CLUE: Exact maximal reduction of kinetic models by constrained lumping of differential equations

Alexey Ovchinnikov[*], Isabel Pérez Verona[†], Gleb Pogudin[‡][§], Mirco Tribastone[¶]

Abstract

Motivation: Detailed mechanistic models of biological processes can pose significant challenges for analysis and parameter estimations due to the large number of equations used to track the dynamics of all distinct configurations in which each involved biochemical species can be found. Model reduction can help tame such complexity by providing a lower-dimensional model in which each macro-variable can be directly related to the original variables.
Results: We present CLUE, an algorithm for exact model reduction of systems of polynomial differential equations by constrained linear lumping. It computes the smallest dimensional reduction as a linear mapping of the state space such that the reduced model preserves the dynamics of user-specified linear combinations of the original variables. Even though CLUE works with nonlinear differential equations, it is based on linear algebra tools, which makes it applicable to high-dimensional models. Using case studies from the literature, we show how CLUE can substantially lower model dimensionality and help extract biologically intelligible insights from the reduction.
Availability: An implementation of the algorithm and relevant resources to replicate the experiments herein reported are freely available for download at https://github.com/pogudingleb/CLUE.
Supplementary information: Supplementary materials are enclosed below.

1 Introduction

Kinetic models of biochemical systems hold the promise of being able to unravel mechanistic insights in living cells as well as predict the behavior of a biological process under unseen circumstances, which is a fundamental premise for many applications including control and synthesis.

In order to obtain an accurate model, however, it is often necessary to incorporate a substantial amount of detail about the specific mechanisms of interaction between the different components of a biological system. In many cases, this may lead to an overall representation which hinders physical intelligibility. For example, a mechanistic description of protein phosphorylation—a basic, ubiquitous process in signaling pathways [Pawson and Scott, 2005]—may yield models with a combinatorially large number of variables, particularly in the case of multisite phosphorylation [Salazar and Höfer, 2009].

Model reduction represents a promising class of methods designed for obtaining a lower-dimensional representation that retains some dynamical features of interest to the modeler. The substantial body of research available is motivated by the fact that it is a cross-cutting concern throughout many scientific and engineering disciplines to be able to effectively work with simple but accurate models of complex systems. Specifically, for applications to systems biology the availability of a smaller model can be particularly beneficial in order to reduce the number of kinetic parameters [Danø et al., 2006], whose measurements and calibration is a well-known hindrance, see, e.g., [Babtie and Stumpf, 2017].

[*]aovchinnikov@qc.cuny.edu, Department of Mathematics, CUNY Queens College, Queens, NY, 11367 and CUNY Graduate Center, New York, NY, 10016, USA
[†]isabel.perez@alumni.imtlucca.it, IMT School for Advanced Studies, Lucca, 55100, Italy
[‡]gleb.pogudin@polytechnique.edu, LIX, CNRS, École Polytechnique, Institute Polytechnique de Paris, Palaiseau, 91120, France
[§]Corresponding author. The authors are ordered alphabetically.
[¶]mirco.tribastone@imtlucca.it, IMT School for Advanced Studies, Lucca, 55100, Italy

Techniques based on balanced truncation and singular value decomposition can dramatically lower the dimensionality of a model with small approximation errors [Antoulas, 2005]. Since the reduced model preserves the input/output behavior, it can be conveniently used in place of the original model to speed up the computation time of a numerical simulation. However, the coordinate transformation typically destroys the structure, leading to a loss of physical interpretability of the model. This is recognized as an important property to be maintained in applications to systems biology, especially if the model is used to validate mechanistic hypotheses [Schmidt et al., 2008, Sunnaker et al., 2011, Apri et al., 2012].

For models of biochemical systems, many reduction methods are based on exploiting time-scale separation [Okino and Mavrovouniotis, 1998]. One of the most well-known approaches is quasi-steady-state approximation [Segel and Slemrod, 1989], in which, roughly speaking, "fast" variables can be approximated as reaching their stationary values such that they can be replaced by constants (solutions of the associated system of equations) in the dynamical model for the "slow" variables. Another class of reduction techniques based on sensitivity analysis studies how model parameters and variables affect the desired output, suggesting the elimination of the least influential ones [Snowden et al., 2017].

Exact model reduction aims at lowering dimensionality without introducing approximation errors in the reduced model. Conservation analysis detects linear combinations of variables that remain constant at all times [Vallabhajosyula et al., 2005]. Exact lumping is a more general approach whereby it is possible to write a self-consistent system of dynamical equations for a set of macro-variables in which each macro-variable represents a combination of the original ones [Okino and Mavrovouniotis, 1998]. Linear lumping, known as early as in Wei and Kuo [1969], expresses such combinations as a linear mapping on the original state variables. To maintain some degree of physical interpretability, the lumping may be restricted only to a part of the state space. Li and Rabitz [1991] allow the specification of linear combination of variables that ought to be preserved. More recently, Cardelli et al. [2017a] presented a lumping algorithm that identifies a partition of the state variables such that in the lumped system each macro-variable represents the sum of the original variables of a block. Specialized lumping criteria have also been studied for classes of biochemical models for signaling pathways, e.g., [Borisov et al., 2005, Conzelmann et al., 2006, Feret et al., 2009], for example by analyzing higher-level descriptions such as rule-based systems from which ordinary differential equation (ODE) models can be generated [Danos and Laneve, 2004, Blinov et al., 2004]. However, these approaches identify specific types of exact linear lumping, namely if the macro-variables represent sums of the original variables of the original model (but not necessarily induced by a partition of the state space as in Cardelli et al. [2017a]).

There are connections between exact model reduction and observability/identifiability (see [Miao et al., 2011, Villaverde, 2019] for discussion of these notions in the biological contexts). If a model admits an exact reduction preserving the observed variables, then it is not observable because the states with the same image under the reduction are indistinguishable. On the other hand, the reduction provides a reparametrization with fewer degrees of freedom with respect to the observations. Moreover, since the reduction is exact, the state (or parameter) identification can be carried out for the reduced model without any loss of information but at lesser computational costs, especially for identification methods sensitive to the model dimension (e.g., sampling-based ones [Ballnus et al., 2017]).

Here we present CLUE, an algorithm for constrained linear lumping, applicable to models as ODEs with polynomial derivatives. The constraints represent the linear combinations of state variables that ought to be maintained in the reduced model, similarly to Li and Rabitz [1991]. The algorithm hinges on the fundamental observation by the same authors [Li and Rabitz, 1989, 1991] that exact lumpings correspond to the subspaces that are invariant under the Jacobian of the ODE system. For finding these subspaces, Li and Rabitz [1989, 1991] suggest two ways: (i) produce a finite set of constant matrices such that every common invariant subspace of these matrices would be invariant for the Jacobian (but not necessarily vice versa); and (ii) find eigenvectors and eigenvalues of the Jacobian symbolically, and explore their combinations. In the former approach, the obtained set of matrices might be too restrictive, so a lumping might not be found even if there is one. The latter approach is limited to small sized systems because it involves finding symbolic expressions to the eigenvalues of a nonconstant matrix (the Jacobian of the system) and requires human intervention for exploring various combinations of these eigenvectors (for example, [Li and Rabitz, 1989, §4, Example 2]).

Our main contribution is twofold. First, we provide a set of constant matrices whose common invariant subspaces are exactly the invariant subspaces of the Jacobian. This allows us to obtain a fully algorithmic method for finding a constrained lumping based purely on linear algebra. Second, we improve the algorithm by eliminating redundant computation from the invariant subspace generation and by using modular computation to avoid intermediate expression swell. This enables the analysis of models with several thousands of equations on commodity hardware. Together, our results allow us to study large-scale biochemical models, of which we present a number of case studies, showing the degree of lumpability achieved as well as the physical interpretation of the reduced system.

2 Approach and method

Definition 1 (Lumping)**.** Consider a system of ODEs with polynomial right-hand side in the form

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

where $\mathbf{x} = (x_1, \dots, x_n)^T$, $\mathbf{f} = (f_1, \dots, f_n)^T$, and $f_1, \dots, f_n \in \mathbb{R}[\mathbf{x}]$. We say that a linear transformation $\mathbf{y} = L\mathbf{x}$ with $\mathbf{y} = (y_1, \dots, y_m)^T$, $L \in \mathbb{R}^{m \times n}$, and $\operatorname{rank} L = m$ is *a lumping of* (1) if there exist $\mathbf{g} = (g_1, \dots, g_m)^T$ with $g_1, \dots, g_m \in \mathbb{R}[\mathbf{y}]$ such that

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y})$$

for every solution $\mathbf{x}$ of (1). We say that m is *the dimension of the lumping*. The variables $\mathbf{y}$ in the reduced system are called *macro-variables*.

Example 1. Consider the system

$$\dot{x}_1 = x_2^2 + 4x_2x_3 + 4x_3^2, \quad \dot{x}_2 = 4x_3 - 2x_1, \;\; \dot{x}_3 = x_1 + x_2. \tag{2}$$

We claim that the matrix

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{pmatrix} \tag{3}$$

gives a lumping of (2) of dimension two. Indeed,

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 + 2\dot{x}_3 \end{pmatrix} = \begin{pmatrix} (x_2 + 2x_3)^2 \\ 2x_2 + 4x_3 \end{pmatrix} = \begin{pmatrix} y_2^2 \\ 2y_2 \end{pmatrix},$$

so we can take $g_1(y_1, y_2) = y_2^2$ and $g_2(y_1, y_2) = 2y_2$.

The lumping matrix of (3) turns out to exactly preserve the solution of variable x_1. In general, one considers a vector $\mathbf{x}_{\text{obs}}$ of combinations of the original variables that are to be recovered in the reduced system; that is, $\mathbf{x}_{\text{obs}}$ is a vector of linearly independent forms in $\mathbf{x}$ such that $\mathbf{x}_{\text{obs}} = A\mathbf{x}$. Then we say that a lumping $\mathbf{y} = L\mathbf{x}$ is a *constrained linear lumping* if each entry of $\mathbf{x}_{\text{obs}}$ is a linear combination of the entries of $\mathbf{y}$.

Example 2. Using the system (2), setting

$$\mathbf{x}_{\text{obs}} = A\mathbf{x}, \qquad \text{with} \qquad A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \end{pmatrix},$$

we see that (3) is a constrained linear lumping because

$$\mathbf{x}_{\text{obs}} = \begin{pmatrix} x_1 \\ x_1 + x_2 + 2x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_1 + y_2 \end{pmatrix}.$$

Instead, setting $\mathbf{x}_{\text{obs}} = (x_2)$ does not give a constrained lumping for L because $(0, 1, 0)$ does not belong to the row space of L.

Algorithm 1 Simplified algorithm for finding a constrained lumping of the smallest possible dimension

```
Input  a system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ of $n$ ODEs with a polynomial right-hand side and an $s \times n$ matrix $A$ over $\mathbb{R}$ of rank
       $s > 0$;
Output  a matrix $L$ such that $\mathbf{y} := L\mathbf{x}$ is a constrained lumping with observables $A\mathbf{x}$ of smallest possible
       dimension.
(Step 1) Compute $J(\mathbf{x})$, the Jacobian matrix of $\mathbf{f}(\mathbf{x})$.
(Step 2) Represent $J(\mathbf{x})$ as $J_1 m_1 + \ldots + J_N m_N$, where $m_1, \ldots, m_N$ are distinct monomials in $\mathbf{x}$, and $J_1, \ldots, J_N$ are
         nonzero matrices over $\mathbb{R}$.
(Step 3) Set $L := A$.
(Step 4) Repeat
    (a) for every $M$ in $J_1, \ldots, J_N$ and row $r$ of $L$, if $rM$ does not belong to the row space of $L$, append $rM$
        to $L$.
    (b) if nothing has been appended in the previous step, exit the repeat loop and go to (Step 5).
(Step 5) Return $L$.
```

For a given vector $\mathbf{x}_{\text{obs}}$, there may be more than one constrained linear lumping. We define two lumpings $\mathbf{y}_1 = L_1\mathbf{x}$ and $\mathbf{y}_2 = L_2\mathbf{x}$ to be equivalent if there exists an invertible matrix T such that $L_1 = TL_2$. It is possible to prove that, for every nonzero vector $\mathbf{x}_{\text{obs}}$, there exists a unique (up to equivalence) lumping of the smallest possible dimension.

Let $J(\mathbf{x})$ be the Jacobian matrix of $\mathbf{f}$. From [Li and Rabitz, 1989], L gives a lumping of (1) if and only if the row space of $LJ(\mathbf{x})$ is contained in the row space of L *for all* $\mathbf{x}$. The universal quantifier in this characterization can be handled in different ways (e.g., see [Li and Rabitz, 1989, §3] and [Brochot et al., 2005, pages 722-723]). We eliminate it as follows. Since the entries of $J(\mathbf{x})$ are polynomials in $\mathbf{x}$, we can write $J(\mathbf{x})$ as

$$J(\mathbf{x}) = \sum_{i=1}^{N} J_i m_i, \tag{4}$$

where $m_1, \ldots, m_N$ are distinct monomials in $\mathbf{x}$ and $J_1, \ldots, J_N$ are matrices over $\mathbb{R}$. Then, the fact that the row space of $LJ(\mathbf{x})$ is contained in the row space of L for every $\mathbf{x}$ is equivalent to the containment of the row space of LJ_i in the row space of L for every $i = 1, \ldots, N$ (proved in the supplementary material, see Lemma I.1; the equivalence does not hold for the method from [Li and Rabitz, 1989, §3], see Remark I.1). This leads to the following algorithm: we start with matrix A and add products of its rows with the matrices $J_1, \ldots, J_N$ as long as the dimension of the row space grows. This is detailed in Algorithm 1.

Example 3. We illustrate Algorithm 1 by applying it to the system in Eq. (2) by choosing $A = (1, 0, 0)$ (thus corresponding to recovering x_1 in the reduced system). The Jacobian matrix of $\mathbf{f}(\mathbf{x})$ in Eq. (2) is

$$J(\mathbf{x}) = \begin{pmatrix} 0 & 2x_2 + 4x_3 & 4x_2 + 8x_3 \\ -2 & 0 & 4 \\ 1 & 1 & 0 \end{pmatrix},$$

which can be decomposed as $J_1 m_1 + J_2 m_2 + J_3 m_3$, where $m_1 = 1$, $m_2 = x_2$, $m_3 = x_3$, and

$$J_1 = \begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 4 \\ 1 & 1 & 0 \end{pmatrix}, \quad J_2 = \begin{pmatrix} 0 & 2 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad J_3 = \begin{pmatrix} 0 & 4 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Starting with $L = (1,0,0)$, for $r = (1,0,0)$ we compute the products:

$$rJ_1 = (0,0,0), \quad rJ_2 = (0,2,4), \quad rJ_3 = (0,4,8).$$

Since rJ_1 belongs to the row space of L while rJ_2 does not, we set

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 4 \end{pmatrix}.$$

The third vector, rJ_3, is proportional to the second row of the new L, so we skip it. Since a new row, $(0,2,4)$, has been added in part **(a)** of **(Step 4)**, we do not exit the loop. Setting now $r = (0,2,4)$, we get

$$rJ_1 = (0,4,8), \quad rJ_2 = (0,0,0), \quad rJ_3 = (0,0,0).$$

Since all these vectors belong to the row space of L, the iteration terminates and the as-computed L gives the lumping of the smallest dimension from which we can recover the original quantities specified through A. This L is not equal to the one in (3) but is equivalent to it in the above sense.

3 Implementation

Algorithm 2 Finding the smallest invariant subspace
(to be used instead of **(Step 4)** of Algorithm 1)

```
Input an $s \times n$ matrix $A$ over field $\mathbb{K}$ and a list $M_1, \ldots, M_\ell$ of $n \times n$ matrices over $\mathbb{K}$;

Output an $r \times n$ matrix $L$ over $\mathbb{K}$ such that
    • the row span of $A$ is contained in the row span of $L$.
    • for every $1 \leqslant i \leqslant \ell$, the row span of span of $LM_i$ is contained in the row span of $L$;
    • $r$ is the smallest possible.

(Step 1) Let $L$ be the reduced row echelon form of $A$.

(Step 2) Set $P$ be the set of indices of the pivot columns of $L$.

(Step 3) While $P \neq \varnothing$ do
    (a) For every $j \in P$ and every $1 \leqslant i \leqslant \ell$
        i. Let $r$ be the row in $L$ with the index of the pivot being $j$.
        ii. Reduce $rM_i$ with respect to $L$. If the result is not zero, append it as a new row to $L$.
        iii. Reduce other rows with respect the new one in order to bring $L$ into the reduced row echelon form.
    (b) Let $\widetilde{P}$ be the set of indices of the pivot columns of $L$.
    (c) Set $P := \widetilde{P} \setminus P$.

(Step 4) Return $L$.
```

The CLUE algorithm was implemented in Python using the SymPy library [Meurer et al., 2017]. The source code and all examples from Section 4 are available at https://github.com/pogudingleb/CLUE. Our implementation accepts models typed manually or provided in the input format of the tool ERODE [Cardelli et al., 2017b] (which has an importer from SBML, see Pérez-Verona et al. [2019] for details).

For our implementation, we keep the general framework of Algorithm 1 but replace **(Step 4)**, the most time-consuming part, with a more efficient algorithm. **(Step 4)** of Algorithm 1 solves the following problem: given a set of n-dimensional vectors and a set of $n \times n$ matrices, find a basis of the smallest vector space that is invariant under the matrices and that contains the vectors.

We present and implement two algorithms, Algorithm 2 and 3 for replacing **(Step 4)** of Algorithm 1 (we apply them with $A = L$, $\ell = N$, and $M_i = J_i$ for every i). The former is typically faster if the intermediate results of the computation are not too large, while the performance of the latter is less sensitive to the intermediate expression swell, and it can tackle models that are out of reach for the former (for more details, see Section V in the Supplementary Materials). Algorithm 3 requires the system to have rational coefficients, but all of the systems we considered are of this type. Therefore, our implementation runs Algorithm 2 first and, if the size of intermediate expressions exceeds a threshold (10000 digits), stops it and runs Algorithm 3. We now describe the key ideas behind these algorithms.

Algorithm 2 is a result of applying the following observations to **(Step 4)** of Algorithm 1:

- If *we maintain L in the reduced row echelon form*, we can test whether a vector rM belongs to the row space of L in $O(n^2)$ instead of computing rank, e.g., in $O(n^3)$ using Gaussian elimination or in $O(n^{2.373})$ using more advanced algorithms [Bürgisser et al., 1997, Section 16.5].
- *We do not need to consider all the products* of the from rM but only the ones corresponding to the pivots of L added at the previous iteration of the loop. The largest number of products considered is reduced from about $n^2N/2$ to at most nN; for justification and details, see the proof of Proposition I.1 in the Supplementary Materials.

In CLUE, we also take advantage of the fact that the input matrices are frequently very sparse (see Remark III.1 in the Supplementary Materials).

Algorithm 3 Finding the smallest invariant subspace (modular)
(to be used instead of **(Step 4)** of Algorithm 1)

Input $s \times n$ matrix A and a list $M_1, \ldots, M_\ell$ of $n \times n$ matrices over $\mathbb{Q}$;

Output an $r \times n$ matrix L over $\mathbb{Q}$ such that:

- the row span of A is contained in the row span of L.
- for every $1 \leqslant i \leqslant \ell$, the row span of LM_i is contained in the row span of L;
- r is the smallest possible.

(Step 1) Repeat the following

(a) Pick a prime number p that does not divide any of the denominators in $A, M_1, \ldots, M_\ell$ and has not been chosen before.

(b) Compute the reductions $\widetilde{A}, \widetilde{M}_1, \ldots, \widetilde{M}_\ell$ modulo p.

(c) Run Algorithm 2 on $\widetilde{A}, \widetilde{M}_1, \ldots, \widetilde{M}_\ell$ as matrices over $\mathbb{F}_p$ and denote the result by $\widetilde{L}$.

(d) Apply the rational reconstruction algorithm ([von zur Garthen and Gerhard, 2013, § 5.10], [Wang et al., 1982]) to construct a matrix L over $\mathbb{Q}$ such that the reduction of L mod p equals $\widetilde{L}$.

(e) Check whether the row span of L contains the row span of L and is invariant under $M_1, \ldots, M_\ell$. If yes, exit the loop.

(Step 2) Return the matrix L from step (d) of the last iteration of the loop.

As mentioned, Algorithm 2 may not finish in reasonable time if the intermediate results of computation (exact rational numbers) grow too large (for examples, see Table 2 in the Supplementary Materials). We overcome this difficulty in **Algorithm 3** as follows. We run Algorithm 2 modulo a prime number. This may be much faster as it replaces computations with big integers using long arithmetic by computation with residues that typically fit into 64 bits. Then we reconstruct a possible output over rational numbers using rational reconstruction (see [von zur Garthen and Gerhard, 2013, § 5.10]) and verify its correctness. Since the integers in the output (unlike the ones in the computation) are typically small, the residue modulo prime usually contains already enough information for correct reconstruction. This verification stage is fast because it deals with the final output that is typically sparse and involves only small integers. We proceed with the a larger prime until the output is correct. In all practical examples we considered, one prime was enough (we start with $2^{31}-1$).

The correctness of Algorithms 2 and 3 is proved in Propositions I.1 and I.3 in Supplementary Materials. We perform a complexity analysis of Algorithms 1 and 2 in Section III of Supplementary Materials.

4 Examples

In this section, we show the applicability of CLUE to the reduction of biological models through a number of case studies published in the literature, including some taken from the BioModels repository [Li et al., 2010]. We additionally compare CLUE against the forward equivalence from [Cardelli et al., 2017a]. Forward equivalence identifies a partition of an ODE system with polynomial derivatives which induces a lumping where each macro-variable is equal to the sum of variables in each partition block. Using established terminology for lumping methods [Wei and Kuo, 1969, Okino and Mavrovouniotis, 1998, Snowden et al., 2017], forward equivalence can be understood as a form of *proper lumping* because each original variable contributes exactly to one macro-variable of the reduced system. By contrast, in general, constrained linear lumping yields an *improper* lumping matrix because the linear combination can be arbitrary. Similarly to constrained linear lumping, for forward equivalence there exists the notion of coarsest partition. This is the partition with the smallest number of blocks, thus leading to a reduced ODE system of the smallest dimension. In addition, forward equivalence can be computed with respect to constraints, which are encoded as an initial partition of variables. The algorithm for computing the coarsest partition iteratively splits each block of the initial partition until the criteria for forward equivalence are satisfied. For the comparison, we used ERODE [Cardelli et al., 2017b], which implements the reduction algorithm for forward equivalence. To the best of our knowledge, ERODE is the only publicly available software tool that supports exact lumping for polynomial differential equations. For each case study, both CLUE and forward equivalence were initialized so as to preserve the same observables in the reduced models.

For this study, we computed reductions that were independent from the specific choice of the kinetic parameters used in the models. This was done as follows. Given the original model in the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{k})$, where $\mathbf{k}$ is the vector of mass-action kinetic parameters, we considered an extended ODE system with the additional set of equations $\dot{\mathbf{k}} = 0$. This ensures that the reduction is independent from the initial conditions of the extended variables, hence of the choice of the original parameters.

Tables 1 and 2 summarize the numerical results for the models that are analyzed in more detail in the next subsections. For each model, Table 1 reports the reductions in both the state variables and the number of kinetic parameters. Overall, CLUE provides better reductions than ERODE because the partition found by forward equivalence is not guaranteed to be the minimal constrained linear lumping in general. Table 2 compares the runtimes of the reduction algorithms and of the numerical ODE solutions of the reduced models, computed on commodity hardware. CLUE requires more computation time. It must be noted that a precise comparison of the reduction runtimes is difficult because the forward equivalence and CLUE are implemented in different programming languages (i.e. Java and Python, respectively); instead, to allow for a fair comparison between the runtimes of the numerical ODE solutions, the reduced models were converted and solved in Matlab (using the stiff ODE solver `ode15s`).

The results indicate that the availability of a reduction becomes more relevant as the size of the original model grows. In particular, it makes feasible the analysis of models that would otherwise take considerable time (i.e., more than six hours in the cases $m > 7$ in the model by Sneddon et al. [2011]. Nevertheless, we remark that the reduction can be still useful if the runtimes of the reduction algorithms are of the same order as the runtimes of the numerical ODE solutions, especially in cases in which the reduced model must be subjected to an analysis involving multiple simulations, e.g. to study sensitivity to parameter changes or to initial conditions.

Table 1: Results for the case studies in Section 4 comparing CLUE with forward equivalence (FE) obtained using ERODE for the reduction of the state variables (Vars) and the kinetic parameters (Params).

	Vars			*Params*		
Model	*Orig.*	*FE*	*CLUE*	*Orig.*	*FE*	*CLUE*
Li et al. [2006]	21	19	15	25	22	22
Sneddon et al. [2011]						
$m = 2$	18	12	6	6	6	6
$m = 3$	66	22	6	6	6	6
$m = 4$	258	37	6	6	6	6
$m = 5$	1026	58	6	6	6	6
$m = 6$	4098	86	6	6	6	6
$m = 7$	16386	122	6	6	6	6
$m = 8$	65538	167	6	6	6	6
Faeder et al. [2003]	354	105	75	22	22	9
Borisov et al. [2008]	213	66	4	14	10	2

Table 2: Runtimes (in seconds) for the numerical solutions of the original model (third column) and the reduced models by forward equivalence (FE) and CLUE using initial conditions as in the original publications and time horizon in column (H); columns *Red.* indicate the runtimes for the respective reduction algorithms. Entries *t/o* indicates timeout after 6 hours.

			FE		*CLUE*	
Model	H	*Sim.*	*Red.*	*Sim.*	*Red.*	*Sim.*
Li et al. [2006]	100	0.004	0.001	0.003	0.023	0.003
Sneddon et al. [2011]	3					
$m = 2$		0.006	0.001	0.004	0.012	0.003
$m = 3$		0.021	0.002	0.007	0.048	0.003
$m = 4$		0.534	0.008	0.010	0.245	0.003
$m = 5$		20.9	0.035	0.032	1.115	0.003
$m = 6$		1125.0	0.185	0.299	5.238	0.005
$m = 7$		t/o	1.103	1.266	23.460	0.005
$m = 8$		t/o	3.552	6.128	97.509	0.005
Faeder et al. [2003]	100	0.268	0.110	0.035	0.863	0.023
Borisov et al. [2008]	50	0.065	0.006	0.010	0.263	<0.001

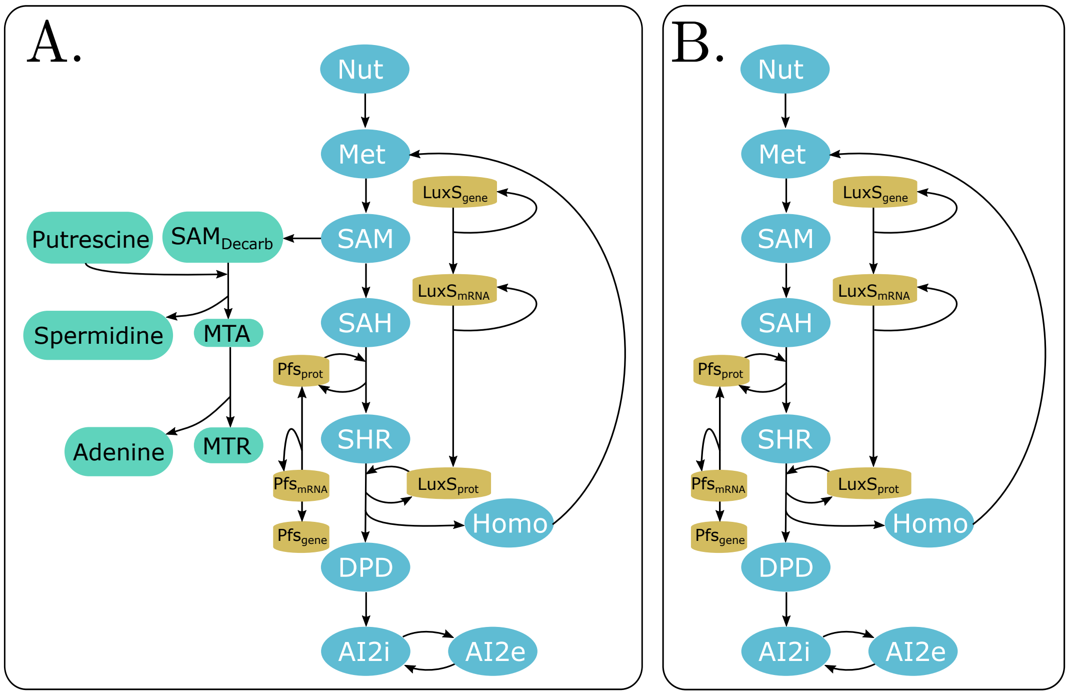

Figure 1: (A) Model for AI-2 synthesis adapted from [Li et al., 2006]. (B) Reduced network obtained with CLUE.

4.1 Modular decomposition of signaling pathways

We first illustrate how CLUE can decompose models of signaling pathways if only certain observables of interest are chosen. Figure 1(A) depicts a quorum sensing network for AI2 biosynthesis and uptake pathways in *E. coli* [Li et al., 2006]; the biochemical model is available in the BioModels repository as *MODEL8262229752*. The substrate Methionine (Met) transforms into S-adenosylmethionine (SAM). The blue branch of the pathway is involved in the production of AI2. The green branch depicts decarboxylation of SAM, which ultimately produces MTR and Adenine. The dynamics of both branches are mediated by Pfs.

The original model has 21 variables, one for each biochemical species depicted in the pathway. By fixing the output signal AI2 as the variable to be preserved, CLUE reduces the system to 15 variables. An inspection of the reduced model reveals that CLUE removes the biochemical species depicted as green boxes in Fig. 1(A), while the remaining variables are not aggregated further. Overall, this leads to a reduced model that can be interpreted as the network in Fig. 1(B). The reduction can be explained by the fact that none of the eliminated variables contribute to the dynamics of the chosen observables. This is because the interactions between the green pathway and the blue one occur only through Pfs, which acts a catalyst in all the reactions in which it is involved.

The largest reduction by forward equivalence that preserves AI 2 has 19 variables. It only aggregates Adenine, MTR, and Spermidine in the same block, while keeping all the other variables separated. This reduction is, however, a trivial one because these are end products of the pathways that do not interact with any other species. Mathematically, this results in the differential equation of an end-product variable not featuring the variable itself in the right-hand side. As a consequence, end-product variables can always be rewritten in terms of a lumped variable that represents their sum. A similar pattern of modular decomposition, on a pathway model of cartilage breakdown from [Proctor et al., 2014], is discussed in Supplementary Materials.

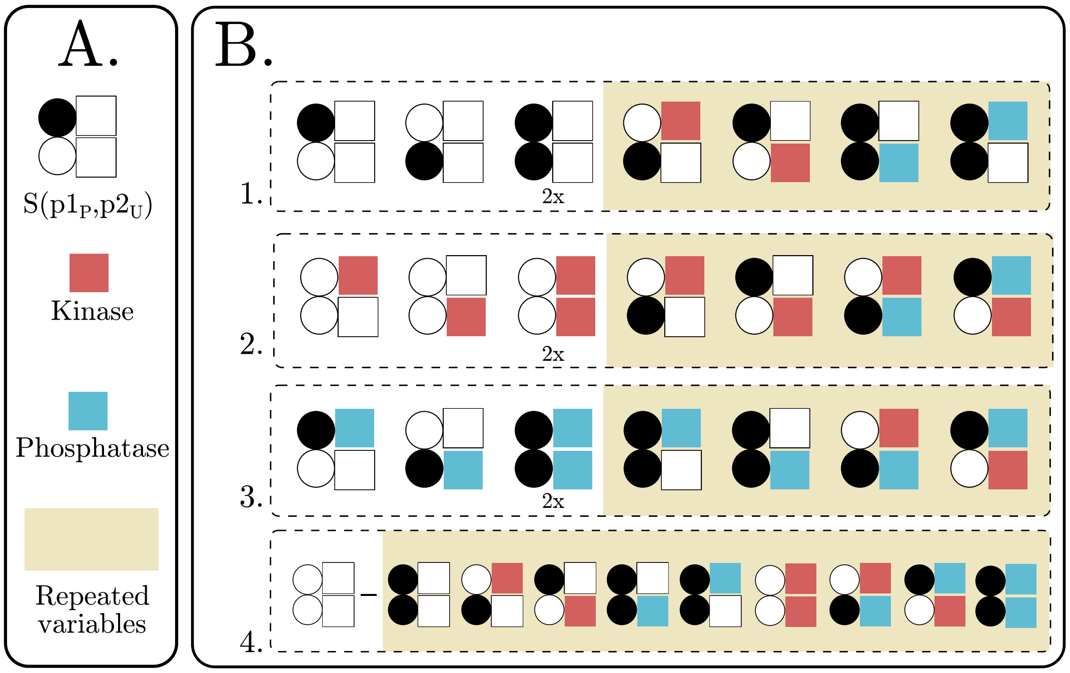

Figure 2: Application of CLUE to a protein phosphorylation model with $m = 2$ identical and independent binding sites. A) Model components: empty/full circles denote whether the site is unphosphorylated/phosphorylated; binding of a kinase or phosphatates is denoted by the color of the square. B) Graphical representation of the four macro-variables obtained by CLUE; the yellow background groups variables that appear in more than one macro-variable; we write '2x' under variables that are counted twice.

4.2 Multisite protein phosphorylation

Here we study a basic mechanism of protein phosphorylation, a fundamental process in eukaryotic cells [Gunawardena, 2005], to show how CLUE can help cope with the combinatorial growth of mechanistic models for proteins with multiple sites [Salazar and Höfer, 2009]. We consider a model phosphorylation/dephosphorylation of a substrate with m independent and identical binding sites, taken from [Sneddon et al., 2011]. Each site can be in four different states: phosphorylated and unbound, unphosphorylated and unbound, phosphorylated and bound to a phosphatase, unphosphorylated and bound to a kinase. Thus, the model is described by $4^m + 2$ variables to track all possible protein configurations, in addition to the concentrations of the free kinase and phosphatase.

For $m = 2$ independent sites, CLUE reduces the model from 18 to 6 variables if observing the free kinase (or the free phosphatase). In the reduced model, 2 macro-variables represent the free kinase and phosphatase, respectively. The other macro-variables are linear combinations of the protein configuration (Fig. 2). An inspection of the aggregation shows that 3 of these macro-variables represent the total concentration of a specific binding-site configuration: free and phosphorylated (Fig. 2-B1); free and bound to a kinase (Fig. 2-B2); phosphorylated and bound to a phosphatase (Fig. 2-B3). Thus, if the two binding sites have the same configuration, the corresponding variable is counted twice. Also, the aggregation results in an improper lumping as there are variables that contribute to more than one macro-variable. The last macro-variable escaped physical intelligibility as it represents the difference between the free substrate with unphosphorylated sites and protein configurations that appear in the aforementioned lumps. Interestingly, running CLUE for models with more binding sites until $m = 8$ always returned a 6-dimensional reduced model that obey the patterns similar to the one from Fig. 2. Instead, forward equivalence detects the assumption of the binding sites being identical [Cardelli et al., 2017a, Table S1]. Each macro-variable represents complexes equal up to permutation of the identities of the binding sites, leading to a polynomial growth in m of the number of variables in the reduced model.

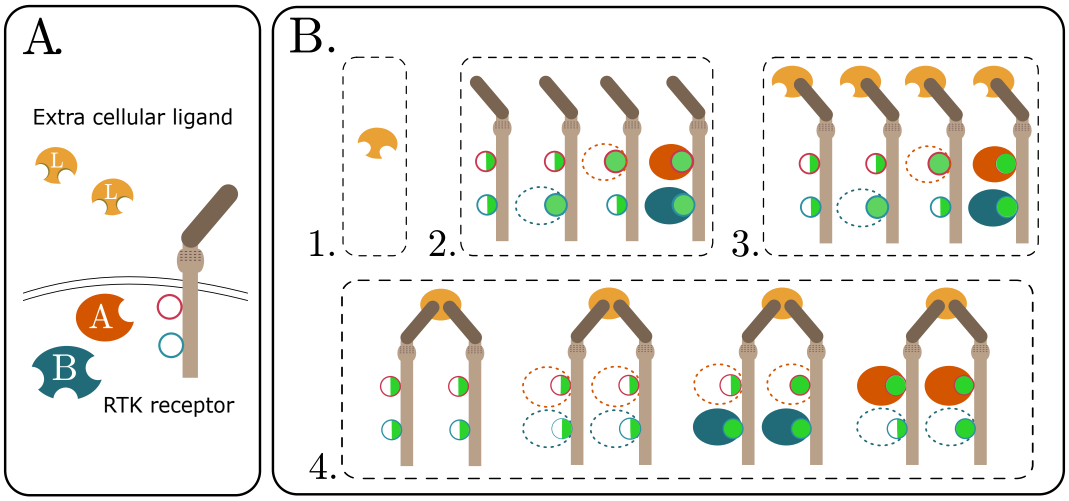

Figure 3: (A) Components in a model for RTK signaling adapted from [Borisov et al., 2008]: a bivalent Ligand (L), 2 adapter proteins A and B and a receptor with 3 binding sites: ligand binding site in the extracellular region (brown); protein binding sites in the intracellular region (red/blue circles). (B) macro-variables obtained in the reduced model.

A different type of aggregation in multisite phosphorylation models is discussed in [Cardelli et al., 2017a] regarding the rule-based model of early events of the signaling pathway of the high-affinity receptor for IgE (FcεRI) in [Faeder et al., 2003]. In this model, the bivalent IgE ligand aggregates FcεRI through phosphorylation of the tyrosine residues on its β and γ subunits by the Lyn kinase, which in turn recruits the tyrosine kinase Syk by phosphorylation of two distinct units. The original model consists of 354 variables, each representing a distinct biochemical complex. In [Cardelli et al., 2017a], the maximal forward equivalence is reported to yield a reduced model with 105 variables in which all complexes that have the same configuration except the phosphorylation state of the Syk units are lumped into the same equivalence class. This model can be further aggregated by CLUE. Observing the concentration of the ligand-receptor complex when both β and γ binding sites are phosphorylated gives a model with 84 macrovariables. This reduction is coarser than the largest forward equivalence because it contains macrovariables that represent linear combinations with negative coefficients or with positive coefficients different than one. Because of this, the reduction cannot be found using specific methods for rule-based systems [Borisov et al., 2005, Conzelmann et al., 2006, Feret et al., 2009]. An inspection of the lumping matrix reveals macrovariables that may carry a physical interpretation. Specifically, it is possible to identify three classes of aggregation. Each class consists of macrovariables that are sums of three original variables, each representing a biomolecular complex consisting of both receptors bound to the bivalent ligand. Across original variables within a macrovariable, one receptor is found in each of the following forms: bound to Syk, phosphorylated at the γ subunit, or unphosphorylated at the γ subunit. The three classes of aggregation are characterized by the state of the other receptor, which is the same within each macrovariable. In particular, there are 7 macrovariables in which the other receptor has a phosphorylated and unbound γ site; 3 macrovariables in which the γ site is phosphorylated and bound to Syk; and 6 macrovariables in which the γ site is unbound and unphosphorylated.

4.3 Aggregation for ordered phosphorylation mechanisms

We now consider an example of ordered phosphorylation, taken from [Borisov et al., 2008], in a receptor tyrosine kinase (RTK) signaling pathway where receptor autophosphorylation via dimerization is preceded by ligand binding. Figure 3(A) shows the molecular complexes involved in the pathway. The receptor interacts with a bivalent ligand and two adapter proteins, A and B. Protein A has a single site that binds to the receptor. Protein B is a scaffold protein with three binding sites: one extracellular site dedicated to receptor-binding and two intracellular tyrosine residues. The phosphorylation state of the tyrosine residues in B is independent of the state of the receptor-binding site. Upon phosphorylation of the intracellular sites, the receptor can bind the adapter proteins A and B.

The model, originally expressed in the rule-based language BioNetGen [Blinov et al., 2004], has 213 variables. Applying CLUE to preserve the concentration of free ligand yields a reduced model where 150 variables are removed and the remaining 63 are lumped into four macro-variables, depicted in Fig. 3-B. These represent: the free ligand (Fig. 3-B1); all configurations of the free receptor regardless of the phosphorylation state of the intracellular terminals or of protein binding (Fig. 3-B2); all variables that represent the bound ligand (Fig. 3-B3) and the dimerized form (Fig. 3-B4) regardless of the intracellular states. Instead, forward equivalence gives 66 variables, aggregating B-bound receptor units regardless of the phosphorylation state of B.

5 Conclusion

We presented CLUE, an algorithm for the reduction of polynomial ODEs by exact lumping, with the possibility to fix original variables (or their linear combinations) to be recovered in the reduced system. From a practical viewpoint, the specification of such constraints allows the preservation of the dynamics of key biochemical species of interest to the modeler. Importantly, although it is acknowledged that linear lumping may lead to loss of structure in the reduced model, e.g., [Snowden et al., 2017], the reductions presented here admitted a biochemical interpretation in most cases. From a computational viewpoint, CLUE casts the analysis of polynomial equations into a linear-algebra framework, allowing reductions for models of dimension over than 15,000 variables using a prototype implementation. This makes CLUE a general-purpose tool that adds to the wide range of existing methods. In particular, since it reduces exactly, it can be used as a pre-processing for techniques that seek more aggressive reductions using approximate methods, or as a complementary method to those that use orthogonal model properties, e.g. time-scale separation.

Funding

This work has been supported by the National Science Foundation [CCF-1563942 and 1564132, DMS-1760448, 1853650, and 1853482], by the MIUR PRIN project “SEDUCE”, no. 2017TWRCNB, and by the Paris Ile-de-France Region.

References

A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. SIAM, 2005.

M. Apri, M. de Gee, and J. Molenaar. Complexity reduction preserving dynamical behavior of biochemical networks. *Journal of Theoretical Biology*, 304(0):16–26, 2012.

A. Babtie and M. Stumpf. How to deal with parameters for whole-cell modelling. *J. of The Royal Society Interface*, 14(133):20170237, 2017.

B. Ballnus, S. Hug, K. Hatz, L. Görlitz, J. Hasenauer, and F. J. Theis. Comprehensive benchmarking of markov chain monte carlo methods for dynamical systems. *BMC Systems Biology*, 11(1), 2017.

M. Blinov, J. Faeder, B. Goldstein, and W. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

N. Borisov, N. Markevich, J. Hoek, and B. Kholodenko. Signaling through receptors and scaffolds: Independent interactions reduce combinatorial complexity. *Biophysical Journal*, 89(2):951 – 966, 2005.

N. Borisov, A. Chistopolsky, J. Faeder, and B. Kholodenko. Domain-oriented reduction of rule-based network models. *IET systems biology*, 2(5):342–351, 2008.

C. Brochot, J. Tóth, and F. Bois. Lumping in pharmacokinetics. *Journal of Pharmacokinetics and Pharmacodynamics*, 32(5):719–736, 2005.

P. Bürgisser, M. Clausen, and A. Shokrollahi. *Algebraic Complexity Theory*. Springer-Verlag Berlin Heidelberg, 1997.

L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Maximal aggregation of polynomial dynamical systems. *PNAS*, 114(38):10029–10034, 2017a.

L. Cardelli, M. Tribastone, A. Vandin, and M. Tschaikowski. ERODE: A tool for the evaluation and reduction of ordinary differential equations. In *TACAS 2017*, volume 10206 of *LNCS*, 2017b.

H. Conzelmann, J. Saez-Rodriguez, T. Sauter, B. Kholodenko, and E. Gilles. A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks. *BMC Bioinformatics*, 7(1):34, 2006.

S. Danø, M. F. Madsen, H. Schmidt, and G. Cedersund. Reduction of a biochemical model with preservation of its basic dynamic properties. *The FEBS Journal*, 273(21):4862–4877, 2006.

V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

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

J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana. Internal coarse-graining of molecular systems. *PNAS*, 106(16):6453–6458, 2009.

J. Gunawardena. Multisite protein phosphorylation makes a good threshold but can be a poor switch. *PNAS*, 102(41):14617–14622, 2005.

C. Li, M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M. Stefan, J. Snoep, M. Hucka, N. Le Novère, and C. Laibe. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4:92, 2010.

G. Li and H. Rabitz. A general analysis of exact lumping in chemical kinetics. *Chemical Engineering Science*, 44(6):1413–1430, 1989.

G. Li and H. Rabitz. New approaches to determination of constrained lumping schemes for a reaction system in the whole composition space. *Chemical Engineering Science*, 46(1):95–111, 1991.

J. Li, L. Wang, Y. Hashimoto, C. Tsao, T. Wood, J. Valdes, E. Zafiriou, and W. Bentley. A stochastic model of escherichia coli AI-2 quorum signal circuit reveals alternative synthesis pathways. *Molecular systems biology*, 2(1), 2006.

A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017.

H. Miao, X. Xia, A. S. Perelson, and H. Wu. On identifiability of nonlinear ODE models and applications in viral dynamics. *SIAM Review*, 53(1):3–39, 2011.

M. Okino and M. Mavrovouniotis. Simplification of mathematical models of chemical reaction systems. *Chemical Reviews*, 2(98):391–408, 1998.

T. Pawson and J. D. Scott. Protein phosphorylation in signaling — 50 years and counting. *Trends in Biochemical Sciences*, 30(6):286–290, 2005.

I. C. Pérez-Verona, M. Tribastone, and A. Vandin. A large-scale assessment of exact model reduction in the BioModels repository. In *Computational Methods in Systems Biology*, pages 248–265. Springer International Publishing, 2019.

C. Proctor, C. Macdonald, J. Milner, A. Rowan, and T. Cawston. A computer simulation approach to assessing therapeutic intervention points for the prevention of cytokine-induced cartilage breakdown. *Arthritis & rheumatology*, 66(4):979–989, 2014.

C. Salazar and T. Höfer. Multisite protein phosphorylation — from molecular mechanisms to kinetic models. *FEBS Journal*, 276(12):3177–3198, 2009.

H. Schmidt, M. Madsen, S. Danø, and G. Cedersund. Complexity reduction of biochemical rate expressions. *Bioinformatics*, 24(6):848–854, 2008.

L. Segel and M. Slemrod. The quasi-steady-state assumption: A case study in perturbation. *SIAM Review*, 31 (3):446–477, 1989.

M. Sneddon, J. Faeder, and T. Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature methods*, 8(2):177, 2011.

T. Snowden, P. van der Graaf, and M. Tindall. Methods of model reduction for large-scale biological systems: A survey of current methods and trends. *Bulletin of Mathematical Biology*, 79(7):1449–1486, 2017.

M. Sunnaker, G. Cedersund, and M. Jirstrand. A method for zooming of nonlinear models of biochemical systems. *BMC Systems Biology*, 5(1):140, 2011.

R. Vallabhajosyula, V. Chickarmane, and H. Sauro. Conservation analysis of large biochemical networks. *Bioinformatics*, 22(3):346–353, 2005.

A. F. Villaverde. Observability and structural identifiability of nonlinear biological systems. *Complexity*, 2019: 1–12, 2019.

J. von zur Garthen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

P. Wang, M. Guy, and J. Davenport. *P*-adic reconstruction of rational numbers. *SIGSAM Bulletin*, 16(2):2–3, 1982.

J. Wei and J. Kuo. Lumping analysis in monomolecular reaction systems. analysis of the exactly lumpable system. *Industrial & Engineering Chemistry Fundamentals*, 8(1):114–123, 02 1969.

Supplementary Materials

CLUE: Exact maximal reduction of kinetic models by constrained lumping of differential equations

Alexey Ovchinnikov, Isabel Pérez Verona, Gleb Pogudin, Mirco Tribastone

This document is structured as follows:

- In Section I, we will prove the correctness and termination the algorithms presented in the paper.
- In Section II, we reprove the criterion for lumping in terms of the Jacobian of the system [Li and Rabitz, 1991, Section 2] for the sake of completeness.
- In Section III, we present a complexity analysis of our algorithms and compare it with the complexity of ERODE.
- In Section IV, we discuss the application of CLUE to a cartilage breakdown model in [Proctor et al., 2014].
- In Section V, we compare the performance of our implementations Algorithms 2 and Algorithm 3.

Remark. In the present paper, we have focused on exact maximal reduction for ODEs with a polynomial right-hand side because our examples are in this class. However, all of our algorithms can be immediately applied to other kinds of systems, such as discrete-time polynomial systems (e.g. $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$).

I Proofs of correctness and termination of algorithms

For the convenience of the reader while navigating between the main paper and the Supplementary materials, we recall:

Algorithm 1 Simplified algorithm for finding a constrained lumping of the smallest possible dimension

```
Input   a system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ of $n$ ODEs with a polynomial right-hand side and an $s \times n$ matrix $A$ over field $\mathbb{K}$
        of rank $s > 0$;
Output  a matrix $L$ such that $\mathbf{y} := L\mathbf{x}$ is a constrained lumping with observables $A\mathbf{x}$ of smallest possible
        dimension.

(Step 1) Compute $J(\mathbf{x})$, the Jacobian matrix of $\mathbf{f}(\mathbf{x})$.
(Step 2) Represent $J(\mathbf{x})$ as $J_1 m_1 + \ldots + J_N m_N$, where $m_1, \ldots, m_N$ are distinct monomials in $\mathbf{x}$, and $J_1, \ldots, J_N$
         are nonzero matrices over $\mathbb{R}$.
(Step 3) Set $L := A$.
(Step 4) Repeat
             (a) for every $M$ in $J_1, \ldots, J_N$ and row $r$ of $L$, if $rM$ does not belong to the row space of $L$, append
                 $rM$ to $L$.
             (b) if nothing has been appended in the previous step, exit the repeat loop and go to (Step 5).
(Step 5) Return $L$.
```

Algorithm 2 Finding the smallest invariant subspace
(to be used instead of (Step 4) of Algorithm 1 for $A = L$, $\ell = N$, and $M_i = J_i$ for $1 \leqslant i \leqslant \ell$)

```
Input  an $s \times n$ matrix $A$ over field $\mathbb{K}$ and a list $M_1, \dots, M_\ell$ of $n \times n$ matrices over $\mathbb{K}$;
Output  an $r \times n$ matrix $L$ over $\mathbb{K}$ such that
    • the row span of $A$ is contained in the row span of $L$.
    • for every $1 \leqslant i \leqslant \ell$, the row span of span of $LM_i$ is contained in the row span of $L$;
    • $r$ is the smallest possible.
(Step 1)  Let $L$ be the reduced row echelon form of $A$.
(Step 2)  Set $P$ be the set of indices of the pivot columns of $L$.
(Step 3)  While $P \neq \varnothing$ do
    (a)  For every $j \in P$ and every $1 \leqslant i \leqslant \ell$
        i.  Let $v$ be the row in $L$ with the index of the pivot being $j$.
        ii.  Reduce $vM_i$ with respect to $L$. If the result is not zero, append it as a new row to $L$.
        iii.  Reduce other rows with respect the new one in order to bring $L$ into the reduced row echelon form.
    (b)  Let $\widetilde{P}$ be the set of indices of the pivot columns of $L$.
    (c)  Set $P := \widetilde{P} \setminus P$.
(Step 4)  Return $L$.
```

Algorithm 3 Finding the smallest invariant subspace (modular)
(to be used instead of (Step 4) of Algorithm 1 for $A = L$, $\ell = N$, and $M_i = J_i$ for $1 \leqslant i \leqslant \ell$)

```
Input  $s \times n$ matrix $A$ and a list $M_1, \dots, M_\ell$ of $n \times n$ matrices over $\mathbb{Q}$;
Output  an $r \times n$ matrix $L$ over $\mathbb{Q}$ such that:
    • the row span of $A$ is contained in the row span of $L$.
    • for every $1 \leqslant i \leqslant \ell$, the row span of $LM_i$ is contained in the row span of $L$;
    • $r$ is the smallest possible.
(Step 1)  Repeat the following
    (a)  Pick a prime number $p$ that does not divide any of the denominators in $A, M_1, \dots, M_\ell$ and has not been chosen before.
    (b)  Compute the reductions $\widetilde{A}, \widetilde{M}_1, \dots, \widetilde{M}_\ell$ modulo $p$.
    (c)  Run Algorithm 2 on $\widetilde{A}, \widetilde{M}_1, \dots, \widetilde{M}_\ell$ as matrices over $\mathbb{F}_p$ and denote the result by $\widetilde{L}$.
    (d)  Apply the rational reconstruction algorithm ([von zur Garthen and Gerhard, 2013, § 5.10], [Wang et al., 1982]) to construct a matrix $L$ over $\mathbb{Q}$ such that the reduction of $L$ mod $p$ equals $\widetilde{L}$.
    (e)  Check whether the row span of $L$ contains the row span of $L$ and is invariant under $M_1, \dots, M_\ell$. If yes, exit the loop.
(Step 2)  Return the matrix $L$ from step (d) of the last iteration of the loop.
```

Notation I.1.

- $\mathrm{Mat}_{m,n}(\mathbb{K})$ denotes the space of $m \times n$ matrices over a field $\mathbb{K}$.
- For $M \in \mathrm{Mat}_{m,n}(\mathbb{K})$, $\mathrm{rspan}_{\mathbb{K}}(M)$ is the row span of M over $\mathbb{K}$.

Lemma I.1 is used by Algorithm 1 to pass from the invariance under the Jacobian to the invariance under a finite set of constant matrices.

Lemma I.1. *Let $M(\mathbf{x}) \in \mathrm{Mat}_{n,n}(\mathbb{K}[\mathbf{x}])$, where $\mathbf{x} = (x_1, \ldots, x_r)$ and $\mathrm{char}\,\mathbb{K} = 0$. We write $M(\mathbf{x}) = M_1 m_1 + \ldots + M_N m_M$ so that $M_1, \ldots, M_N \in \mathrm{Mat}_{n,n}(\mathbb{K})$ and $m_1, \ldots, m_N$ are distinct monomials in $\mathbf{x}$. Then, for a vector subspace $V \subset \mathbb{K}^n$, the following are equivalent:*

(1) V is invariant under $M(\mathbf{x}^*)$ for every $\mathbf{x}^* \in \mathbb{K}^r$;

(2) V is invariant under M_i for every $1 \leqslant i \leqslant N$.

Proof. Assume that V is invariant under $M_1, \ldots, M_N$. Since, for every $\mathbf{x}^* \in \mathbb{K}^r$, $M(\mathbf{x}^*)$ is an $\mathbb{K}$-linear combination of $M_1, \ldots, M_N$, V is invariant under $M(\mathbf{x}^*)$ as well.

Assume that V is invariant under $M(\mathbf{x}^*)$ for every $\mathbf{x}^* \in \mathbb{K}^r$. Consider $v \in V$. Since for all $\mathbf{x}^* \in \mathbb{K}^r$, $M(\mathbf{x}^*)v \in V$, for every $1 \leqslant i \leqslant r$, $\frac{\partial M}{\partial x_i}(\mathbf{x}^*)v \in V$ as well. Consider one of $M_1, \ldots, M_N$, say M_1. Let $m_1 = x_1^{d_1} \ldots x_r^{d_r}$. Iterating the argument with derivative, we obtain

$$\forall \mathbf{x}^* \in \mathbb{K}^r \quad \frac{\partial^{d_1 + \ldots + d_r} M}{\partial x_1^{d_1} \ldots \partial x_r^{d_r}}(\mathbf{x}^*)v \in V.$$

Taking $\mathbf{x}^* = \mathbf{0}$, we deduce that $M_1 v \in V$. $\square$

Remark I.1. A different approach to replacing the Jacobian with a finite set of constant matrices was suggested in [Li and Rabitz, 1989, Sect. 3(A)]:

1. Write the Jacobian $J(\mathbf{x}) = \sum a_{ij}(\mathbf{x}) E_{ij}$, where E_{ij} is the matrix with one in the (i, j)-th cell and zeroes everywhere else;
2. Combine together summands with proportional $a_{ij}(\mathbf{x})$ obtaining a representation $J(\mathbf{x}) = \sum b_j(\mathbf{x}) B_j$ with constant B_j;
3. Return B_j's.

Consider the system

$$\begin{cases} \dot{x}_1 = (x_2 + x_3)^2 + (x_2 + x_4)^2, \\ \dot{x}_2 = \dot{x}_3 = \dot{x}_4 = 0 \end{cases}$$

with the observable x_1. Then the procedure from [Li and Rabitz, 1989, Section 3(A)] will lead to the following decomposition

$$J(\mathbf{x}) = 2(2x_2 + x_3 + x_4)E_{12} + 2(x_2 + x_3)E_{13} + 2(x_2 + x_4)E_{14}.$$

The smallest subspace containing $(1, 0, 0, 0)$ and right-invariant under E_{12}, E_{13}, E_{14} is the whole space, so this approach will not produce a nontrivial lumping. On the other hand, using Lemma I.1, we arrive at

$$J(\mathbf{x}) = 2x_2(2E_{12} + E_{13} + E_{14}) + 2x_3(E_{12} + E_{13}) + 2x_4(E_{12} + E_{14}).$$

The matrices $2E_{12} + E_{13} + E_{14}, E_{12} + E_{13}$, and $E_{12} + E_{14}$ have a common proper invariant subspace containing $(1, 0, 0, 0)$, and this yields a nontrivial lumping:

$$y_1 = x_1, \quad y_2 = x_2 + x_3, \quad y_3 = x_2 + x_4.$$

Proposition I.1. *Algorithm 2 is correct.*

Proof. Bringing a matrix to the reduced row echelon form does not change the row span, and adding extra rows might only enlarge it, so the row span of the output of Algorithm 2 contains the row span of A.

Now we will show that the row span of the output of the algorithm is invariant under $M_1, \dots, M_N$. We denote the values of L and P before the i-th iteration of the while loop **(Step 3)** by L_i and P_i, respectively. We set L_0 and P to be the $0 \times n$ matrix and $\varnothing$, respectively. We will show by induction on k that, for every $k \geqslant 0$ and every $1 \leqslant i \leqslant \ell$, we have

$$\operatorname{rspan}_{\mathbb{K}}(L_k M_i) \subset \operatorname{rspan}_{\mathbb{K}}(L_{k+1}). \tag{1}$$

The case $k = 0$ is true. Assume that the statement is true for all numbers less than some $k > 0$. Let L_+ be the matrix consisting of the rows of L_k with the pivot columns in P_k, and let L_- be the matrix consisting of the remaining rows. Fix $1 \leqslant i \leqslant \ell$. Then $\operatorname{rspan}_{\mathbb{K}}(L_+ M_i) \subset \operatorname{rspan}_{\mathbb{K}} L_{k+1}$ because the rows of L_+ will be processed in the next iteration of the while loop. By the construction, $\operatorname{rspan}_{\mathbb{K}} L_{k-1} \subset \operatorname{rspan}_{\mathbb{K}} L_k$. The rows of L_{k-1} and L_+ are linearly independent because they form a (nonreduced) row echelon form after permuting rows and columns. Therefore, $\operatorname{rspan}_{\mathbb{K}} L_k = \operatorname{rspan}_{\mathbb{K}} L_+ + \operatorname{rspan}_{\mathbb{K}} L_{k-1}$. This implies

$$\operatorname{rspan}_{\mathbb{K}}(L_- M_i) \subset \operatorname{rspan}_{\mathbb{K}}(L_+ M_i) + \operatorname{rspan}_{\mathbb{K}}(L_{k-1} M_i).$$

The inductive hypothesis implies that

$$\operatorname{rspan}_{\mathbb{K}}(L_- M_i) \subset \operatorname{rspan}_{\mathbb{K}}(L_+ M_i) + \operatorname{rspan}_{\mathbb{K}} L_k \subset \operatorname{rspan}_{\mathbb{K}} L_{k+1}.$$

Therefore, $\operatorname{rspan}_{\mathbb{K}}(L_k M_i) \subset \operatorname{rspan}_{\mathbb{K}} L_{k+1}$.

Assume that there were N iterations of the while loop. Then we consider one extra iteration. Since $P = \varnothing$, this iteration will not do anything, so $L_{N+2} = L_{N+1}$. Therefore, $\operatorname{rspan}_{\mathbb{K}}(L_{N+1} M_i) \subset \operatorname{rspan}_{\mathbb{K}}(L_{N+1})$ for every $1 \leqslant i \leqslant \ell$ due to (1). This implies that $\operatorname{rspan}_{\mathbb{K}}$ of the output of the algorithm is invariant under $M_1, \dots, M_\ell$.

To prove the minimality of r, consider V, the smallest subspace of $\mathbb{K}^n$ invariant under $M_1, \dots, M_\ell$ and containing the rows of the input matrix A. We will show by induction on i that $\operatorname{rspan}_{\mathbb{K}}(L_i) \subset V$. Since $\operatorname{rspan}_{\mathbb{K}}(L_1) = \operatorname{rspan}_{\mathbb{K}} A$, $\operatorname{rspan}_{\mathbb{K}}(L_1) \subset V$. Assume that the statement is true for some $i \geqslant 1$. At the i-th iteration of the while loop, we consider vectors of the form vM_i, where $v \in \operatorname{rspan}_{\mathbb{K}}(L_i)$. Since $v \in V$ and V is M_i-invariant, these vectors also belong to V. Consequent computation of the row echelon form does not change the row span. Hence, the row span of the output is invariant under $M_1, \dots, M_\ell$ and contained in V, so it coincides with V. This proves the minimality of r. □

The following lemma is used in Proposition I.2 for showing the correctness and termination of Algorithm 3.

Lemma I.2. *Let $A \in \mathrm{Mat}_{s,n}(\mathbb{Q})$, $M_1, \dots, M_\ell \in \mathrm{Mat}_{n,n}(\mathbb{Q})$ and L the result of applying Algorithm 2 to these matrices. For every prime number p that does not divide the denominators of the entries of $A, M_1, \dots, M_\ell$, we denote the result of applying Algorithm 2 to the reductions of these matrices modulo p by L_p^*. Then*

(1) for all but finitely many primes, L_p^* is equal to L modulo p;

(2) the number of rows in L_p^* does not exceed the number of rows in L.

Proof. To show (1), consider the run of Algorithm 2 on $A, M_1, \dots, M_\ell$. The operations performed with the matrix entries in the algorithm are arithmetic operations and checking for nullity. There is a finite list of nonzero rational numbers $q_1, \dots, q_N$ checked for nullity in the algorithm. Consider a prime number p such that the reductions of $q_1, \ldots, q_N$ modulo p are defined and not zero. Since the arithmetic operations commute with reducing modulo p and we have chosen p so that all nullity checks will also commute with reduction modulo p, the result of the algorithm modulo p, that is L_p^*, will be equal to the reduction of L modulo p.

We now show (1). The number of rows in L is the dimension of the space generated by the rows of A and their images under all possible products of $M_1, \ldots, M_\ell$. Consider the $\infty \times n$ matrix R formed from the matrices of the form AX, where X ranges over all possible products of $M_1, \ldots, M_\ell$, stacked on top of each other. Let R_p be the reduction of R modulo p. For every integer r, having rank at most r can be expressed as a system of polynomial conditions in the matrix entries (that is, all $(r+1) \times (r+1)$ minors are zero). Therefore, $\operatorname{rank} R_p \leqslant \operatorname{rank} R$. Since the numbers of rows in L and L_p^* are equal to $\operatorname{rank} R$ and $\operatorname{rank} R_p$, respectively, the second part of the lemma is proved. $\square$

Proposition I.2. *Algorithm 3 is correct and terminates in finite time.*

Proof. First we will show the correctness. Consider the output of Algorithm 3, call it L_0. Since the stopping criterion for the loop in **(Step 1)** is $\operatorname{rspan}_{\mathbb{Q}}(A) \subset \operatorname{rspan}_{\mathbb{Q}}(L_0)$ and the invariance of $\operatorname{rspan}_{\mathbb{Q}}(L_0)$ under $M_1, \ldots, M_\ell$, it remains to prove the minimality of the number of rows in L_0. Due to Proposition A.1 from the main paper (correctness of Algorithm 2), it would be equivalent to show that the number of rows in L_0 is equal to the number of rows in the output of Algorithm 2 on $A, M_1, \ldots, M_\ell$, call it L. The second part of Lemma I.2 implies that the number of rows of every matrix $\widetilde{L}$ computed in **(Step 1)** does not exceed the number of rows in L. Then the same is true for L_0. Since the number of rows in L is the smallest possible, it is the same as the number of rows in L_0, so the output of the algorithm will be correct.

Now we will prove the termination. Let N be the maximum of the absolute values of the numerators and denominators of the entries of L. Consider a prime number p such that L_p^* (see Lemma I.2) is equal to the reduction of L modulo p and $p > 2N^2$. Then [Wang et al., 1982] and [Wang, 1981, Lemma 2] imply that the result of rational reconstruction in (d) for $\widetilde{L} = L_p^*$ will be equal to L, so the algorithm will terminate. Lemma I.2(1) implies that all but finitely many primes satisfy the above properties, so the algorithm will reach one of these numbers and terminate. $\square$

II Proof for the lumping criterion from Li and Rabitz [1989]

In Lemma II.1 and Proposition II.1, we reprove the criterion for lumping in terms of the Jacobian of the system [Li and Rabitz, 1991, Section 2] for the sake of completeness.

Lemma II.1. *Let $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$, where $\mathbf{x} = (x_1, \ldots, x_n)$, and $L \in \operatorname{Mat}_{s,n}(\mathbb{R})$. Let $V \subset \mathbb{R}^n$ be the orthogonal complement to $\operatorname{rspan}_{\mathbb{R}}(L)$. Then $p(\mathbf{x})$ can be written as a polynomial in $L\mathbf{x}$ if and only if $\forall \mathbf{v} \in \mathbb{R}^n$ the operator $D_{\mathbf{v}} := v_1 \frac{\partial}{\partial x_1} + \ldots + v_n \frac{\partial}{\partial x_n}$ annihilates $p(\mathbf{x})$.*

Proof. Denote the rows of L by $\mathbf{r}_1, \ldots, \mathbf{r}_s$. Assume that there exists a polynomial q in $y_1, \ldots, y_s$ such that $p(\mathbf{x}) = q(L\mathbf{x})$. Then

$$\forall \mathbf{v} \in V \quad D_{\mathbf{v}} p(\mathbf{x}) = D_{\mathbf{v}} q(L\mathbf{x}) = (\mathbf{v}, \mathbf{r}_1) \frac{\partial q}{\partial y_1}(L\mathbf{x}) + \ldots + (\mathbf{v}, \mathbf{r}_s) \frac{\partial q}{\partial y_s}(L\mathbf{x}) = 0.$$

To prove the lemma in the other direction, choose an orthonormal basis $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ of V. Since the rows of L and $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ span the whole space, there exists a polynomial q in $y_1, \ldots, y_{s+\ell}$ such that $p(\mathbf{x}) = q(L\mathbf{x}, (\mathbf{u}_1, \mathbf{x}), \ldots, (\mathbf{u}_\ell, \mathbf{x}))$. Then, for every $1 \leqslant i \leqslant \ell$, using $D_{\mathbf{v}}(\mathbf{u}, \mathbf{x}) = (\mathbf{v}, \mathbf{u})$, we have

$$\begin{aligned} D_{\mathbf{u}_i} p(\mathbf{x}) &= D_{\mathbf{u}_i} q(L\mathbf{x}, (\mathbf{u}_1, \mathbf{x}), \ldots, (\mathbf{u}_\ell, \mathbf{x})) = (\mathbf{u}_i, \mathbf{u}_i) \frac{\partial q}{\partial y_{s+i}}(L\mathbf{x}, (\mathbf{u}_1, \mathbf{x}), \ldots, (\mathbf{u}_\ell, \mathbf{x})) \\ &= \frac{\partial q}{\partial y_{s+i}}(L\mathbf{x}, (\mathbf{u}_1, \mathbf{x}), \ldots, (\mathbf{u}_\ell, \mathbf{x})). \end{aligned}$$

Therefore, q does not involve y_{s+i}, so we get a representation of p as a polynomial in $L\mathbf{x}$. $\square$

Proposition II.1. *A matrix $L \in \mathrm{Mat}_{s,n}(\mathbb{R})$ is a lumping for a n-dimensional system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ if and only if, $\forall \mathbf{x} \in \mathbb{R}^n$, $\mathrm{rspan}_{\mathbb{R}}(L)$ is invariant under $J(\mathbf{x})$, the Jacobian matrix of $\mathbf{f}$.*

Proof. We will use the notation from Lemma II.1. For $\mathbf{v} \in V$,

$$D_{\mathbf{v}} L\mathbf{f}(\mathbf{x}) = \left(\mathbf{v}, \left(\tfrac{\partial}{\partial x_1}, \ldots, \tfrac{\partial}{\partial x_n}\right)\right) L\mathbf{f}(\mathbf{x}) = (LJ(\mathbf{x}))\mathbf{v}.$$

Therefore, Lemma II.1 implies that L is a lumping of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ if and only if $\mathrm{rspan}_{\mathbb{R}}(LJ(\mathbf{x}))$ is orthogonal to V for every $\mathbf{x}$. The latter is equivalent to the invariance of $\mathrm{rspan}_{\mathbb{R}}(L)$ under $J(\mathbf{x})$ for every $\mathbf{x} \in \mathbb{R}^n$. $\square$

III Complexity analysis

In this section, we give upper bounds on the arithmetic complexity (that is, each operation with rational numbers is assumed to have unit cost) of Algorithms 1 and 2 (Propositions III.1 and III.2) and their comparison with the complexity bound of the algorithm implemented in ERODE from [Cardelli et al., 2017, Supporting Information, Theorem 3] (Remark III.3).

As we explain in Section IV, our implementation runs Algorithm 2 first and switches to Algorithm 3 only if it encounters large numbers (more than 10000 digits). For the majority of the models, the switch did not happen. In these cases, the numbers occurring during the computation will have lengths bounded by the constants, so the arithmetic complexity will be the same as the bit-size complexity, and, therefore, can be used to reason about the runtime.

Remark III.1. Before estimating the complexity of the algorithms, we explain the data structures we use for representing vectors and matrices.

- *Vectors.* Each vector is represented by an ordered list of indices of the coordinates with nonzero values and by a hashtable with keys being these indices and the values being the values of the corresponding coordinates.

  For example, the vector $\mathbf{v} = (0,0,3,1,0,0,5,1,0)$ will be represented by the list $(3,4,7,8)$ and hashtable $\{3 \to 3, 4 \to 1, 7 \to 5, 8 \to 1\}$.

  If two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ have n_1 and n_2 nonzero coordinates, respectively, then their sum and inner product can be computed with expected arithmetic complexitites $O(n_1 + n_2)$ and $O(\min(n_1, n_2))$, respectively.

- *Matrices.* Each matrix is represented as a sparse vector (as described above) of its rows represented also as sparse vectors. Then if a matrix M has n nonzero entries, then the product $M\mathbf{v}$ with a sparse vector $\mathbf{v}$ can be computed with expected arithmetic complexity $O(n)$ by computing inner products of $\mathbf{v}$ with the nonzero rows of M,

Proposition III.1. *Let A be a full row rank $s \times n$ matrix over a computable field $\mathbb{K}$ and $M_1, \ldots, M_\ell$ be $n \times n$ nonzero sparse matrices (represented as in Remark III.1) with the total number of nonzero entries being T. Then the expected arithmetic complexity of Algorithm 2 is $O(rn(T + r))$ (this is bounded by $O(n^2(T + n))$ since $r \leqslant n$), where r is the number of rows in the output.*

Proof. We will analyze the complexity step-by-step. The complexity of **(Step 1)** and **(Step 2)** is equal to the complexity of Gaussian elimination, so it can be bounded by $O(s^2 n)$ field operations (similarly to [Trefethen and Bau, 1997, p. 165]). **(Step 3)** involves three different operations: computing matrix-vector products, reducing a vector with respect to the rows of L, and reducing rows of L with respect to a newly added vector.

We will bound the complexities of these steps separately:

- *Matrix-vector multiplications.* The number of vectors v considered in this step does not exceed the number of pivots in the resulting matrix L, which is r. For each such vector, we multiply it by the matrices $M_1, \ldots, M_\ell$. Remark III.1 implies that this can be done in $O(T)$ operations. Thus, the total complexity will be $O(rT)$.

- *Reducing with respect to the rows of L.* Consider the vector v from **(Step 3)**. The total number of nonzero entries in $vM_1, \ldots, vM_\ell$ does not exceed T. Since L is in row reduced echelon form, the total number of elementary row operations used while reducing these vectors with respect to the rows of L will not exceed T. Each such row operation has complexity $O(n)$, so the total complexity for the fixed vector v is $O(nT)$. Since there will be at most r such vectors, the overall complexity is $O(rnT)$.

- *Reducing rows of L with respect to a newly added vector.* There will be $r-s$ newly added vectors. The total number of elementary row operations will be $s+(s+1)+\ldots+r$. Hence, the total complexity will be $O((r^2-s^2)n)$.

Summing up, we obtain

$$O(s^2n)+O(rT)+O(rnT)+O((r^2-s^2)n)=O(rn(T+r)). \qquad \square$$

Proposition III.2. *Consider a system $\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x})$ of n ODEs with polynomial right-hand side. Let*

- *M be the total number of monomials in the right-hand side;*

- *p be the maximal number of different variables occuring in a monomial;*

- *r be the dimension of the reduced system (so $r \leqslant n$).*

*Then the expected arithmetic complexity of Algorithm 1 with **(Step 4)** performed by Algorithm 2 is $O(rn(pM+r))$.*

Remark III.2. If the ODE system represents a chemical reaction network with mass-action kinetics, then n will be the number of species, M will be the number of reactions, and p will be the maximal number of different species among the reactants or products of a reaction.

Proof of Proposition III.2. We will analyze the complexity of **(Step 1)** and **(Step 2)** together. Each monomial in $\mathbf{f}(\mathbf{x})$ will yield at most p nonzero entries in the matrices $J_1, \ldots, J_N$. Therefore, the complexity of constructing these matrices will be $O(pM)$, and the total number of nonzero entries in these matrices will not exceed pM. The complexity of **(Step 3)** is $O(1)$. Now we apply Proposition III.1 to matrices $J_1, \ldots, J_N$, and obtain that the complexity of **(Step 4)** is $O(rn(pM+r))$. The overall complexity will be $O(rn(pM+r))$. $\square$

Remark III.3 (Comparison with ERODE)**.** The complexity of the algorithm implemented in ERODE given by [Cardelli et al., 2017, Supporting infomration, Theorem 3] can be written in the notation of Proposition III.2 as $O(Mp^2\ell\log n) \leqslant O(M^2p^3\log n)$ (this is the worst-case complexity of a deterministic algorithm, so it is also the expected complexity), where ℓ is the number of distinct partial derivatives among the monomials with different signs (we do not use this parameter in our complexity analysis).

Bringing our bound and this bound to a common set of parameters, we get $O(n^2(pM+n))$ and $O(M^2p^3\log n)$, respectively. These bounds indicate that one algorithm can outperform the other one depending on the parameters of the model considered.

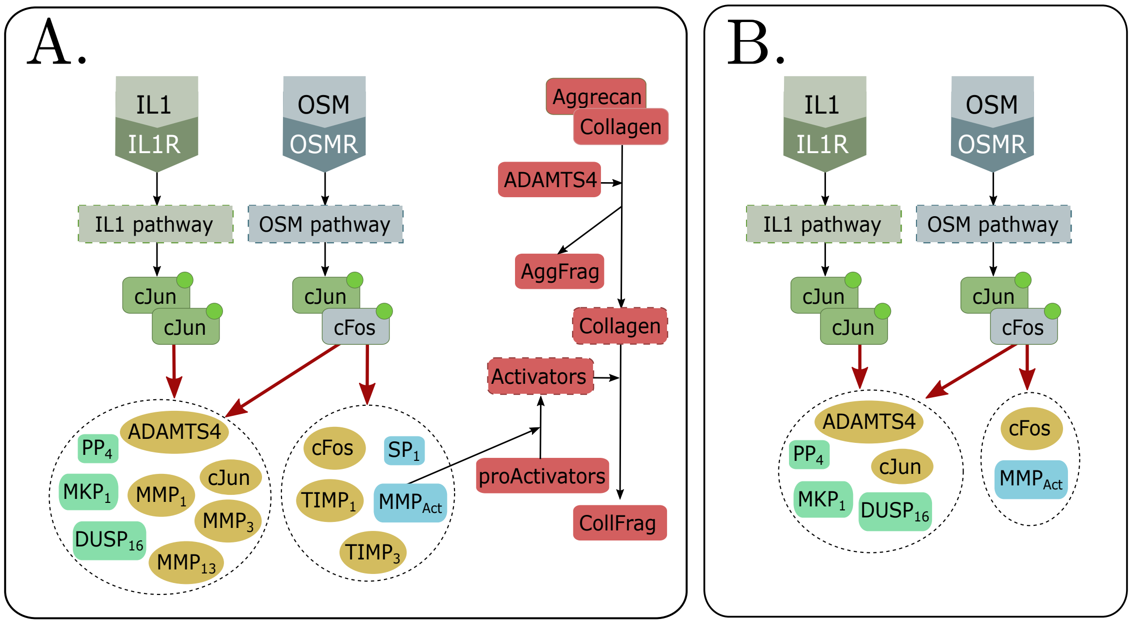

Figure 1: (A) Adaptation of the three molecular pathways from Proctor et al. [2014]. (B) Reduced model obtained while preserving the phosphorylated forms of cJun and cFos. Dotted boxes represent abstractions of groups of biochemical species which are not fully shown here to reduce clutter.

IV Modular decomposition for cartilage breakdown model in Proctor et al. [2014]

This section discusses a pattern of modular decomposition similar to Section 4.1 in the main text, on a model of cartilage breakdown pathway from [Proctor et al., 2014], illustrated in Fig. 1(A). The model is available in the BioModels repository as *BIOMD0000000504*. The system comprises three modules: an Interleukin-1 (IL_1) signaling pathway, an OSM signaling pathway, and a circuit of activation of proMMPs that concludes with the degradation of Aggrecan and Collagen.

In the first module, IL_1 binds its receptor (ISMR) to start a cascade of phosphorylation events (not shown) that activates cJun. After dimerization, cJun upregulates collagenases $MMP_{\{1,3,13\}}$ and phosphatases MKP1, PP44 and DUSP16. In the second module, OSM binds to the receptor OSMR; the pathway concludes with the phosphorylation of cFos. The active cFos can reversibly bind to phosphorylated cJun in a complex cJun-cFos which acts as transcription factor and upregulates the transcription factor SP1, $TIMPs_{\{1,3\}}$, cFos, cJun, a generic $MMP_{\text{Activator}}$ and all the upregulated components from IL1 module. In the third module, the Aggrecan-Collagen complex separates due to the interaction with ADAMTS4, and the units of Aggrecan in the complex transform into fragments (AggFrag). The units of Collagen interact with several Activators (collagenases such as $MMP_{\{1,3,13\}}$ or MMP_{Act}) that destroy the protein structure, producing collagen fragments (CollFrag).

The original model consists of 74 variables. By preserving the phosphorylated molecules of cFos and cJun, which are some of the species of interest in the study by Proctor et al. [2014], CLUE removes the pathway for the decomposition of the Aggrecan-Collagen complex, together with the mRNA variants of $MMP_{\{1,3,13\}}$, $TIMP_{\{1,3\}}$, and SP_1. The reduced model with 43 variables can be interpreted as the network in Fig. 1 (B). Again, CLUE simplifies branches of the pathway that do not affect the dynamics of the observables. The reduction by forward equivalence, instead, collapses only the variables corresponding to the species Aggrecan, AggFrag, Collagen, and CollFrag, providing a model with 71 variables. Differently from the previous example, this block collapses end species (AggFrag and CollFrag) together with an input species (Aggrecan) which is assumed to have no dynamics (i.e., zero derivative), as well as a species (Collagen) that undergoes degradation.

V Comparison of Algorithm 2 and Algorithm 3

As mentioned in the main text, Algorithm 2 is typically faster for simpler cases, while the performance of Algorithm 3 is more robust. The ratios of the runtime of Algorithm 3 and the runtime of Algorithm 2 for an extended set of benchmarks are collected in Table 1 below. The value < 0.01 refers to the fact that Algorithm 2 has been running for 100 times more than the runtime of Algorithm 3 but did not produce any result and has been stopped. The benchmarks are available in the repository https://github.com/pogudingleb/CLUE/tree/master/examples. For three of the models, we had several sets of observables, the indexes of the sets (as listed in the repository) are given in the parenthesis.

From the table, one can see that Algorithm 2 is faster than Algorithm 3 by about a factor of 6 for the majority of given examples. Typically, this happens if the dimension of the reduced model is relatively small or the form of reduction is relatively simple. On the other hand, in the cases in which Algorithm 2 encounters very long integers during the computation (like [Barua et al., 2009] and [Faeder et al., 2003] models), it is likely to get stuck while Algorithm 3 terminates in reasonable yielding to more than 100-fold speed up.

Model	time(Alg. 3) / time(Alg. 2)
Li et al. [2006]	2.5
Proctor et al. [2014] (1)	3.0
Proctor et al. [2014] (2)	4.0
Proctor et al. [2014] (3)	3.2
Proctor et al. [2014] (4)	4.0
Borisov et al. [2008]	6.0
Sneddon et al. [2011], $m = 2$	5.0
Sneddon et al. [2011], $m = 3$	5.0
Sneddon et al. [2011], $m = 4$	6.0
Sneddon et al. [2011], $m = 5$	6.7
Sneddon et al. [2011], $m = 6$	6.9
Sneddon et al. [2011], $m = 7$	6.7
Sneddon et al. [2011], $m = 8$	6.6
Barua et al. [2009] (1)	$<$ **0.01**
Barua et al. [2009] (1)	$<$ **0.01**
Pepke et al. [2010]	4.0
Faeder et al. [2003] (1)	5.2
Faeder et al. [2003] (2)	$<$ **0.01**
Faeder et al. [2003] (3)	5.8
Faeder et al. [2003] (4)	5.6
Faeder et al. [2003] (5)	6.6

Table 1: The ratio of the runtimes of Algorithm 3 and Algorithm 2 for an extended set of benchmarks
The numbers in parenthesis after a reference refer to the index of the chosen set of observables.

In our implementation, these algorithms are combined to benefit from their strengths as follows. We first run Algorithm 2, and if the algorithm encounters very long rational numbers (we use 10000 digits as the threshold), then we stop it and run Algorithm 3 instead. In the most frequent case of not so long rational numbers, the runtime is the same as that of Algorithm 2. In the cases in which using Algorithm 3 is preferable, first trying Algorithm 2 in our implementation adds only a small overhead (less than 10%) compared to running Algorithm 3 by itself.

References

D. Barua, J. R. Faeder, and J. M. Haugh. A bipolar clamp mechanism for activation of jak-family protein tyrosine kinases. *PLoS Comput. Biol.*, 5(4):e1000364, 04 2009. URL http://dx.doi.org/10.1371/journal.pcbi.1000364.

N. Borisov, A. Chistopolsky, J. Faeder, and B. Kholodenko. Domain-oriented reduction of rule-based network models. *IET systems biology*, 2(5):342–351, 2008. URL https://dx.doi.org/10.1049/iet-syb:20070081.

L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Maximal aggregation of polynomial dynamical systems. *PNAS*, 114(38):10029–10034, 2017. URL https://doi.org/10.1073/pnas.1702697114.

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

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

G. Li and H. Rabitz. New approaches to determination of constrained lumping schemes for a reaction system in the whole composition space. *Chemical Engineering Science*, 46(1):95–111, 1991. URL https://doi.org/10.1016/0009-2509(91)80120-N.

J. Li, L. Wang, Y. Hashimoto, C. Tsao, T. Wood, J. Valdes, E. Zafiriou, and W. Bentley. A stochastic model of escherichia coli AI-2 quorum signal circuit reveals alternative synthesis pathways. *Molecular systems biology*, 2(1), 2006. URL https://dx.doi.org/10.1038/msb4100107.

S. Pepke, T. Kinzer-Ursem, S. Mihalas, and M. B. Kennedy. A dynamic model of interactions of ca2+, calmodulin, and catalytic subunits of ca2+/calmodulin-dependent protein kinase II. *PLoS Computational Biology*, 6(2):e1000675, 2010. URL https://doi.org/10.1371/journal.pcbi.1000675.

C. Proctor, C. Macdonald, J. Milner, A. Rowan, and T. Cawston. A computer simulation approach to assessing therapeutic intervention points for the prevention of cytokine-induced cartilage breakdown. *Arthritis & rheumatology*, 66(4):979–989, 2014. URL https://doi.org/10.1002/art.38297.

M. Sneddon, J. Faeder, and T. Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature methods*, 8(2):177, 2011. URL https://doi.org/10.1038/nmeth.1546.

L. N. Trefethen and D. I. Bau. *Numerical Linear Algebra*. SIAM, 1997.

J. von zur Garthen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

P. Wang. A p-adic algorithm for univariate partial fractions. In *Proceedings of SYMSAC'81*, pages 212–217, 1981. URL https://doi.org/10.1145/800206.806398.

P. Wang, M. Guy, and J. Davenport. *P*-adic reconstruction of rational numbers. *SIGSAM Bulletin*, 16(2): 2–3, 1982. URL https://doi.org/10.1145/1089292.1089293.