Exact linear reduction for rational dynamical systems

Antonio Jiménez-Pastor[*], Joshua Paul Jacob[†], Gleb Pogudin[‡]

Abstract

Detailed dynamical systems models used in life sciences may include dozens or even hundreds of state variables. Models of large dimension are not only harder from the numerical perspective (e.g., for parameter estimation or simulation), but it is also becoming challenging to derive mechanistic insights from such models. Exact model reduction is a way to address this issue by finding a self-consistent lower-dimensional projection of the corresponding dynamical system. A recent algorithm CLUE allows one to construct an exact linear reduction of the smallest possible dimension such that the fixed variables of interest are preserved. However, CLUE is restricted to systems with polynomial dynamics. Since rational dynamics occurs frequently in the life sciences (e.g., Michaelis-Menten or Hill kinetics), it is desirable to extend CLUE to the models with rational dynamics.

In this paper, we present an extension of CLUE to the case of rational dynamics and demonstrate its applicability on examples from literature. Our implementation is available in version 1.5 of CLUE[1].

Keywords: exact reduction, dynamical systems, constrained lumping

This work was supported by the Paris Ile-de-France region (via project "XOR"). GP was partially supported by NSF grants DMS-1853482, DMS-1760448, and DMS-1853650; and AJP was partially supported by Poul Due Jensen Grant 883901.

1 Introduction

Dynamical systems modeling is one of the key mathematical tools for describing phenomena in life sciences. Making such models realistic often requires taking into account a wide range of factors yielding models of high dimension (dozens or hundreds of state variables). Because of their size, these models are challenging from the numerical standpoint (e.g.,

[*] jimenezpastor@lix.polytechnique.fr, LIX, CNRS, École Polytecnique, Institute Polytechnique de Paris, Palaiseau, 91120, France.

[†] joshuapjacob@berkeley.edu, University of California, Berkeley, 94720, California, USA.

[‡] gleb.pogudin@polytechnique.edu, LIX, CNRS, École Polytecnique, Institute Polytechnique de Paris, Palaiseau, 91120, France.

[1] https://github.com/pogudingleb/CLUE

parameter estimation) and it may be hard to derive mechanistic insight from them. One possible workaround is to use *model reduction* techniques that replace a model with a simpler one while preserving, at least approximately, some of the important features of the original model. For approximate model reduction, many powerful techniques have been developed including singular value decomposition [1] and time-scale separation [24].

A complementary approach is to perform *exact model reduction*, that is, lower the dimension of the model without introducing approximation errors. For example, exact linear lumping aims at writing a self-consistent system of differential equations for a set of *macro-variables* in which each macro-variable is a linear combination of the original variables. The case of the macro-variables being sums of the original variables has been studied for some important classes of biochemical models (see, e.g., [5, 13, 17]) and for general rational dynamical systems in [10, 11]. The latter line of research has culminated in the powerful ERODE software [10] which finds the optimal partition of the original variables into macro-variables.

A recent algorithm from [25] (and implemented in the software CLUE) allows, for a given set of linear forms in the state variables (the *observables*), constructs a linear lumping of the smallest possible dimension such that the observables can be written as combinations of the macro-variables (i.e., the observables are preserved). Unlike the prior approaches, the macro-variables produced by CLUE are allowed to involve any coefficients (not just zeroes and ones as before). Thanks to this, one may obtain reductions of significantly lower dimension than it was possible before, see [25, Table 1].

However, unlike ERODE, the algorithm used in CLUE was restricted to the models with polynomial dynamics. Since rational dynamics appears frequently in life sciences (e.g., via Michaelis-Menten or Hill kinetics), it is desirable to extend CLUE to such models. The goal of the present paper is to fill this gap. We design and implement an algorithm computing an optimal linear lumping of a rational dynamical system given a set of observables. We demonstrate the efficiency and applicability of this algorithm on several models from literature. Our new algorithm is available in CLUE starting from version 1.5 at `https://github.com/pogudingleb/CLUE`.

In fact, we present two algorithms: one is a relatively direct generalization of the original algorithm from [25] which works well if there are only few different denominators in the model, and another is a randomized algorithm based on evaluation-interpolation techniques which can efficiently handle models with multiple denominators (see Table 4 for comparison). Both algorithms are based on the same key mathematical fact, going back to [20], that linear lumpings are in a bijective correspondence with the invariant subspaces of the Jacobian $J(\mathbf{x})$ of the system [25, Proposition II.1]. However, the construction of such invariant subspaces as common invariant subspaces of the coefficients of $J(\mathbf{x})$ used in [25] relies on the polynomiality of the ODE model. The workaround used in the first algorithm is to make $J(\mathbf{x})$ a polynomial matrix by multiplying the common denominator if the entries. However, if there is more than a couple of denominators, the size of the intermediate expressions becomes prohibitive. Therefore, in the second algorithm, we used a different approach to replace a non-constant matrix $J(\mathbf{x})$ with a collection of constant matrices: we replace $J(\mathbf{x})$ by its evaluations at several points which can be exactly computed by means of automatic differentiation without writing down $J(\mathbf{x})$ itself. Since the algorithm is sampling-based, it becomes probabilistic, and we guarantee that the result will be correct with a user-specified probability and also provide a possibility to perform a post-verification of the result.

The rest of the paper is organized as follows. Section 2 contains the preliminaries on lumping and summarizes the main steps of the algorithms from [25] for polynomial dynamics. In Section 3, we describe our two algorithms mentioned above. Section 4 described our implementation of the algorithms and reports its runtimes. In Section 5, we illustrate how our algorithms can be applied to models from the literature. We conclude in Section 6. Some proofs omitted in the main text are collected in the Appendix.

2 Preliminaries and prior results

2.1 Preliminaries on lumping

In this paper we study models defined by ODE systems of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ are the state variables and $\mathbf{f} = (f_1, \ldots, f_n)^T$ with $f_1, \ldots, f_n \in \mathbb{R}(\mathbf{x})$ (where $\mathbb{R}(\mathbf{x})$ denotes the set of rational function in $\mathbf{x}$ with real coefficients). We will describe lumping and the related notions following [25, Section 2] but extending to the case of rational dynamics.

Definition 1 (Lumping)**.** Consider a system of the form (1). A linear transformation $\mathbf{y} = L\mathbf{x}$, where $\mathbf{y} = (y_1, \ldots, y_m)^T$ and $L \in \mathbb{R}^{m\times n}$, is called *lumping* if $\operatorname{rank}(L) = m$ and there exist rational functions $g_1, \ldots, g_m \in \mathbb{R}(\mathbf{y})$ such that

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}), \quad \text{where} \quad \mathbf{g} = (g_1, \ldots, g_m)^T$$

for every solution $\mathbf{x}$ of (1). We call m the *dimension* of the lumping, and we will refer to $\mathbf{y}$ as *macro-variables.*

In other words, a lumping of dimension m is a linear change of variables from n to m variables such that the new variables satisfy a self-contained ODE system. In the language of differential geometry, a lumping is a linear map φ from the state space such that there exists a vector field on the range of the map which is φ-related to $\mathbf{f}$ (see [33, Definition 1.54]).

Example 2. Consider the following differential system:

$$\begin{cases} \dot{x}_1 &= \dfrac{x_2^2 + 4x_2x_3 + 4x_3^2}{x_1^3 - x_2 - 2x_3}, \\ \dot{x}_2 &= \dfrac{4x_3 - 2x_1}{x_1 + x_2 + 2x_3}, \\ \dot{x}_3 &= \dfrac{x_1 + x_2}{x_1 + x_2 + 2x_3}, \end{cases}$$

Then the matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{pmatrix},$$

is a lumping of dimension 2, since:

$$\begin{cases} \dot{y}_1 &= \dot{x}_1 = \dfrac{(x_2 + 2x_3)^2}{x_1^3 - x_2 - 2x_3} = \dfrac{y_2^2}{y_1^3 - y_2} \\ \dot{y}_2 &= \dot{x}_2 + 2\dot{x}_3 = \dfrac{2x_2 + 4x_3}{x_1 + x_2 + 2x_3} = \dfrac{2y_2}{y_1 + y_2}. \end{cases}$$

One way to force a lumping to keep the information of interest is to fix a set of observables to be preserved. This leads to the notion of *constrained lumping.*

Definition 3 (Constrained lumping)**.** Let $\mathbf{x}_{obs}$ be a vector of linear forms in $\mathbf{x}$ (i.e., there is a matrix $M \in \mathbb{R}^{p\times n}$ with $\mathbf{x}_{obs} = M\mathbf{x}$). We say that a lumping L of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ is a *constrained lumping* with observables $\mathbf{x}_{obs}$ if each entry of $\mathbf{x}_{obs}$ can be expressed as a linear combination of $\mathbf{y} = L\mathbf{x}$.

Remark 4 (On the constrained and partition-based lumpings)*.* Software ERODE [9] can efficiently produce the minimal (in the sense of the dimension) lumping, in which the macro-variables correspond to a partition of the state variables. This means that $y_1 = \sum_{i\in S_1} x_i, \dots, y_m = \sum_{i\in S_m} x_i$ such that $\{1,\dots,n\} = S_1 \bigsqcup S_2 \bigsqcup \dots \bigsqcup S_m$.

In this case, we always have $y_1 + \dots + y_m = x_1 + \dots + x_n$, hence a lumping found by ERODE will be always a constrained lumping with the sum of the state variables being the only observable. Therefore, an algorithm (like the one proposed in this paper) for computing a constrained linear lumping of the smallest possible dimension will be always able to find either the lumping found by ERODE or even lump it further. To be fair, we would like to point out that ERODE is typically faster than our algorithm.

Remark 5 (Lumping via polynomialization)*.* It is known that, by introducing new variables, a rational dynamical system can always be embedded into a polynomial one. Therefore, a natural approach to finding constrained linear lumpings for (1) would be to combine such an embedding with any algorithm applicable to polynomial dynamics (e.g., CLUE). However, it has been demonstrated in [11, p. 149] that such an embedding not only increases the dimension of the ambient space but also may miss some of the reductions of original model.

2.2 Overview of the CLUE algorithm for polynomial dynamics [25]

The algorithm for computing a constrained lumping of the minimal dimension for a given system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ with polynomial right-hand side and observables $\mathbf{x}_{obs}$, designed and implemented in [25], can be summarized as follows:

Algorithm 1 Finding constrained linear lumping (polynomial case)

```
Input: a polynomial ODE system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ of dimension $n$; a list of observables
    $\mathbf{x}_{obs} = A\mathbf{x}$ for $A \in \mathbb{R}^{s\times n}$.
Output: minimal lumping $L$ containing $\mathbf{x}_{obs}$.
  (1) Compute the Jacobian matrix $J(\mathbf{x})$ of $\mathbf{f}$;
  (2) Write $J(\mathbf{x})$ as $J_1 m_1 + \dots + J_N m_N$, where $m_1, \dots, m_N$ are distinct monomials
      in $\mathbf{x}$ and $J_1, \dots, J_N$ are constant matrices;
  (3) Compute the minimal subspace $V$ of the space of linear forms in $\mathbf{x}$ containing
      $\mathbf{x}_{obs}$ and invariant under $J_1, \dots, J_N$ (using [25, Alg. 3 or 4]);
  (4) Return matrix $L$ with rows being basis vectors of $V$.
```

This algorithm is based on the criterion from [20], which states that a matrix L is a lumping for the system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ if an only if the row space of L is invariant under $J(\mathbf{x})$

for every value of $\mathbf{x}$. In order to use this criterion, it was shown in [25, Supplementary Materials, Lemma I.1], that L is a lumping for $\dot{x} = \mathbf{f}(\mathbf{x})$ if and only if the row space of L is invariant under J_i for $1 \leqslant i \leqslant N$ (as defined in Step **(2)** of Algorithm 1). This reduces the problem of finding the lumping to the one solved in Step **(3)** of Algorithm 1.

3 Algorithm for rational dynamical systems

In this section, we will use the following "finite" version of the Jacobian-based criterion from [20] which is proved in the Appendix.

Lemma 6. *Consider the rational dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and $J(\mathbf{x})$ the Jacobian matrix of $\mathbf{f}(\mathbf{x})$. Let $\mathcal{B}$ be any set of matrices spanning the vector space $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. Then $L \in \mathbb{R}^{r\times n}$ is a lumping if and only if the row space of L is invariant with respect to all $J \in \mathcal{B}$.*

Remark 7 (Lemma 6 and the polynomial case). In the context of Algorithm 1, each value of $J(\mathbf{x})$ is a linear combination of $J_1, \ldots, J_N$, so $\mathcal{B}$ can be taken to be $\{J_1, \ldots, J_N\}$. Thus Lemma 6 implies the correctness of Algorithm 1.

3.1 Straightforward extension of Algorithm 1

In the case of rational dynamics, the Jacobian matrix $J(\mathbf{x})$ has only rational function entries, so we can compute the common denominator $q(\mathbf{x})$ such that $J(\mathbf{x}) = \frac{B(\mathbf{x})}{q(\mathbf{x})}$, where $B(\mathbf{x})$ is a matrix with polynomial entries. Let $B(\mathbf{x}) = B_1 m_1 + \ldots + B_N m_N$ be a decomposition where $B_1, \ldots, B_N$ are constant matrices and $m_1, \ldots, m_N$ are distinct monomials appearing in $B(\mathbf{x})$ (compare with Step **(2)** of Algorithm 1). Then, for each value of $\mathbf{x}$ not annihilating q, $J(\mathbf{x})$ is a linear combination of $B_1, \ldots, B_N$, so one can take $\mathcal{B} = \{B_1, \ldots, B_N\}$ in Lemma 6. This yields the following algorithm for rational case:

Algorithm 2 Finding constrained linear lumping (rational case)

Input: a *rational* ODE system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ of dimension n; a list of observables $\mathbf{x}_{obs} = A\mathbf{x}$ for $A \in \mathbb{R}^{s\times n}$.

Output: minimal lumping L containing $\mathbf{x}_{obs}$.

```
(1) Compute $J(\mathbf{x})$, the Jacobian of $f(\mathbf{x})$;
(2.1) Compute $p(\mathbf{x})$, the common denominator of the entries of $J(\mathbf{x})$
(2.2) Set $B(\mathbf{x}) := p(\mathbf{x}) \cdot J(\mathbf{x})$
(2.3) Write $B(\mathbf{x})$ as $B_1 m_1 + \ldots + B_N m_N$, where $m_1, \ldots, m_N$ are distinct monomials and $B_1, \ldots, B_N$ are constant matrices.
(3) Compute the minimal subspace $V$ of the space of linear forms in $\mathbf{x}$ containing $\mathbf{x}_{obs}$ and invariant under $B_1, \ldots, B_N$ (using [25, Alg. 3 or 4]);
(4) Return matrix $L$ with rows being basis vectors of $V$;
```

However, the size of the expressions involved after bringing everything to the common denominator and differentiation can be prohibitively large. The following toy example illustrates this phenomenon (for a comparison with the more refined Algorithm 3, see Section 4.2).

Example 8. Consider the following differential system:

$$\dot{x} = \frac{y - z}{x - y}, \quad \dot{y} = \frac{x + z}{x + y}, \quad \dot{z} = \frac{x + y + z}{z - x - y}.$$

Here is the Jacobian matrix:

$$J(x, y, z) = \begin{pmatrix} -\frac{y-z}{(x-y)^2} & \frac{y-z}{(x+y)^2} & \frac{2z}{(x+y-z)^2} \\ -\frac{x-z}{(x-y)^2} & -\frac{x+z}{(x+y)^2} & \frac{2z}{(x+y-z)^2} \\ -\frac{-1}{x-y} & \frac{1}{x+y} & \frac{-2(x+y)}{(x+y-z)^2} \end{pmatrix}$$

The polynomial $p(\mathbf{x})$ from Algorithm 2 will be equal to $(x - y)^2(x + y)^2(x + y - z)^2$. Then the matrix $B(x, y, z)$ will satisfy

$$J(x, y, z) = \frac{1}{(x - y)^2(x + y)^2(x + y - z)^2} B(x, y, z).$$

In this example, the matrix $B(x, y, z)$ has all entries of degree 5 that we need to expand which is substantially more complicated than the original expressions.

3.2 The main algorithm based on evaluation-interpolation

Our main algorithm is built upon the following observation: we can evaluate $J(\mathbf{x})$ efficiently at a given point without writing down the symbolic matrix explicitly (which would be quite large even in small examples such as Example 8), using automatic differentiation techniques. Then we can use sufficiently many such evaluations to span the space $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n \text{ and } J(\mathbf{x}) \text{ is well-defined}\rangle$ from Lemma 6. The resulting Algorithm 3 is shown below. It relies on Algorithm 4 for generating "sufficiently many" evaluations which we present in Section 3.3.

Algorithm 3 Finding constrained linear lumping (probabilistic)

Input: a rational ODE system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ of dimension n; a list of observables $\mathbf{x}_{obs} = A\mathbf{x}$ for $A \in \mathbb{R}^{s \times n}$; and a real number $\varepsilon \in (0, 1)$.

Output: minimal lumping L containing $\mathbf{x}_{obs}$.
The result is correct with probability at least $1 - \varepsilon$.

```
(1) Compute $J(\mathbf{x})$, the Jacobian of $f(\mathbf{x})$.
(2) Compute points $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{Q}^n$ such that $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_M)$ span $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$ with probability at least $1 - \varepsilon$ (using Algorithm 4).
(3) Compute the minimal subspace $V$ of linear forms in $\mathbf{x}$ containing $\mathbf{x}_{obs}$ and invariant under $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_M)$ (using [25, Alg. 3 or 4]);
(4) Return matrix $L$ with rows being basis vectors of $V$;
```

Remark 9 (On probabilities in algorithms). The outputs of Algorithm 3 and 4 are guaranteed to be correct with a user-specified probability. This should be understood as follows. The algorithm makes some random choices, that is, draws a point from a probability space. The specification of the algorithm means that, for the fixed input, the probability of the output being correct (with respect to the probability space above) is at least $1-\varepsilon$.

In the highly unlikely case (the used probability bounds are quite conservative) the computation at Step **(2)** of Algorithm 3 was incorrect, the computed space V will be a subspace of the "true" V, so the resulting matrix will not provide a lumping, and the reduced model returned by the software will be incorrect. The returned model can be checked using a direct substitution. This substitution is not performed in our implementation by default, but we offer a method to gain full confidence in the result.

Proposition 10. *Algorithm 3 is correct, that is the returned matrix L is a minimal lumping with probability at least $1-\varepsilon$ (see Remark 9).*

Proof. Assume that we have found points $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{R}^n$ such that the evaluations $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_M)$ span $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$, and consider L, a matrix with the rows being basis vectors of V, computed using [25, Algorithm 3 or Algorithm 4]. Then it will be a lumping by Lemma 6.

If that is not the case, then it means that $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_M)$ do not span $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. This event only happens with probability at most ε. Hence the algorithm is correct with at least probability $1-\varepsilon$. $\square$

3.3 Generating "sufficiently many" evaluations

In order to complete Algorithm 3, we will present in this section a procedure (Algorithm 4) for sampling values of $J(\mathbf{x})$ spanning the whole $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$ with high probability. The main theoretical tool to achieve the desired probability is the following proposition (proved in the appendix) based on the Schwartz-Zippel lemma [36, Proposition 98].

Proposition 11. *Let $\mathbf{f} = (f_1, \ldots, f_n)^T$ be a vector of elements of $\mathbb{R}(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)^T$, and $\varepsilon \in (0,1)$ be a real number. Let D_d (resp., D_n) be an integer such that the degree of the denominator (resp., numerator) of f_i does not exceed D_d (resp., D_n) for every $1 \leqslant i \leqslant n$.*

Let $J(\mathbf{x})$ be the Jacobian matrix of $\mathbf{f}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_m$ be points such that $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)$ do not span $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. Consider a point $\mathbf{x}_{m+1}$ with each coordinate being an integer sampled uniformly at random from $\{1, 2, \ldots, N\}$ where

$$N > \frac{D_n + (2m+1)D_d}{\varepsilon} + nD_d.$$

Then we have

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)\rangle \mid J(\mathbf{x}_{m+1}) \textit{ is well-defined}] < \varepsilon.$$

Proposition 11 tells us that, when the points are sampled from a large enough range, if a value of $J(\mathbf{x})$ belongs to the space spanned by the previous evaluations, then these evaluations span, with high probability, the whole space $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. This yields the following algorithm.

Algorithm 4 Sampling the values of the Jacobian

Input:

- n-dimensional vector $\mathbf{f}$ of rational functions in $\mathbf{x} = (x_1, \ldots, x_n)$;
- real number $\varepsilon \in (0, 1)$.

Output: Points $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{Q}^n$ such that

$$\langle J(\mathbf{x}_i) \mid 1 \leqslant i \leqslant n\rangle = \langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n \text{ and } J(\mathbf{x}) \text{ is well-defined}\rangle, \tag{2}$$

with probability at least $1-\varepsilon$, where $J(\mathbf{x})$ is the Jacobian matrix of $\mathbf{f}$.

```
(1) Compute $D_d$ (resp., $D_n$) as the maximum of the degrees of denominators (resp.,
    numerators) of entries of $\mathbf{f}$, respectively.
(2) $M \leftarrow 0$
(3) Repeat
    (a) Compute $\mathbf{x}_{M+1}$ by sampling each coordinate uniformly at random from
        $\{1, 2, \ldots, N\}$, where
            $N = \left\lceil \frac{D_n + (2M+1)D_d}{\varepsilon} + nD_d \right\rceil + 1.$
        Repeat sampling until none of the denominators of $\mathbf{f}$ vanishes at $\mathbf{x}_{M+1}$.
    (b) Compute $J(\mathbf{x}_{M+1})$ using automatic differentiation (see Section 3.4).
    (c) If $J(\mathbf{x}_{M+1}) \in \langle J(\mathbf{x}_i) \mid 1 \leqslant i \leqslant M\rangle$, return $\mathbf{x}_1, \ldots, \mathbf{x}_M$.
    (d) $M \leftarrow M + 1$
```

Proposition 12. *Algorithm 4 is correct, that is, for the returned points* $\mathbf{x}_1, \ldots, \mathbf{x}_M$ *the relation* (2) *holds with probability at least* $1 - \varepsilon$ *(see Remark 9).*

Proof. Assume that the output condition (2) for Algorithm 4 does not hold. Since the algorithm has returned, the value $J(\mathbf{x}_{M+1})$ belonged to the span of $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_M)$. Then Proposition 11 implies that the probability of this event is at most ε, so the output of the algorithm is correct with a probability of at least $1-\varepsilon$. □

3.4 Improving the efficiency of Algorithm 3

Evaluating the Jacobian. An attractive feature of Algorithm 3 is that it does not require a symbolic expression for the Jacobian $J(\mathbf{x})$ but only a way to efficiently evaluate it at chosen points. These evaluations can be preformed efficiently using exact automatic differentiation [4, 15, 35, 18].

More precisely, our implementation uses a forward algorithm for automatic differentiation by computing with the extended dual numbers. Extended dual numbers are tuples $(a_0, a_1, \ldots, a_n)$ of real numbers (where n is the dimension of the model) with the following arithmetic rules:

$$\begin{aligned}(a_0, a_1, \ldots, a_n) + (b_0, b_1, \ldots, b_n) &= (a_0 + b_0, \ldots, a_n + b_n),\\ (a_0, a_1, \ldots, a_n)(b_0, b_1, \ldots, b_n) &= (a_0 b_0, a_1 b_0 + a_0 b_1, \ldots, a_0 b_n + a_n b_0).\end{aligned}$$

If one evaluates a rational function $f(\mathbf{x})$ at the point

$$((x_1, 1, 0, \ldots, 0), (x_2, 0, 1, \ldots, 0), \ldots, (x_n, 0, 0, \ldots, 1)), \tag{3}$$

the output will be (see [18, Section 4]) precisely the vector

$$\left(f(\mathbf{x}), \frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x})\right).$$

Thus, we can evaluate $J(\mathbf{x})$ by evaluating $f_1, \ldots, f_n$ at (3). The complexity of such evaluation is directly related to the cost of evaluating $f_1, \ldots, f_n$. If evaluating each of $f_i(\mathbf{x})$ costs A operations, then evaluating the whole $J(\mathbf{x})$ has cost $\mathcal{O}(n^2 A)$. There are techniques (based on backward mode automatic differentiation) that could compute these evaluations within $\mathcal{O}(nA)$ operations [3]. For the moment, we decided to stick to the forward mode due to its simplicity and generalizability.

Selecting starting evaluation points. The sampling strategy for the evaluation points $\mathbf{x}_1, \ldots, \mathbf{x}_M$ at Step **(3)** of Algorithm 4 derived from Proposition 11 is used to ensure (with the prescribed probability) that we do not stop sampling too early. Therefore, before going into Step **(3)**, we can start with choosing several evaluation points $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ in a different way than it is described in Step **(3)**, and the probability of correctness will be preserved.

More precisely, before starting to sample evaluation points as in Step **(3)**, our implementation of Algorithm 4 proceeds as follows:

1. *Sparse points.* For every $1 \leqslant i \leqslant n$, we do the following. Let $\mathbf{e}_1$ be the i-th standard basis vector in $\mathbb{R}^n$. We consider the point $\mathbf{e}_i$ and analyze the variables that need to be made different from zero so that the Jacobian $J(\mathbf{x})$ will be well-defined. We do this modifications in $\mathbf{e}_i$ and obtain (typically sparse) evaluation point.

2. *"Small points".* Then we consider randomly sampled evaluation points but sample them from a small range until we detect a linear dependence between the evaluations of the Jacobian at already sampled points.

These evaluation points yield simpler evaluations of the Jacobian $J(\mathbf{x})$ which simplifies further computations. The optimization has led to significant speedup, see Table 1 (the models used in the table are described in Section 4.1).

Model	**Speedup**	**Model**	**Speedup**
BIOMD0000000013	4.72	*Section 5.1* ($n = 4$)	5.31
BIOMD0000000023	2.35	*Section 5.1* ($n = 5$)	31.45
BIOMD0000000033	1.92	*Section 5.1* ($n = 6$)	> 100
MODEL1502270000	8.98		

Table 1: Speedup through a refined choice of the evaluation points

4 Implementation and performance

We implement both our algorithms described in Section 3 in the software CLUE [25]. This software is written in Python and before it allowed to compute optimal constrained lumpings for any polynomial dynamical system. We have extended the functionality to the case of rational dynamics (starting with version 1.5). CLUE is available on the following GitHub repository:

https://github.com/pogudingleb/CLUE

and can be installed using `pip` directly from GitHub using the command line

```
pip install git+https://github.com/pogudingleb/CLUE
```

We refer to the README and tutorial in the repository for further details on how to use the software.

4.1 Performance of Algorithm 3

In this section, we report the runtimes for our implementation of our main algorithm, Algorithm 3. The timings reported below were measured on a laptop with Intel i7-9850H, 16GB RAM, and Python 3.8.10. The runtimes are average through 5 independent executions on each model. We have measured runtimes for two sets of models

- Several models with rational dynamics from the BioModels database [22] in Table 2. For each model, the table contains the name in the database and a reference to the related paper. The observables in Table 2 were chosen as the states yielding nontrivial reductions, further analysis of these reductions is a question of future research.
- The models we use as examples in this paper (see Section 5) in Table 3.

In these examples, we have fixed the probability bound $\varepsilon = 0.01$. Changing this value to 0.001 may affect the timings measured, although its effect very moderate (less than 3% in all models). In both tables, we also include the observable used in the lumping and the change of the dimension (in the format *before* $\to$ *after*).

Name	Reference	Obs.	Size	Time (min.)
BIOMD0000000013	[28]	*x_CO2*	$28 \to 25$	2.19
BIOMD0000000023	[30]	*Fru*	$13 \to 11$	0.04
BIOMD0000000113	[14]	*Y*	$20 \to 12$	0.01
BIOMD0000000182	[23]	*AC_cyto_mem*	$45 \to 14$	44.1
BIOMD0000000313	[29]	*IL13_DecoyR*	$35 \to 5$	0.09
BIOMD0000000448	[7]	*mTORC1a*	$67 \to 48$	1.37
BIOMD0000000526	[19]	*DISC*	$32 \to 19$	0.1
MODEL1502270000	[34]	*rmr*	$46 \to 45$	56.44

Table 2: Execution time for Algorithm 3 for models from BioModels

Example	Obs.	Size	Time (min.)
Section 5.2	freeEGFReceptor	$80 \to 5$	0.25
Section 5.1, $n = 6$	x_1	$6 \to 2$	0.02
Section 5.1, $n = 7$	x_1	$7 \to 2$	0.04
Section 5.1, $n = 8$	x_1	$8 \to 2$	0.17
Section 5.1, $n = 9$	x_1	$9 \to 2$	0.33
Section 5.1, $n = 10$	x_1	$10 \to 2$	3.74

Table 3: Execution time for Algorithm 3 for the examples from Section 5

For all the models in Table 2, the optimal reduction found by the algorithm is not of the type that could be found by ERODE, so the reduction by ERODE would be of higher dimension (but faster to compute). For example, the reduction found by ERODE for BIOMD0000000182 would be of dimension 33 while we reduce further to 14. We leave a more detailed comparison of CLUE and ERODE for future research.

From the timings above one can see that the complexity of the model for our algorithm is not solely determined by its order but rather by its structure: compare, for example, BIOMD0000000013 and BIOMD0000000448.

We have analyzed the breakdown of the total runtimes for these examples and observed that the most time-consuming part is Algorithm 4 while reading the models and computing the actual lumping typically take less than 1% of the full computation time. Since the sampled matrices depend only on the model, not on the observables, we enhanced CLUE with caching of the matrices so that one can reuse them if wants to check several different sets of observables for a single model. In this case, every subsequent computation will be much faster ($\sim$ 100 times) than reported in the table.

4.2 Comparing Algorithm 2 and Algorithm 3

In our implementation, both Algorithms 2 and 3 are available. For rational dynamical systems, the latter (probabilistic) is used by default. For the polynomial systems, the original algorithm from CLUE is used.

Table 4 contains the runtimes of Algorithms 2 and 3 for several biological models and, for each model, we count the number of distinct denominators appearing in the right-hand side of the ODE system. We separate the polynomial systems that have been taken from the paper [25] from the rational systems we studied in this paper. Note that, for the polynomial systems, Algorithm 3 is essentially the original algorithm from CLUE. One can observe that, if there is only a couple of denominators, then Algorithm 2 may be much faster but once the number of denominators grows, Algorithm 3 outperforms it substantially. In the future, it would be interesting to determine an algorithm to use on the fly.

Polynomial models	Reference	# denoms	Algorithm 2	Algorithm 3
Barua	[2]	0	1.1913	> 30
OrderedPhosphorylation	[6]	0	0.0131	> 30
MODEL1001150000	[26]	0	0.0116	> 30
MODEL8262229752	[21]	0	0.0005	0.0262
fceri_fi	[16]	0	0.0494	> 30
ProteinPhosphorylation (7)	[32]	0	3.433	> 30
Rational models				
BIOMD0000000526	[19]	1	0.0222	0.0972
BIOMD0000000448	[7]	2	0.0387	1.3719
BIOMD0000000313	[29]	2	0.0111	0.0881
BIOMD0000000113	[14]	4	0.4914	0.0073
BIOMD0000000013	[28]	10	> 30	2.1906
BIOMD0000000023	[30]	11	> 30	0.0435
BIOMD0000000033	[8]	22	> 30	2.3012

Table 4: Execution times for Algorithms 2 and 4 (in minutes)

5 Examples

5.1 Michaelis-Menten kinetics with competing substrates

Consider an enzymatic reaction with a single enzyme E and multiple competing substrates $S_1, \ldots, S_n$ (and the corresponding products $P_1, \ldots, P_n$). The reaction can be described by the following chemical reaction network

$$S_i + E \underset{k_{i,-1}}{\overset{k_{i,1}}{\rightleftharpoons}} SE \xrightarrow{k_{i,2}} P_i \quad \text{for } \; i = 1, \ldots, n.$$

Then quasi-steady-state approximation can be used to deduce a system of ODEs for the substrate concentrations only. For n competing substrates, using the general approach due to [12] (for competing substrates, see also [31, Section 3]), we obtain the following ODE system

$$\dot{x}_i = \frac{a_i x_i}{1 + \sum\limits_{j=1}^{n} \frac{x_j}{K_j}} \quad \text{for } i = 1, \ldots, n, \tag{4}$$

where x_i is the concentration of S_i, $K_i = \frac{k_{i,-1}+k_{i,2}}{k_{i,1}}$, $a_i = \frac{k_{i,2}E_0}{K_i}$, and E_0 is the total enzyme concentration. Assume that we are interested in the dynamics of a particular substrate, say S_1. We will now analyze possible constrained lumpings of the system (4) depending on the relations between the parameters (K_i's and a_i's). If a_i's are arbitrary distinct numbers or distinct symbolic parameters, our algorithm shows that there are no nontrivial reductions. However, if some of a_i's are equal, the situation becomes more interesting.

- *Simplest case:* $a_2 = a_3 = \ldots = a_n$. In this case, independently from n (checked for $n \leqslant 10$), the algorithm produces the system

$$\begin{cases} y_1' = \frac{a_1 y_1}{1 + \frac{y_1}{K_1} + y_2}, \\ y_2' = \frac{a_2 y_2}{1 + \frac{y_1}{K_1} + y_2}, \end{cases} \quad \text{where} \quad \begin{cases} y_1 = x_1, \\ y_2 = \sum\limits_{i=2}^{n} \frac{y_i}{K_i}. \end{cases}$$

- *More general case: some of a_i's are equal.* For example, if $n = 6$ and we have $a_2 = a_3$ and $a_4 = a_5 = a_6$, the optimal reduction produced by the algorithm will be

$$\begin{cases} y_1' = \frac{a_1 y_1}{1 + \frac{y_1}{K_1} + y_2 + y_3}, \\ y_2' = \frac{a_2 y_2}{1 + \frac{y_1}{K_1} + y_2 + y_3}, \\ y_3' = \frac{a_4 y_3}{1 + \frac{y_1}{K_1} + y_2 + y_3}, \end{cases} \quad \text{where} \quad \begin{cases} y_1 = x_1, \\ y_2 = \frac{x_2}{K_2} + \frac{x_3}{K_3}, \\ y_3 = \frac{x_4}{K_4} + \frac{x_5}{K_5} + \frac{x_6}{K_6}. \end{cases}$$

  More generally, if several of a_i's are equal, the corresponding x_i's are lumped together with the coefficients $\frac{1}{K_i}$, and the reduced model defines again the Michaelis-Menten kinetics for competing substrates. Interestingly, the substrates can be lumped together if the corresponding a_i's are equal but not all the $k_{i,1}, k_{i,-1}, k_{i,2}$ (cf. the scaling transformation in [31, p. 161]).

The reduction above cannot be found by ERODE [9] unless the $K_i = K_j$ whenever $a_i = a_j$ since the coefficients are not only ones and zeros. Note also that we treat all the parameters symbolically (instead simulating them as states with zero derivatives) so that they can appear in the coefficients of the lumping. To the best of our knowledge, CLUE is the only lumping software with this feature.

Table 3 in Section 4 reports the runtimes for different values of n.

5.2 Nerve growth factor signaling

Motivated by the study of differentiation of neuronal cells, Brown et al [8] considered a model describing the actions of nerve growth factor (NGF) and mitogenic epidermal growth factor (EGF) in rat pheochromocytoma (PC12) cells. In the model, these factors stimulated extracellular regulated kinase (Erk) phosphorylation with distinct dynamical profiles via a network of intermediate signaling proteins, the network is shown in Figure 1. Each intermediate protein (and Erk) is modeled using two species: active and inactive states. The resulting model is described by a system of 32 differential equations with 48 parameters, the full system can be found in [8, Supplementary materials] or as BIOMD0000000033 in the BioModels database [22]. The exact reduction of this model has been earlier studied in [27, Section 5.4] using ERODE.

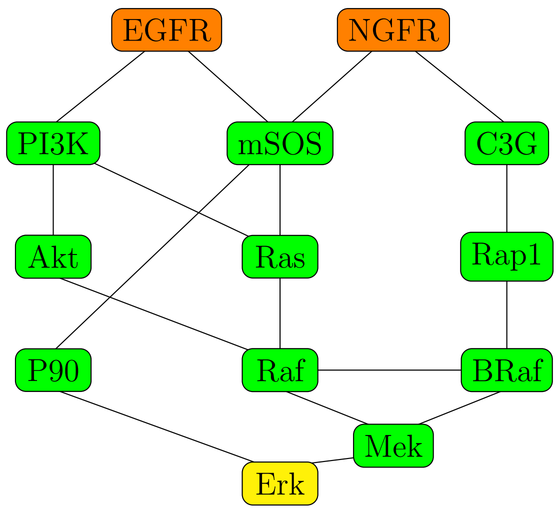

Figure 1: A network diagram describing the action of NGF and EGR on Erk (a simplified version of [8, Figure 1])

We have applied our algorithm to the model with the observable being the sum of all species (see Remark 4). The resulting reduction agrees with the one from [27, Section 5.4], that is, the macro-variables are the following

- concentrations of free and bound EGF and NGF and the corresponding receptors (EGFR and NGFR) remain separate variables;
- a single variable with constant dynamics equal to the sum of the concentrations of all other species (intermediate proteins and Erk) is introduced;
- only 4 out of 48 parameters remain in the reduced model.

Therefore, the reduced model is defined by 7 variables and 4 parameters and captures exactly the dynamics of EGF and NGF.

6 Conclusions

We have presented the first (to the best of our knowledge) algorithms for finding optimal constrained linear lumping of a rational dynamical system which non trivially extends the existing algorithm for the polynomial case.

While being based on the Jacobian-invariance criterion going back to [20] and used in the polynomial case [25], our main algorithm approaches the key step of [25], turning a nonconstant Jacobian matrix into a finite collection of constant matrices, from a different angle, via automatic differentiation and randomized evaluation. We implement our algorithms, report runtimes for them on a set of benchmarks, and demonstrate how they can be applied to models from the literature.

Directions for future research include extending the algorithm to models involving other functions such as exponential, logarithmic, and trigonometric. Our current approach is not directly applicable to such functions as one can evaluate them at rational points only approximately, not exactly.

Acknowledgements

We are grateful to Mirco Tribastone for helpful discussions. We are grateful to the referees for their helpful suggestions.

Appendix: Proofs

Lemma 6. *Consider the rational dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and $J(\mathbf{x})$ the Jacobian matrix of $\mathbf{f}(\mathbf{x})$. Let $\mathcal{B}$ be any set of matrices spanning the vector space $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. Then $L \in \mathbb{R}^{r\times n}$ is a lumping if and only if the row space of L is invariant with respect to all $J \in \mathcal{B}$.*

Proof. Since $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$ has finite dimension, then:

- There are points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^n$ such that $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_N)$ is a basis.
- There are matrices $J_1, \ldots, J_N \in \mathcal{B}$ that form also a basis.

Hence, there is a invertible matrix $C \in \mathbb{R}^{N \times N}$ such that (the $n \times n$ matrices are considered as n^2-dimensional vectors)

$$\begin{pmatrix} J_1 \\ \vdots \\ J_N \end{pmatrix} = C \begin{pmatrix} J(\mathbf{x}_1) \\ \vdots \\ J(\mathbf{x}_N) \end{pmatrix}.$$

Assume that L is a lumping, then [25, Propsition II.1] implies that there is $A(\mathbf{x})$ such that $A(\mathbf{x})L = LJ(\mathbf{x})$. Hence, for the matrices defined by

$$\begin{pmatrix} A_1 \\ \vdots \\ A_N \end{pmatrix} = C \begin{pmatrix} A(\mathbf{x}_1) \\ \vdots \\ A(\mathbf{x}_N) \end{pmatrix},$$

we obtain $A_i L = LJ_i$. So for all $J \in \mathcal{B}$ there is A_J such that $A_J L = LJ$.

On the other hand, if for all $J \in \mathcal{B}$ there is A_J such that $A_J L = LJ$, then we can check that, if $J(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i(\mathbf{x}) J(\mathbf{x}_i)$, then

$$LJ(\mathbf{x}) = \left[(\alpha_1(\mathbf{x}), \ldots, \alpha_N(\mathbf{x})) C^{-1} \begin{pmatrix} A_{J_1} \\ \vdots \\ A_{J_N} \end{pmatrix} \right] L,$$

so the row space of L is invariant under $J(\mathbf{x})$ and L is a lumping. $\square$

Proposition 11. *Let $\mathbf{f} = (f_1, \ldots, f_n)^T$ be a vector of elements of $\mathbb{R}(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)^T$, and $\varepsilon \in (0,1)$ be a real number. Let D_d (resp., D_n) be an integer such that the degree of the denominator (resp., numerator) of f_i does not exceed D_d (resp., D_n) for every $1 \leqslant i \leqslant n$.*

Let $J(\mathbf{x})$ be the Jacobian matrix of $\mathbf{f}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_m$ be points such that $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)$ do not span $\langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n$ and $J(\mathbf{x})$ is well-defined$\rangle$. Consider a point $\mathbf{x}_{m+1}$ with each coordinate being an integer sampled uniformly at random from $\{1, 2, \ldots, N\}$ where

$$N > \frac{D_n + (2m+1)D_d}{\varepsilon} + nD_d.$$

Then we have

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m) \rangle \mid J(\mathbf{x}_{m+1}) \text{ is well-defined}] < \varepsilon.$$

Proof. Define $\mathbf{v}_1, \ldots \mathbf{v}_m \in \mathbb{R}^{n^2}$ as the vector representations of the matrices $J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)$ and $\mathbf{w}(x)$ be the corresponding vector representation for the matrix $J(\mathbf{x})$ with the entries being rational functions. By removing the corresponding evaluation points if necessary, we will further assume that $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are linearly independent. The fact that $\langle J(\mathbf{x}_i) \mid 1 \leqslant i \leqslant m \rangle \neq \langle J(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n \rangle$ implies that

$$\operatorname{rank} E = m + 1, \quad \text{where} \quad E := (\mathbf{v}_1 \mid \mathbf{v}_2 \mid \ldots \mid \mathbf{v}_m \mid \mathbf{w}(\mathbf{x})).$$

Each maximal minor of E is a linear combination of the entries of $\mathbf{w}(\mathbf{x})$. Since $\operatorname{rank} E = m+1$, at least one of these minors is a nonzero rational function, we write it as $\frac{A(\mathbf{x})}{B(\mathbf{x})}$. Note the degrees of the denominator and numerator of each entry of $\mathbf{w}(\mathbf{x})$ are bounded by $2D_d$ and $D_d + D_n$, respectively. Therefore, we have

$$\deg(A(\mathbf{x})) \leqslant D_n + (2m+1)D_d, \qquad \deg(B(\mathbf{x})) \leqslant 2(m+1)D_d.$$

Let $q(\mathbf{x})$ be the product of the denominators of $f_1, \ldots, f_n$. Then $\deg q(\mathbf{x}) \leqslant nD_d$.

Let $\mathbf{x}_{m+1} \in \{1, \ldots N\}^n$ as in the statement of the proposition. We would like to find an upper bound for

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)\rangle \mid J(\mathbf{x}_{m+1}) \text{ is well-defined}]. \tag{5}$$

We write this as

$$\frac{\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)\rangle \text{ and } J(\mathbf{x}_{m+1}) \text{ is well-defined}]}{\mathbb{P}[J(\mathbf{x}_{m+1}) \text{ is well-defined}]}.$$

For the numerator:

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)\rangle \text{ and } J(\mathbf{x}_{m+1}) \text{ is well-defined}] \leqslant$$
$$\leqslant \mathbb{P}[A(\mathbf{x}_{m+1}) = 0] \leqslant \frac{D_n + (2m+1)D_d}{N},$$

where the latter inequality follows from the Schwartz-Zippel lemma [36, Proposition 98]. Using this lemma again, we bound the probability of the denominator:

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \text{ is well-defined}] = \mathbb{P}[q(\mathbf{x}) \neq 0] \geqslant 1 - \frac{nD_d}{N}.$$

Putting everything together, we obtain

$$\mathbb{P}[J(\mathbf{x}_{m+1}) \in \langle J(\mathbf{x}_1), \ldots, J(\mathbf{x}_m)\rangle \mid J(\mathbf{x}_{m+1}) \text{ is well-defined}] \leqslant \frac{D_n + (2m+1)D_d}{N - nD_d}.$$

Now a direct computation shows that

$$\frac{D_n + (2m+1)D_d}{N - nD_d} < \varepsilon \iff N > \frac{D_n + (2m+1)D_d}{\varepsilon} + nD_d.$$

□

References

[1] Antoulas, A.: Approximation of Large-Scale Dynamical Systems. Adv. in Design and Control, SIAM (2005)

[2] Barua, D., Faeder, J.R., Haugh, J.M.: A bipolar clamp mechanism for activation of jak-family protein tyrosine kinases. PLoS Computational Biology **5**(4), e1000364 (2009), https://doi.org/10.1371/journal.pcbi.1000364

[3] Baur, W., Strassen, V.: The complexity of partial derivatives. Theoretical Computer Science **22**(3), 317–330 (1983), https://doi.org/10.1016/0304-3975(83)90110-X

[4] Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research **18**(153), 1–43 (2018), http://jmlr.org/papers/v18/17-468.html

[5] Borisov, N., Markevich, N., Hoek, J., Kholodenko, B.: Signaling through receptors and scaffolds: Independent interactions reduce combinatorial complexity. Biophysical Journal **89**(2), 951–966 (2005), http://dx.doi.org/10.1529/biophysj.105.060533

[6] Borisov, N., Kholodenko, B., Faeder, J., Chistopolsky, A.: Domain-oriented reduction of rule-based network models. IET Systems Biology **2**(5), 342–351 (2008), https://doi.org/10.1049/iet-syb:20070081

[7] Brännmark, C., Nyman, E., Fagerholm, S., Bergenholm, L., Ekstrand, E.M., Cedersund, G., Strålfors, P.: Insulin signaling in type 2 diabetes. Journal of Biological Chemistry **288**(14), 9867–9880 (2013), https://doi.org/10.1074/jbc.m112.432062

[8] Brown, K.S., Hill, C.C., Calero, G.A., Myers, C.R., Lee, K.H., Sethna, J.P., Cerione, R.A.: The statistical mechanics of complex signaling networks: nerve growth factor signaling. Physical Biology **1**(3), 184–195 (2004), https://doi.org/10.1088/1478-3967/1/3/006

[9] Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: ERODE: A tool for the evaluation and reduction of ordinary differential equations. In: Legay, A., Margaria, T. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 310–328 (2017)

[10] Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Maximal aggregation of polynomial dynamical systems. Proceedings of the National Academy of Sciences **114**(38), 10029–10034 (2017), https://www.pnas.org/content/114/38/10029

[11] Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Symbolic computation of differential equivalences. Theoretical Computer Science **777**, 132–154 (2019), https://doi.org/10.1016/j.tcs.2019.03.018

[12] Chou, T.C., Talalay, P.: A simple generalized equation for the analysis of multiple inhibitions of Michaelis-Menten kinetic systems. Journal of Biological Chemistry **252**, 6438–6442 (1977), https://www.jbc.org/article/S0021-9258(17)39978-7/pdf

[13] Conzelmann, H., Fey, D., Gilles, E.: Exact model reduction of combinatorial reaction networks. BMC Systems Biology **2**(1), 78 (2008), http://dx.doi.org/10.1186/1752-0509-2-78

[14] Dupont, G., Goldbeter, A.: Protein phosphorylation driven by intracellular calcium oscillations: A kinetic analysis. Biophysical Chemistry **42**(3), 257–270 (1992), https://doi.org/10.1016/0301-4622(92)80018-z

[15] Elliott, C.: Beautiful differentiation. In: International Conference on Functional Programming (ICFP) (2009), http://conal.net/papers/beautiful-differentiation

[16] Faeder, J.R., Hlavacek, W.S., Reischl, I., Blinov, M.L., Metzger, H., Redondo, A., Wofsy, C., Goldstein, B.: Investigation of early events in fcεRI-mediated signaling using a detailed mathematical model. The Journal of Immunology **170**(7), 3769–3781 (2003), https://doi.org/10.4049/jimmunol.170.7.3769

[17] Feret, J., Danos, V., Krivine, J., Harmer, R., Fontana, W.: Internal coarse-graining of molecular systems. Proceedings of the National Academy of Sciences **106**(16), 6453–6458 (2009), http://dx.doi.org/10.1073/pnas.0809908106

[18] Hoffmann, P.H.: A Hitchhiker's Guide to Automatic Differentiation. Numerical Algorithms **72**, 775–811 (2016), https://doi.org/10.1007/s11075-015-0067-6

[19] Kallenberger, S.M., Beaudouin, J., Claus, J., Fischer, C., Sorger, P.K., Legewie, S., Eils, R.: Intra- and interdimeric caspase-8 self-cleavage controls strength and timing of CD95-induced apoptosis. Science Signaling **7**(316) (2014), https://doi.org/10.1126/scisignal.2004738

[20] Li, G., Rabitz, H.: A general analysis of exact lumping in chemical kinetics. Chemical Engineering Science **44**(6), 1413–1430 (1989), https://doi.org/10.1016/0009-2509(89)85014-6

[21] Li, J., Wang, L., Hashimoto, Y., Tsao, C.Y., Wood, T.K., Valdes, J.J., Zafiriou, E., Bentley, W.E.: A stochastic model ofEscherichia coliAI-2 quorum signal circuit reveals alternative synthesis pathways. Molecular Systems Biology **2**(1), 67 (2006), https://doi.org/10.1038/msb4100107

[22] Malik-Sheriff, R.S., Glont, M., Nguyen, T.V.N., Tiwari, K., Roberts, M.G., Xavier, A., Vu, M.T., Men, J., Maire, M., Kananathan, S., Fairbanks, E.L., Meyer, J.P., Arankalle, C., Varusai, T.M., Knight-Schrijver, V., Li, L., Dueñas-Roca, C., Dass, G., Keating, S.M., Park, Y.M., Buso, N., Rodriguez, N., Hucka, M., Hermjakob, H.: BioModels — 15 years of sharing computational models in life science. Nucleic Acids Research **48**(D1), D407–D415 (2020), https://doi.org/10.1093/nar/gkz1055

[23] Neves, S.R., Tsokas, P., Sarkar, A., Grace, E.A., Rangamani, P., Taubenfeld, S.M., Alberini, C.M., Schaff, J.C., Blitzer, R.D., Moraru, I.I., Iyengar, R.: Cell shape and negative links in regulatory motifs together control spatial information flow in signaling networks. Cell **133**(4), 666–680 (2008), https://doi.org/10.1016/j.cell.2008.04.025

[24] Okino, M., Mavrovouniotis, M.: Simplification of mathematical models of chemical reaction systems. Chemical Reviews **2**(98), 391–408 (1998), https://doi.org/10.1021/cr950223l

[25] Ovchinnikov, A., Pérez Verona, I., Pogudin, G., Tribastone, M.: CLUE: exact maximal reduction of kinetic models by constrained lumping of differential equations. Bioinformatics **37**(19), 3385–3385 (2021), https://doi.org/10.1093/bioinformatics/btab258

[26] Pepke, S., Kinzer-Ursem, T., Mihalas, S., Kennedy, M.B.: A dynamic model of interactions of Ca^{2+}, calmodulin, and catalytic subunits of Ca^{2+}/calmodulin-dependent protein kinase II. PLoS Computational Biology **6**(2), e1000675 (2010), https://doi.org/10.1371/journal.pcbi.1000675

[27] Perez-Verona, I.C., Tribastone, M., Vandin, A.: A large-scale assessment of exact lumping of quantitative models in the BioModels repository. Theoretical Computer Science **893**, 41–59 (2021), https://doi.org/10.1016/j.tcs.2021.06.026

[28] Poolman, M.G., Assmus, H.E., Fell, D.A.: Applications of metabolic modelling to plant metabolism. Journal of Experimental Botany **55**(400), 1177–1186 (2004), https://doi.org/10.1093/jxb/erh090

[29] Raia, V., Schilling, M., Böhm, M., Hahn, B., Kowarsch, A., Raue, A., Sticht, C., Bohl, S., Saile, M., Möller, P., Gretz, N., Timmer, J., Theis, F., Lehmann, W.D.,

Lichter, P., Klingmüller, U.: Dynamic mathematical modeling of IL13-induced signaling in hodgkin and primary mediastinal b-cell lymphoma allows prediction of therapeutic targets. Cancer Research **71**(3), 693–704 (2010), https://doi.org/10.1158/0008-5472.can-10-2987

[30] Rohwer, J.M., Botha, F.C.: Analysis of sucrose accumulation in the sugar cane culm on the basis of in vitro kinetic data. Biochemical Journal **358**(2), 437–445 (2001), https://doi.org/10.1042/bj3580437

[31] Schnell, S., Mendoza, C.: Enzyme kinetics of multiple alternative substrates. Journal of Mathematical Chemistry **27**(1/2), 155–170 (2000), https://doi.org/10.1023/a:1019139423811

[32] Sneddon, M.W., Faeder, J.R., Emonet, T.: Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nature Methods **8**(2), 177–183 (2010), https://doi.org/10.1038/nmeth.1546

[33] Warner, F.W.: Foundations of Differentiable Manifolds and Lie Groups. Springer New York, NY (1983), https://doi.org/10.1007/978-1-4757-1799-0

[34] Weiße, A.Y., Oyarzún, D.A., Danos, V., Swain, P.S.: Mechanistic links between cellular trade-offs, gene expression, and growth. Proceedings of the National Academy of Sciences **112**(9), E1038–E1047 (2015), https://doi.org/10.1073/pnas.1416533112

[35] Wengert, R.E.: A simple automatic derivative evaluation program. Commun. ACM **7**(8), 463–464 (aug 1964), https://doi.org/10.1145/355586.364791

[36] Zippel, R.: Effective Polynomial Computation. Springer (1993), http://dx.doi.org/10.1007/978-1-4615-3188-3