–301.
- [12] C. W. BROWN, *QEPCAD B: A program for computing with semi-algebraic sets using CADs*, ACM SIGSAM Bull., 37 (2003), pp. 97–108, <https://doi.org/10.1145/968708.968710>.

- [13] C. W. BROWN AND C. GROSS, *Efficient preprocessing methods for quantifier elimination*, in Computer Algebra in Scientific Computing: 9th International Workshop, CASC 2006, Lecture Notes in Comput. Sci. 4194, V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, eds., Springer, New York, 2006, pp. 89–100.
- [14] B. BUCHBERGER, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Doctoral dissertation, Mathematical Institute, University of Innsbruck, Innsbruck, Austria, 1965.
- [15] V. CHELLIAH, N. JUTY, I. AJMERA, R. ALI, M. DUMOUSSEAU, M. GLONT, M. HUCKA, G. JALOWICKI, S. KEATING, V. KNIGHT-SCHRIJVER, A. LLORET-VILLAS, K. NATH NATARAJAN, J.-B. PETTIT, N. RODRIGUEZ, M. SCHUBERT, S. M. WIMALARATNE, Y. ZHAO, H. HERMJAKOB, N. LE NOVÈRE, AND C. LAIBE, *BioModels: Ten-year anniversary*, Nucl. Acids Res., 43 (2015), pp. D542–D548, <https://doi.org/10.1093/nar/gku1181>.
- [16] G. CHÈZE, *Computation of Darboux polynomials and rational first integrals with bounded degree in polynomial time*, J. Complexity, 27 (2011), pp. 246–262.
- [17] D. A. COX, J. LITTLE, AND D. O’SHEA, *Using Algebraic Geometry*, Grad. Texts in Math. 185, Springer, New York, 2006.
- [18] D. A. COX, J. B. LITTLE, AND D. O’SHEA, *Ideals, Varieties, and Algorithms*, 4th ed., Springer Undergrad. Texts Math., Springer, New York, 2015.
- [19] M. D. DARMIAN, A. HASHEMI, AND A. MONTES, *Erratum to “A new algorithm for discussing Gröbner bases with parameters”* [J. Symbolic Comput., 33 (1–2) (2002) 183–208], J. Symbolic Comput., 46 (2011), pp. 1187–1188, <https://doi.org/10.1016/j.jsc.2011.05.002>.
- [20] W. DECKER, G.-M. GREUEL, G. PFISTER, AND H. SCHÖNEMANN, *Singular 4-2-0—A Computer Algebra System for Polynomial Computations*, 2020, <http://www.singular.uni-kl.de>.
- [21] A. DESOEUVRES, A. IOSIF, C. LÜDERS, O. RADULESCU, H. RAHKOORY, M. SEISS, AND T. STURM, *Reduction of chemical reaction networks with approximate conservation laws*, SIAM J. Appl. Dyn. Syst., 23 (2024), pp. 256–296.
- [22] A. DOLZMANN AND T. STURM, *REDLOG: Computer algebra meets computer logic*, ACM SIGSAM Bull., 31 (1997), pp. 2–9, <https://doi.org/10.1145/261320.261324>.
- [23] A. DOLZMANN AND T. STURM, *Simplification of quantifier-free formulae over ordered fields*, J. Symbolic Comput., 24 (1997), pp. 209–231.
- [24] A. DOLZMANN, T. STURM, AND V. WEISPFENNING, *Real quantifier elimination in practice*, in Algorithmic Algebra and Number Theory, B. H. Matzat, G.-M. Greuel, and G. Hiss, eds., Springer, New York, 1998, pp. 221–247, [https://doi.org/10.1007/978-3-642-59932-3\\_11](https://doi.org/10.1007/978-3-642-59932-3_11).
- [25] D. EISENBUD, *The Geometry of Syzygies: A Second Course in Commutative Algebra and Algebraic Geometry*, Grad. Texts in Math. 229, Springer, New York, 2005.
- [26] M. FEINBERG, *Foundations of Chemical Reaction Network Theory*, Appl. Math. Sci. 202, Springer, New York, 2019, <https://doi.org/10.1007/978-3-030-03858-8>.
- [27] M. FEINBERG AND F. J. HORN, *Dynamics of open chemical systems and the algebraic structure of the underlying reaction network*, Chem. Eng. Sci., 29 (1974), pp. 775–787.
- [28] E. FELIU AND C. WIUF, *Preclusion of switch behavior in networks with mass-action kinetics*, Appl. Math. Comput., 219 (2012), pp. 1449–1467.
- [29] N. FENICHEL, *Geometric singular perturbation theory for ordinary differential equations*, J. Differential Equations, 31 (1979), pp. 53–98.
- [30] S. GAO, F. V. IV, AND M. WANG, *A new framework for computing Gröbner bases*, Math. Comp., 85 (2016), pp. 449–465, <https://doi.org/10.1090/mcom/2969>.
- [31] A. GOEKE, S. WALCHER, AND E. ZERZ, *Classical quasi-steady state reduction—a mathematical characterization*, Phys. D, 345 (2017), pp. 11–26, <https://doi.org/10.1016/j.physd.2016.12.002>.
- [32] L. GOLDMAN, *Integrals of multinomial systems of ordinary differential equations*, J. Pure Appl. Algebra, 45 (1987), pp. 225–240.
- [33] A. N. GORBAN, O. RADULESCU, AND A. Y. ZINOVYEV, *Asymptotology of chemical reaction networks*, Chem. Eng. Sci., 65 (2010), pp. 2310–2324.
- [34] A. GORIELY, *Integrability and Nonintegrability of Dynamical Systems*, Adv. Ser. Nonlinear Dynam. 19, World Scientific, River Edge, NJ, 2001.
- [35] A. HASHEMI, M. D. DARMIAN, AND M. BARKHORDAR, *Gröbner systems conversion*, Math. Comput. Sci., 11 (2017), pp. 61–77, <https://doi.org/10.1007/s11786-017-0295-3>.

- [36] M. HAUKE, *Signature Gröbner Bases: A Comprehensive Survey and a New Algorithmic Approach*, Master's thesis, Graz University of Technology, Graz, Austria, 2020.
- [37] A. C. HEARN, *REDUCE: A user-oriented interactive system for algebraic simplification*, in Proceedings of the Symposium on Interactive Systems for Experimental Applied Mathematics, ACM, 1967, <https://doi.org/10.1145/2402536.2402544>.
- [38] A. C. HEARN, *REDUCE: The first forty years*, in Algorithmic Algebra and Logic: Proceedings of the A3L 2005, BOD, Norderstedt, Germany, 2005, pp. 19–24.
- [39] D. KAPUR, *An approach for solving systems of parametric polynomial equations*, in Principles and Practice of Constraint Programming, Springer, New York, 1995, pp. 217–224.
- [40] D. KAPUR, *Comprehensive Gröbner basis theory for a parametric polynomial ideal and the associated completion algorithm*, J. Syst. Sci. Complex., 30 (2017), pp. 196–233, <https://doi.org/10.1007/s11424-017-6337-8>.
- [41] D. KAPUR, Y. SUN, AND D. WANG, *An efficient method for computing comprehensive Gröbner bases*, J. Symbolic Comput., 52 (2013), pp. 124–142, <https://doi.org/10.1016/j.jsc.2012.05.015>.
- [42] M. KOŠTA, *New Concepts for Real Quantifier Elimination by Virtual Substitution*, Doctoral dissertation, Saarland University, Germany, 2016, <https://doi.org/10.22028/D291-26679>.
- [43] M. KREUZER AND L. ROBBIANO, *Computational Commutative Algebra 1*, Springer, New York, 2008.
- [44] N. KRUFF, C. LÜDERS, O. RADULESCU, T. STURM, AND S. WALCHER, *Algorithmic Reduction of Biological Networks with Multiple Time Scales*, Math. Comput. Sci., 15 (2021), pp. 499–534.
- [45] A. M. LOZANO, *The Gröbner Cover*, Algorithms Comput. Math. 27, Springer, New York, 2018, <https://doi.org/10.1007/978-3-030-03904-2>.
- [46] D. LU, Y. SUN, AND D. WANG, *A survey on algorithms for computing comprehensive Gröbner systems and comprehensive Gröbner bases*, J. Syst. Sci. Complex., 32 (2019), pp. 234–255, <https://doi.org/10.1007/s11424-019-8357-z>.
- [47] C. LÜDERS, T. STURM, AND O. RADULESCU, *ODEbase: A repository of ODE systems for systems biology*, Bioinform. Adv., 2 (2022), vbac027.
- [48] C. LÜDERS, T. STURM, AND O. RADULESCU, *ODEbase: A Repository of ODE Systems for Systems Biology*, CoRR, <http://arxiv.org/abs/2201.08980>, 2022.
- [49] A. MAHDI, A. FERRAGUT, C. VALLS, AND C. WIUF, *Conservation laws in biochemical reaction networks*, SIAM J. Appl. Dyn. Syst., 16 (2017), pp. 2213–2232.
- [50] Y.-K. MAN, *Computing closed form solutions of first order ODEs using the Prelle-Singer procedure*, J. Symbolic Comput., 16 (1993), pp. 423–443.
- [51] L. MICHAELIS AND M. L. MENTEN, *Die kinetik der invertinwirkung*, Biochem. Z., 49 (1913), p. 352.
- [52] A. MONTES, *A new algorithm for discussing Gröbner bases with parameters*, J. Symbolic Comput., 33 (2002), pp. 183–208, <https://doi.org/10.1006/jsc.2001.0504>.
- [53] K. NABESHIMA, *Generic Gröbner basis of a parametric ideal and its application to a comprehensive Gröbner system*, in Proceedings of the Applications of Computer Algebra Conference, 2022, <https://www.rs.tus.ac.jp/nabeshima/ACA2022/2022aca/nabeshima22.pdf>.
- [54] R. NIEUWENHUIS, A. OLIVERAS, AND C. TINELLI, *Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T)*, J. ACM, 53 (2006), pp. 937–977, <https://doi.org/10.1145/1217856.1217859>.
- [55] M. J. PRELLE AND M. F. SINGER, *Elementary first integrals of differential equations*, Trans. Amer. Math. Soc., 279 (1983), pp. 215–229.
- [56] O. RADULESCU, A. N. GORBAN, A. ZINOVYEV, AND V. NOEL, *Reduction of dynamical biochemical reactions networks in computational biology*, Front. Genetics, 3 (2012), 131.
- [57] K. R. SCHNEIDER AND T. WILHELM, *Model reduction by extended quasi-steady-state approximation*, J. Math. Biol., 40 (2000), pp. 443–450.
- [58] F. O. SCHREYER, *A standard basis approach to syzygies of canonical curves*, J. Reine Angew. Math., 421 (1991), pp. 83–124.
- [59] S. SCHUSTER AND T. HÖFER, *Determining all extreme semi-positive conservation relations in chemical reaction systems: A test criterion for conservativity*, J. Chem. Soc. Faraday Trans., 87 (1991), pp. 2561–2566.
- [60] A. SEIDL, *Cylindrical Decomposition under Application-Oriented Paradigms*, Doctoral dissertation, University of Passau, Germany, 2006.

- [61] A. M. SEIDL AND T. STURM, *Boolean quantification in a first-order context*, in Computer Algebra in Scientific Computing. Proceedings of the CASC 2003, V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, eds., Institut für Informatik, Technische Universität München, München, Germany, 2003, pp. 329–345.
- [62] W. Y. SIT, *An algorithm for solving parametric linear systems*, J. Symbolic Comput., 13 (1992), pp. 353–394, [https://doi.org/10.1016/S0747-7171\(08\)80104-6](https://doi.org/10.1016/S0747-7171(08)80104-6).
- [63] A. SOGOKON, S. MITSCH, Y. K. TAN, K. CORDWELL, AND A. PLATZER, *Pegasus: Sound continuous invariant generation*, Formal Methods Syst. Des., 58 (2021), pp. 5–41.
- [64] S. SOLIMAN, *Invariants and other structural properties of biochemical models as a constraint satisfaction problem*, Algorithms Mol. Biol., 7 (2012), pp. 1–9.
- [65] M. SPIVAK, *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*, CRC Press, Boca Raton, FL, 2018.
- [66] T. STURM, *Redlog online resources for applied quantifier elimination*, Acta Acad. Ab., 67 (2007), pp. 177–191.
- [67] T. STURM, *A survey of some methods for real quantifier elimination, decision, and satisfiability and their applications*, Math. Comput. Sci., 11 (2017), pp. 483–502, <https://doi.org/10.1007/s11786-017-0319-z>.
- [68] T. STURM, *Thirty years of virtual substitution*, in Proceedings of ISSAC 2018, ACM, 2018, pp. 11–16.
- [69] A. SUZUKI AND Y. SATO, *An alternative approach to comprehensive Gröbner bases*, J. Symbolic Comput., 36 (2003), pp. 649–667, [https://doi.org/10.1016/S0747-7171\(03\)00098-1](https://doi.org/10.1016/S0747-7171(03)00098-1).
- [70] A. N. TIKHONOV, *Systems of differential equations containing small parameters in the derivatives*, Mat. Sb. (N. S.), 73 (1952), pp. 575–586.
- [71] F. VOLNY, *New Algorithms for Computing Groebner Bases*, Doctoral dissertation, Clemson University, Clemson, NC, 2011.
- [72] J. WEI AND C. D. PRATER, *The structure and analysis of complex reaction systems*, Adv. Catalysis, 13, (1962), pp. 203–392.
- [73] V. WEISPFENNING, *Comprehensive Gröbner bases*, J. Symbolic Comput., 14 (1992), pp. 1–30, [https://doi.org/10.1016/0747-7171\(92\)90023-W](https://doi.org/10.1016/0747-7171(92)90023-W).
- [74] X. ZHANG, *Integrability of Dynamical Systems: Algebra and Analysis*, Dev. Math. 47, Springer, New York, 2017.