Faster Gröbner bases for Lie derivatives of ODE systems via monomial orderings

Mariya Bessonov
Department of Mathematics, CUNY
NYC College of Technology
New York, NY, USA
mariya.bessonov@gmail.com

Ilia Ilmer
Ph.D. Program in Computer Science,
CUNY Graduate Center
New York, NY, USA
i.ilmer@icloud.com

Tatiana Konstantinova
Department of Mathematics, CUNY
Queens College
New York, NY, USA
tatiana.v.konst@gmail.com

Alexey Ovchinnikov
Department of Mathematics, CUNY
Queens College; Ph.D. Programs in
Mathematics and Computer Science,
CUNY Graduate Center
New York, NY, USA
aovchinnikov@qc.cuny.edu

Gleb Pogudin
LIX, CNRS, École Polytechnique,
Institute Polytechnique de Paris
Paris, France
gleb.pogudin@polytechnique.edu

Pedro Soto*
Ph.D. Program in Computer Science,
CUNY Graduate Center
New York, NY, USA
pedrosoto@vt.edu

ABSTRACT

Symbolic computation for systems of differential equations is often computationally expensive. Many practical differential models have a form of polynomial or rational ODE system with specified outputs. A basic symbolic approach to analyze these models is to compute and then symbolically process the polynomial system obtained by sufficiently many Lie derivatives of the output functions with respect to the vector field given by the ODE system.

In this paper, we present a method for speeding up Gröbner basis computation for such a class of polynomial systems by using specific monomial ordering, including weights for the variables, coming from the structure of the ODE model. We provide empirical results that show improvement across different symbolic computing frameworks and apply the method to speed up structural identifiability analysis of ODE models.

KEYWORDS

differential algebra, ODE Systems, F4 algorithm, weighted monomial ordering, parameter identifiability, mathematical biology

ACM Reference Format:
Mariya Bessonov, Ilia Ilmer, Tatiana Konstantinova, Alexey Ovchinnikov, Gleb Pogudin, and Pedro Soto. 2024. Faster Gröbner bases for Lie derivatives of ODE systems via monomial orderings. In *Proceedings of ISSAC 2024 (ISSAC '24).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3666000.3669695

*Work was partially done at the Department of Mathematics at Virginia Tech, the Mathematical Institute at the University of Oxford, and the Wellcome Centre for Human Genetics at the University of Oxford.

ISSAC '24, July 16–19, 2024, Raleigh, NC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/3666000.3669695

1 INTRODUCTION

Differential equations are widely used in modeling. Symbolic computation via differential algebra provides a broad range of tools for analyzing such models [30]. However, efficiency has been a significant bottleneck in using such tools. There has been much progress in efficiency for ODE systems with specified output functions by symbolically processing the Lie derivatives of the output functions using Gröbner bases. However, for some particular examples of relatively small ODE systems (even < 10 equations), the computation would not finish in weeks consuming over 100GB of RAM (see, e.g. [11, Table 4] and [22, Table 6.1]).

The Gröbner basis (itself and its computation) of a polynomial system can vary based on the chosen monomial ordering. The most common and empirically reliable in terms of computing time monomial ordering is the so-called total-degree-reverse-lexicographic order, or `tdeg` in Maple notation. Weighted ordering adds a layer of comparison to monomial orderings where one first compares variables by the weight value multiplied by its degree exponent and then breaks ties by applying any applicable monomial rule [18]. Properly chosen weights may have tremendous impact on the computation time. To illustrate this, consider the following motivating example of a well-known benchmark polynomial system, Jason-210 [12]. This example shows benefits of weights in general:

$$P := \begin{cases} x_1^2x_3^4 + x_1x_2x_3^2x_5^2 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_6x_8 + \\ +x_1x_2x_4^2x_6^2 + x_2^2x_4^4, \quad x_2^6, \quad x_1^6 \end{cases} \tag{1}$$

Computing the Gröbner basis of this system with `tdeg`-order of $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ takes approximately *670 seconds* of total CPU time and 26 seconds of total elapsed time (multiple cores were used) as computed in Maple 2021[1]. Modifying the system by assigning a weight of 2 to the variable x_8 results in approximately *2 seconds* of CPU time and only about *1 second* of total elapsed time. Assigning weights of 2 to some of the other variables, e.g. to x_7, result in a speed-up as well.

[1]Computation done on MacBook Pro with 16 GB of RAM and 16-core M1 processor

In this paper, we present a method for significantly speeding up Gröbner basis computation for the class of poylnomial systems that are formed by taking Lie derivatives and all variables and their derivatives are interpreted as indeterminates. Our method is based on a careful automatic selection of a monomial ordering, which is based on the structure of the ODE system. Our orderings are weighted total-degree-reverse-lexicograpic, with the weight assignment following the ODE model structure.

The presented ordering is a result of conducting numerous experiments and analyzing the results. It is thus motivated by empirical observations, just like the fact that the total degree lexicographic ordering is also an empirical achievement accepted universally as the most advantageous monomial order. Proving that the given choice behaves best is outside the scope of our manuscript, and we hope that our work will inspire more investigation in the area. We provide experimental results showing improvements in runtime and memory use for Maple and Magma.

One of the applied contexts in which such Lie derivative computation appears is the parameter identifiability problem. Parameter identifiability is a property crucial for designing high-quality mathematical models of real-world phenomena using ODEs. The question of identifiability arises when one seeks a value for a particular parameter of the model. A parameter can have either finitely many such values (*local* structural identifiability), the value can be unique (*global* structural identifiability), or there may be infinitely many values and the parameter is *unidentifiable*.

Distinguishing between unidentifiable and locally identifiable is rather efficient [39]. On the other hand, knowing only local identifiability in practice is typically insufficient. For example, if one then uses an optimization-based parameter estimation algorithm [3, 13, 23, 26, 36, 38], one typically obtains only one solution for the parameter values even if there are multiple solutions fitting into a physically meaningful range. Knowing whether the system is globally identifiable would give the user a guarantee that the solution returned by the parameter estimation algorithm is unique.

The Gröbner basis computation with Lie derivatives described above, for instance, lies at the core of the global identifiability algorithm SIAN [21, 24, 32] (see further details in Section 3.3), used in [1, 9, 28, 40, 42]. We refer to a recent survey [37, Table 3] showing that SIAN compares favorably to other identifiability software tools. The orderings proposed in the paper allow to speed up global identifiability analysis with SIAN significantly, and are included in the latest release of SIAN [21] and SIAN-Julia [31].

The rest of this paper is organized as follows. In Section 2, we provide an overview of works related to identifiability and Gröbner basis computation. Section 3 describes Gröbner bases and how they appear in the identifiability analysis. Section 4 contains the weight generation algorithm. In Section 5, we show the experimental results and benchmarks with our new weight assignment approach. We conclude in Section 6 with final remarks regarding the work done and future directions of this research.

2 RELATED WORK

The analysis of connection between weights and homogenization of ideals appeared in [17] and later in more detail in [18]. Homogeneous ideals are an intriguing special case of inputs for a Gröbner basis algorithm because of the additional structure [4, Section 10.2] and because it has been proven to lower the overall complexity of the F5 aglrotithm [17, 18]. In the mentioned works, weights were used as a homogenization tool, e.g., there are systems that can be homogenized by raising variables to the power given by a choice of weights. However, we have observed in the motivating example (1) above that a weighted ordering can break homogenization, offering large benefits.

The problem of finding convenient variables orderings for Gröbner bases computation or similar tasks has recently been actively investigated using tools from machine learning [14, 19, 25, 34]. These results typically allow arbitrary input systems and learn a black-box algorithm for choosing the ordering (for some recent work towards explainable ordering choice, see [35]). In this work, we focus on a specific class of input system only, but, for this class, we were able to find a simple human-understandable rule, which incorporates domain-specific information.

3 PRELIMINARIES

3.1 Gröbner bases

We begin by defining *monomial orderings* and *Gröbner bases*.

Definition 1 (Monomial Orderings). *A monomial ordering* $<$ *of a polynomial ring is a total order on the set of monomials such that, for all monomials* $M_1,\ M_2,\ M_3$, *we have:* $1 \leq M_1$ *and* $M_1 < M_2 \implies M_1M_3 < M_2M_3$.

Definition 2 (Gröbner Basis). *Fix a monomial ordering* $<$ *on the polynomial ring* $k[x_1, \ldots, x_n]$. *A subset* $G = \{g_1, \ldots, g_m\}$ *of an ideal* $I \subseteq k[x_1, \ldots, x_n]$ *such that* $G \neq \{0\}$, *is called* Gröbner basis *if*

$$\langle LT(g_1), \ldots, LT(g_m)\rangle = \langle LT(I)\rangle$$

where $LT(g_i)$ *is the leading term of* g_i, $LT(I)$ *are the leading terms of nonzero elements of* I, *and* $\langle LT(I)\rangle$ *is the ideal generated by* $LT(I)$.

3.2 Differential algebra and ODE systems with parameters

In this section, we set up the language we will use to connect ODE systems and polynomial systems.

Definition 3 (Differential rings and fields). *A* differential ring (R, δ) *is a commutative ring with a derivation* $\delta : R \to R$, *that is, a map such that, for all* $a, b \in R$, $\delta(a + b) = \delta(a) + \delta(b)$ *and* $\delta(ab) = \delta(a)b + a\delta(b)$. *A differential ring that is also a field is called* a differential field.

Definition 4 (Differential polynomials and differential ideals). *The* ring of differential polynomials *in the variables* $x_1, \ldots, x_n$ *over a field* K *is the ring* $K[x_j^{(i)} \mid i \geqslant 0,\ 1 \leqslant j \leqslant n]$ *with a derivation defined on the ring by*

$$\delta\left(x_j^{(i)}\right) := x_j^{(i+1)}.$$

This differential ring is denoted by $K\{x_1, \ldots, x_n\}$. *An ideal* I *of a differential ring* (R, δ) *is called a* differential ideal *if, for all* $a \in I$, *we have* $\delta(a) \in I$. *For* $F \subset R$, *the smallest differential ideal containing set* F *is denoted by* $[F]$.

For an ideal I and element a in a ring R, we denote

$$I\colon a^\infty = \{r \in R \mid \exists \ell\colon a^\ell r \in I\}.$$

This set is an ideal in R.

Definition 5 (Model in the state-space form). A model in the *state-space* form is a system

$$\Sigma := \begin{cases} \dot{\mathbf{x}} & = \mathbf{f}(\mathbf{x},\ \boldsymbol{\mu},\ \mathbf{u}), \\ \mathbf{y} & = \mathbf{g}(\mathbf{x},\ \boldsymbol{\mu},\ \mathbf{u}), \\ \mathbf{x}(0) & = \mathbf{x}^*, \end{cases} \tag{2}$$

where $\mathbf{f} = (f_1, \ldots, f_n)$ and $\mathbf{g} = (g_1, \ldots, g_m)$ with $f_i = f_i(\mathbf{x},\ \boldsymbol{\mu},\ \mathbf{u})$, $g_i = g_i(\mathbf{x},\ \boldsymbol{\mu},\ \mathbf{u})$ are rational functions over the complex numbers $\mathbb{C}$. The vector $\mathbf{x} = (x_1, \ldots, x_n)$ represents the time-dependent state variables and $\dot{\mathbf{x}}$ represents the derivative. The vectors $\mathbf{u} = (u_1, \ldots, u_s)$, $\mathbf{y} = (y_1, \ldots, y_m)$, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_\lambda)$, and $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ represent the input variables, output variables, parameters, and initial conditions, respectively.

The analytic notion of identifiability [22, Definition 2.5] is equivalent (see [22, Proposition 3.4] and [33, Proposition 4.7]) to the following algebraic definition.

We write $\mathbf{f} = \frac{\mathbf{F}}{Q}$ and $\mathbf{g} = \frac{\mathbf{G}}{Q}$, where $\mathbf{F}$ and $\mathbf{G}$ are tuples from $\mathbb{C}(\boldsymbol{\mu})[\mathbf{x}, \mathbf{u}]$ and Q is the common denominator of $\mathbf{f}$ and $\mathbf{g}$. Define the following differential ideal, where we use $\dot{}$ in place of δ.

$$I_\Sigma := [Q\dot{x}_1 - F_1, \ldots, Q\dot{x}_n - F_n, Qy_1 - G_1, \ldots, Qy_m - G_m]\colon Q^\infty, \tag{3}$$

which is in $\mathbb{C}(\boldsymbol{\mu})\{\mathbf{x}, \mathbf{y}, \mathbf{u}\}$. Observe that every solution of (2) is a solution of I_Σ.

Definition 6 (Generic solution). *A tuple $(\mathbf{x}_s, \mathbf{y}_s, \mathbf{u}_s)$ from a differential field $K \supset \mathbb{C}(\boldsymbol{\mu})$ is called* a generic solution *of* (2) *if, for every differential polynomial $P \in \mathbb{C}(\boldsymbol{\mu})\{\mathbf{x}, \mathbf{y}, \mathbf{u}\}$,*

$$P(\mathbf{x}_s, \mathbf{y}_s, \mathbf{u}_s) = 0 \iff P \in I_\Sigma.$$

Definition 7 (Identifiability). *Let $\mathbb{C}(\boldsymbol{\mu})$ be a field of functions in $\boldsymbol{\mu}$ with complex coefficients. A function (or parameter) $h \in \mathbb{C}(\boldsymbol{\mu})$ is said to be identifiable (or globally identifiable) in model* (2) *if, for every generic solution $(\mathbf{x}_s, \mathbf{y}_s, \mathbf{u}_s)$ of ODE* (2)*, it follows that $h \in \mathbb{C}(\boldsymbol{\mu}, \mathbf{y}_s, \dot{\mathbf{y}}_s, \ldots, \mathbf{u}_s, \dot{\mathbf{u}}_s, \ldots)$. The function h is said to be locally identifiable if h is algebraic over the field $\mathbb{C}(\boldsymbol{\mu}, \mathbf{y}_s, \dot{\mathbf{y}}_s, \ldots, \mathbf{u}_s, \dot{\mathbf{u}}_s, \ldots)$.*

3.3 Lie-derivative algorithm for parameter identifiability in ODE models

We will now summarize (following [21]) how the class of polynomial systems we consider can be computed from ODE models (2):

(1) The original differential system (2) is transformed into a polynomial system in the functions' derivatives and parameters through successive differentiation of the original equations.
(2) Random values are sampled for $\mathbf{x}^*, \boldsymbol{\mu}$ and the derivatives of u_i's. Then the corresponding values of the derivatives of y_i's are computed. Finally, the values for y_i's and u_i's are plugged into the polynomial system. This corresponds to sampling a random input-output pair for the model.

After these steps, the polynomial system is, for instance, ready for an immediate use for the parameter identifiability problem for the ODE system using Gröbner bases. In particular, we then check whether the sampled values $\mathbf{x}^*, \boldsymbol{\mu}$ are the only possible solutions of the specialized polynomial system. If yes, the corresponding parameter is globally identifiable. Due to random sampling, this algorithm may produce incorrect results, but the probability of correctness can be made arbitrarily close to 1 by choosing an appropriate sampling range [22, Theorem 4.2].

The aforementioned Gröbner basis computation is typically the bottleneck in the identifiability analysis. We would like to emphasize two features of this computation:

- The uniqueness of the value of a coordinate can easily be checked using a Göbner basis with respect to *any ordering*.
- The resulting Gröbner basis is typically *simple*, e.g., for a globally identifiable system, it defines a maximal ideal.

Notice that, for simplicity of presentation, from this point on, we do not separately discuss input variables $\mathbf{u}$.

3.4 Toy example

In this section, we will show how a concrete (toy) ODE model is transformed into a polynomial system, which will further be a subject for Gröbner basis computation. Consider the following ODE model in state-space form:

$$\Sigma = \begin{cases} \dot{x} = ax + c^2 \\ y = x, \\ x(0) = x^* \end{cases} \tag{4}$$

Using the Jacobian-based termination criterion [21, Theorem 3.16 and Proposition 3.20], which is not relevant in the context of this paper, we will differentiate the first and the second equations one and two times, respectively. As a result, the following polynomial system will be obtained:

$$E^t = \left[y - x^*, \dot{x} - ax^* - c^2, \dot{x} - \dot{y}, \ddot{x} - a\dot{x}, \ddot{x} - \ddot{y}, \dddot{x} - a\ddot{x}, \dddot{y} - \dddot{x}\right].$$

Here, the superscript t stands for "truncated", which is the wording we use to represent the fact that some of the equations were differentiated a smaller number of times (but still sufficiently many) than one would naively do.

Then, to restrict to a random output trajectory, we randomly sample values for x^* and a, substitute them into E^t and solve the resulting system for $y, \dot{y}, \ddot{y}, \dddot{y}$ (the solution will be unique thanks to the triangular shape of the system). We will denote this solution by $\hat{Y} := [\hat{y}_0,\ \hat{y}_1,\ \hat{y}_2]$. Then we substitute the solution into E^t obtaining $\widehat{E}^t$. For a sample of $a = 119791, x^* = 139697, c = 75091$, we have:

$$\widehat{E}^t = \begin{cases} 139697 - x^*, \dot{x} - ax^* - c^2, \\ -\dot{x} + 22373101608, -a\dot{x} + \ddot{x}, \\ -\ddot{x} + 2680096214723928, \dddot{x} - a\ddot{x}, \\ -\dddot{x} + 321051405657994059048 \end{cases} \tag{5}$$

Then comes the *key step* of symbolically processing this polynomial system to determine the property of the ODE system, global identifiability of the parameters. This is done by computing a Gröbner basis of (5) (the ordering does not matter). A parameter/initial condition of the ODE model is globally identifiable if and only if, modulo the basis, it reduces to a constant. For example, for (5), we

obtain a basis

$$\mathcal{B} = \begin{cases} a - 119791, -139697 + x^*, \\ \dot{x} - 22373101608, \ddot{x} - 2680096214723928, \\ \ddot{x} - 321051405657994059048, c^2 - 5638658281. \end{cases}$$

Notice that we have $a - 119791$, $x^* - 139697$ in the basis, so the reductions of a and x^* will be constants yielding that these parameters are globally identifiable. At the same time, we do not have a unique value for c thus concluding that it is *only locally identifiable*.

Our goal: find a weight assignment to each variable of the polynomial system $\widehat{E}^t$ to speed up the Gröbner basis computation. In the example above, the weighted ordering would be applied before the step of computing the Gröbner basis, but after we generate a sampled system $\widehat{E}^t$. We provide more technical details about how exactly this is performed in Section 5.

4 MAIN RESULT

4.1 The monomial ordering.

The monomial ordering we propose compares two monomials first by their weights (we describe the weight assignment below) and then breaks ties by the reverse-lexicographic ordering in which the variables are first compared with respect to their derivative order , e.g., $x < \dot{x} < \ddot{x} < \dots$ (and then any ordering could be used, we used reverse alphabetical order, e.g., $\dot{z} > x > z$).

Now we describe the key component of our monomial ordering, the *weight assignment*. Given a system (2), one can define *the Lie derivative* $\mathcal{L}(h)$ of a function $h \in \mathbb{C}(\mathbf{x}, \boldsymbol{\mu}, \mathbf{u}, \mathbf{u}', \dots)$ with respect to the system by

$$\mathcal{L}(h) = \sum_{i=1}^{n} f_i \frac{\partial h}{\partial x_i} + \sum_{j=1}^{s} \dot{u}_j \frac{\partial h}{\partial u_j}. \tag{6}$$

By applying this formula to each output function y_i, we can define, for each state variable or a parameter $a \in \{\mathbf{x}, \boldsymbol{\mu}\}$, the *level* as

$$\boxed{\text{Level}(a) := \min_i \left[\exists\, y_j \in \mathbf{y} \colon a \text{ appears in } \mathcal{L}^i(y_j)\right]}. \tag{7}$$

Using that value, we assign weight as follows:

- for a state variable $x_i \in x$ (and all its derivatives)

$$\text{Weight}(x_i) := \text{Level}(x_i) + 1; \tag{8}$$

- for a parameter $\mu_i \in \boldsymbol{\mu}$:

$$\text{Weight}(\mu_i) := \begin{cases} \text{Level}(\mu_i) + 1, \text{ if Level}(\mu_i) = \max\limits_{e \in \boldsymbol{\mu} \cup \mathbf{x}} \text{Level}(e), \\ 1, \text{ otherwise.} \end{cases} \tag{9}$$

4.2 Example

Consider the following ODE system

$$\Sigma = \begin{cases} \dot{x}_1 = a x_1 + b x_2, \\ \dot{x}_2 = c x_1, \\ y_1 = x_1. \end{cases} \tag{10}$$

Differentiating once:

$$\mathcal{L}(y_1) = \mathcal{L}(x_1) = a x_1 + b x_2.$$

We see that a, b, x_2 all occur after the first differentiation and hence will have level of 1. At the same time, state x_1 was already at level 0 and will not be considered further. If we differentiate once more, we get

$$\mathcal{L}(\mathcal{L}(y_1)) = a\mathcal{L}(x_1) + b\mathcal{L}(x_2) = a(a x_1 + b x_2) + c x_2,$$

bringing out c. Differentiating further leads to no new information. The final weight assignment then is as follows:

$$x_1 \Rightarrow 1, \;\; x_2 \Rightarrow 2$$

4.3 Do our weights homogenize the system?

We discussed earlier how it has been shown that polynomial systems benefit from homogenizing weight assignment (see [17] and [18]). One may be tempted to hypothesize that homogenization would be the explanation behind the speed-up, but this does not happen because our systems are rarely homogeneous; we instead offer the hypothesis that avoiding reductions to zero, which we observed in our experiments, is the more likely cause of the speed-up, see Section 4.4. Polynomial systems obtained by Lie derivatives in ODE models contain non-homogeneous polynomials in most cases by the nature of the problem statements and approach. For example, consider an output function (see Definition 5) of the form

$$y_i = g_i(\dots), \tag{11}$$

where g_i is a polynomial. Since polynomial elimination typically significantly speeds up after reducing the number of variables keeping the rest the same, the next step we take is to replace the y-variables with numbers, such as in (5). This way, (11) is inevitably transformed into an equation with a free term of degree 0. Therefore, the polynomial systems from the class we consider always have a non-homogeneous polynomial.

By design of our weight-assignment algorithm (9), the weight of any variable in g_i will be 1 , since the variables of g_i are exaxtly the base case of i in Equation 7. For other polynomials that do not have a free term and may be homogeneous, the maximum possible degree in the system will either increase or remain the same because we raise variables to the power of their weight similarly to the procedure described in [18]. In this sense, we do not necessarily make polynomials "more homogeneous" with our weight assignment.

4.4 Possible rationale behind the weight assignment

While the idea to use differential rev-lex ordering can be motivated by results in monomial ideals [43], the mechanism behind the weight assignment seems to be more mysterious. In this section, we propose an explanation why the weight assignment eqs. (8) and (9) speeds up the computation.

We start with a brief overview of the F4 algorithm [15]. The original Buchberger algorithm [7] iteratively picks a pair of polynomials f, g from the already computed set and computes their S-polynomial

$$\begin{aligned} S(f,g) &:= \frac{M(f,g)}{\text{LT}(f)} f - \frac{M(f,g)}{\text{LT}(g)} g, \\ M(f,g) &:= \text{lcm}(\text{LM}(f), \text{LM}(g)), \end{aligned} \tag{12}$$

where LM and LT stand for the leading monomial and leading term, respectively. Then $S(f,g)$ is reduced with respect to already computed polynomials and the result, if nonzero, is added to the computed set. The key idea of the F4 algorithm by [15] is to select several S-polynomial at each step and then reduce them simultaneously using linear algebra. This is done by constructing a matrix from the S-polynomials and all the multiples of the already computed polynomials which could be used in the reduction as follows: the columns correspond to the monomials appearing in at least one of the polynomials, so every polynomial can be transformed into a row in such a matrix. We would like to point out two features of the algorithm important for our discussion:

- The way a set of S-polynomials is chosen at each step may have dramatic impact on the performance of the algorithm. A popular approach is *the normal strategy* [15, p. 73] which takes all pairs for which the *formal degree*, $\deg M(f,g)$ (see (12)), is the minimal possible.
- The matrix is highly structured, in particular, the part containing the reducers (that is, not the S-polynomials) is by construction in a row echelon form and often has block-triangular shape. Therefore, the time for reducing such a matrix may depend more on the number of S-polynomials rather than the total number of rows in the matrix.

The S-polynomials which are reduced to zero can be considered as "waste of time". *Avoiding reductions* to zero is a recurring theme in the Gröbner bases computation, including the Buchberger criterion, F5 algorithm by [16], and connections to regular sequences [41, Section 2.4.3]. We believe that one can explain the performance gains achieved by our weight assignment within this framework, although not directly through casting the system into a regular one.

Polynomial systems produced by our Lie derivative process for ODE systems typically have relatively small and simple Gröbner bases, so one may expect that few reductions are necessary. On the other hand, the number of equations is large (starting with 40 in real examples) and the degrees are low (may not go beyond 3 in many applications). Therefore, the normal selection strategy may select too many pairs at once, yielding a large number of zero reductions. We claim that our weight assignment alleviates this issue by spreading possible values of $\deg M(f,g)$ used for selecting pairs. Let us explain this in more detail using the following model as an example

$$y = x_1, \tag{13}$$

$$x_1' = -ax_1^2 + 2bx_2, \tag{14}$$

$$x_2' = ax_1^2 - bx_2 - cx_2. \tag{15}$$

This system corresponds to a chemical reaction network with two species X_1 and X_2 and reactions:

$$2X_1 \xrightarrow{a} X_2,\ X_2 \xrightarrow{b} 2X_1,\ X_2 \xrightarrow{c} \varnothing.$$

Variables x_2 and c will be assigned a weight of 2 and 3, respectively and all the others will be assigned a weight of 1.

By default, the algorithm will order variables "alphabetically" $x_1 > x_2$. If we do not use weights, then the leading monomial of $(13)^{(i)}$, the i-th derivative of (13), will be $x_1^{(i)}$ while the leading monomials of $(14)^{(i)}$ and $(15)^{(i)}$ will be equal and come from $(ax_1^2)^{(i)}$ because these monomials will have higher total degree. Now we consider the pairs of small formal degree. In degree two, we will only have S-polynomials of derivatives of (13), which will be reducible to zero because the leading monomials in the pairs forming the S-polynomials are relatively prime (Buchberger's criterion). Nontrivial pairs start with degree three, and there will be many of them, including

- simple S-polynomials such as $S((13),(14))$, which basically correspond to plugging the known value for one of the x_1's inside x_1^2 in (14) , i.e., $S((13),(14)) = ax_1y + x_1' - 2bx_2$;
- less trivial S-polynomials such as $S((14)^{(i)},(15)^{(i)})$, which do not have such an immediate interpretation.

In the weighted case, the leading monomial of $(13)^{(i)}$ and $(14)^{(i)}$ will stay the same, while the leading monomial of $(15)^{(i)}$ will become $cx_2^{(i)}$. This will change the situation significantly because the only remaining pairs of the formal degree 3 (earlier defined as $\deg M(f,g)$) will be the natural ones corresponding to plugging the known values of x_1 and its derivatives to (14) while considering more complicated S-polynomials is postponed. As a result, the maximal number of pairs selected at a single iteration of F4 reduces from 20 in the no-weights case to 10 (see also Table 1).

Generalizing this example, one can observe that the weight assignments attempt to force the variables of high level to appear in the leading monomials of the corresponding equations, thus making the pairs involving these equations to be of higher formal degree. This will avoid selecting too many pairs at once and steer the computation towards first working with the variables and equations of small level only, thus taking advantage of the known y-values. We suspect this heuristic is particularly well-suited for ODEs in the form of Eq. 2 since the outputs are truly modeling "known" output values from real world applications. In particular, both globally identifiable and parameter estimation algorithms set the output functions equal to a number, and the heuristic for system solving would be to solve for unknown quantities that are "close" to the known quantities first; our weight assignments precisely measure this "closeness".

We used `msolve` [5] to check whether using weights indeed reduces the maximal number of pairs selected at the same time and the number of zero reductions. Thanks to being open source, `msolve` allows us to extract all information of interest easily. The results are given in Table 1 and confirm our expectations. We also double-checked smaller examples using our own basic implementation of F4 in Maple[2], and observed the same phenomena.

5 EXPERIMENTAL RESULTS

In this section, we present several examples of ODE systems, for which we observe reduction in both the runtime and memory. All simulations were run on a cluster with 64 Intel Xeon CPU with 2.30GHz clock frequency and 755 GB RAM. We ran the computation using Maple and Magma computer algebra systems.

The original SIAN algorithm [22] computed Gröbner bases over rationals. However, many popular F4 implementations (including

[2]available at https://github.com/iliailmer/BasicF4Algorithm

	Max. # of pairs selected		# of zero reductions	
Model	No weights	Weights	No weights	Weights
(13)-(15)	20	10	21	19
(20)	25451	3472	42857	20581
(26)	34570	2731	59804	11546
(27)	10370	2021	27972	8953
(24)	10555	6653	27795	18102

Table 1: F4 statistics on benchmarks with/without weights

the one in Maple) are multimodular, that is, the actual computation is in fact performed modulo several prime numbers, and then the result is lifted to the rationals. To reduce the uncertainty related to different possibilities of the choices of primes and focus on the performance gains of the proposed weighted ordering, we run all experiments modulo a fixed prime number $p = 11863279$. We have conducted additional experiments to verify that the speedup is similar for the computations over rationals.

Maple does not directly support the use of weighted orderings with a compiled F4 implementation that is sufficiently fast. To avoid any potential slowdowns, we substitute any variable v in the polynomial system that has weight w greater than 1 with v^w. To illustrate this, if we have a polynomial system $E = \{x+y, x-y\}$, and we wish to use the weight of 2 for variable x, our approach is to compute the basis for a new polynomial system $E_1 = \{x^2+y, x^2-y\}$ keeping the variable order as total degree reverse lexicographic. Empirically, there may be a difference observed between computing Gröbner basis with $x > y$ and $y > x$. In our computations, we order the variables by the degree of the derivative. For example, consider a simple input ODE model

$$\begin{cases} \dot{x}_1 = ax_1, \\ \dot{x}_2 = -bx_1 + cx_2, \\ y = x_1 + x_2. \end{cases} \tag{16}$$

We then produce the following polynomial system, where the double index in $x_{i;j}$ shows that the variable is the j-th derivative of x_i in jet-notation.

$$E = \begin{cases} 7828371 - x_{1;0} - x_{2;0}, -ax_1 + x_{1;1}, \\ bx_1 - cx_2 + x_{2;1}, -x_{1;1} - x_{2;1} + 22382588610034, \\ -ax_{1;1} + x_{1;1}, bx_{1;1} - cx_{2;1} + x_{2;2}, \\ -x_{1;1} - x_{2;2} + 98741152216384012556, \\ -ax_{1;1} + x_{1;3}, bx_{1;1} - cx_{2;2} + x_{2;3}, \\ -x_{1;3} - x_{2;3} + 53800518036300051751092314 4, \\ -ax_{1;3} + x_{1;4}, bx_{1;3} - cx_{2;3} + x_{2;4}, \\ -x_{1;4} - x_{2;4} + 31270158213516309840633850303387 36 \end{cases} \tag{17}$$

the order of variables for the best speed without weights is

$$x_{2;4}, x_{1;4}, x_{2;3}, x_{1;3}, x_{2;2}, x_{1;2}, x_{2;1}, x_{1;1}, x_{2;0}, x_{1;0}, a, b, c. \tag{18}$$

That is, we use differential deg-rev-lex ordering which orders variables from higher to lower derivative grouping the same degree together (all order-4 derivatives, all order-3, etc.).

In what follows, we apply the weights on top of the default variable ordering (18) that has proven itself to be empirically faster.

We will consider several ODE models and provide Gröbner basis results over a field of integers with positive prime characteristic $p = 11863279$. Each example will be summarized by the following metrics in Tables 2 and 3:

(1) Number of polynomials and variables in the polynomial system.
(2) Default (without weights) CPU time (min) and memory (GB).
(3) CPU time (min) and memory (GB) with weights.
(4) Speedup calculated as $\frac{\text{old time}}{\text{new time}}$.
(5) Memory improvement calculated as $\frac{\text{old memory}}{\text{new memory}}$.

Once the Gröbner basis computation is finished, the weights are removed by a back substitution to answer the identifiability query.

Model information			Time (min)				Memory (GB)			
Model name	num. polys.	num. vars.	old SIAN ordering	differential degrevlex	our final ordering	speedup	old SIAN ordering	differential degrevlex	our final ordering	reduction
COVID Model 2, (7.5)	49	48	N/A	N/A	602.0	∞	N/A	N/A	23.2	∞
Pharmacokinetics, (23)	48	47	N/A	N/A	21.0	∞	N/A	N/A	7.7	∞
HPV, (28), (29)	97	92	N/A	N/A	13.9	∞	N/A	N/A	3.7	∞
HPV, (28), (30)	79	75	N/A	N/A	5.1	∞	N/A	N/A	11.0	∞
COVID Model 1, (21)	51	50	377.0	321.9	1.0	327.6	15.3	15.2	0.3	52.6
Goodwin Oscillator (19)	42	43	44.1	29.8	1.5	18.9	10.8	10.6	0.7	14.6
SEIR-1, (26)	44	45	3.5	2.2	0.1	17.4	3.3	3.3	0.1	44.8
NF-κB, (24)	120	109	10.6	7.1	2.3	3.0	11.8	6.1	3.1	1.9
SEIRP, (20)	50	42	2.6	2.0	0.8	2.5	1.0	1.6	0.2	8.5
SEIR-2, (27)	44	43	1.3	0.8	0.4	2.2	0.8	0.7	0.1	6.1

Table 2: Results of applying the weighted ordering to only Gröbner basis computation step of Lie derivative processing (SIAN algorithm), with characteristic $p = 11863279$ using Maple 2021.2. We compare three monomial orderings: originally used in SIAN, differential degrevlex, and our main weighted ordering. "N/A" stands for the Maple error "`Error, (in Groebner:-F4:-GroebnerBasis) numeric exception: division by zero`" without clear direct cause.

Model information			Time (min)				Memory (GB)			
Model name	num. polys.	num. vars.	old SIAN ordering	differential degrevlex	our final ordering	speedup	old SIAN ordering	differential degrevlex	our final ordering	reduction
COVID Model 2, (7.5)	49	48	4000.6	3471.2	517.4	6.7	38.6	36.4	21.6	1.7
Pharmacokinetics, (23)	48	47	757.6	248.3	44.5	5.6	14.7	8.4	10.7	0.8
HPV, (28), (29)	97	92	321.7	126.6	51.5	2.4	21.4	9.8	18.6	0.5
HPV, (28), (30)	79	75	6.8	5.9	3.2	1.4	2.7	2.6	2.4	1.1
COVID Model 1, (21)	51	50	1331.1	1272.1	0.6	2207.9	9.2	8.7	1.8	4.8
Goodwin Oscillator (19)	42	43	26.9	22.4	0.8	28.5	3.5	3.1	0.5	6.0
SEIR-1, (26)	44	45	8.6	3.9	0.1	76.0	2.0	2.0	0.2	9.8
NF-κB, (24)	120	109	14.6	9.1	1.7	5.2	3.3	2.0	0.6	3.5
SEIRP, (20)	50	42	10.0	6.8	36.5	11.2	1.4	1.3	1.6	0.8
SEIR-2, (27)	44	43	3.4	1.2	0.2	7.6	2.0	1.2	0.7	1.6

Table 3: Results of applying the weighted ordering to only Gröbner basis computation step of Lie derivative processing (SIAN algorithm) with characteristic $p = 11863279$ in Magma 2.26-8. We compare three underlying monomial orderings: originally used in SIAN, differential degrevlex, and our main weighted ordering.

6 CONCLUDING REMARKS

We presented an approach to automatically choose a weighted monomial ordering for Gröbner basis computation for a class of polynomial systems obtained by computing Lie derivatives of output functions in ODE models. This is, for example, a key component of assessing parameter identifiability of the ODE models [21, 22]. We observe significant improvements for multiple models that vary in complexity, number of polynomials, and number of variables.

Our main idea for weight generation lies in the observation that the "closedness" of parameters and states in the ODE to the outputs makes a difference for the effect of a weighted ordering. These empirical observations translated into a sequential Lie differentiation of output functions. Effectively, this differentiation produces Taylor coefficients of output functions $\mathbf{y}$ in terms of states at a fixed time $t = 0$ and parameters. We assign weights depending on the depth of these Taylor coefficients, thus, effectively, leveraging the outputs "sensitivity".

If the systems were already relatively quick to return the answer, the weights did not have a negative impact. In fact, in examples where computation slowed down (see e.g. Section 7.10), the memory usage still showed a positive effect, decreasing by around 80%. There was also a case where the program ran around 44% faster but consumed 30% more memory. These non-trivial examples constitute a minority of systems. In some cases, a user would not require a weighted ordering because the Gröbner basis computation runs fast without weights.

ACKNOWLEDGMENTS

The authors are grateful to CCiS at CUNY Queens College for the computational resources, to Andrew Brouwer for pointing out the HPV example, to Mohab Safey El Din for bringing our attention to regular sequences, to Alexander Demin for helpful discussions about F4 algorithm, and to the referees for their useful comments. This work was partially supported by the NSF grants CCF-2212460, 1563942, 1564132 and DMS-1760448, 1853650, 1853482, and the French ANR-22-CE48-0008 OCCAM project.

REFERENCES

[1] M. A. Al-Radhawi, M. Sadeghi, and E. Sontag. 2021. Long-term regulation of prolonged epidemic outbreaks in large populations via adaptive control: a singular perturbation approach. *IEEE Control System Letters* (2021).

[2] E. Balsa-Canto, A. A Alonso, and J. R. Banga. 2010. An iterative identification procedure for dynamic modeling of biochemical networks. *BMC Systems Biology* 4, 1 (2010), 1–18.

[3] E. Balsa-Canto, D. Henriques, A. Gábor, and J. Banga. 2016. AMIGO2, a toolbox for dynamic modeling, optimization and control in systems biology. *Bioinformatics* 32, 21 (2016), 3357–3359.

[4] T. Becker and V. Weispfenning. 1993. *Gröbner bases.* Springer New York, NY.

[5] J. Berthomieu, C. Eder, and M. Safey El Din. 2021. msolve: A Library for Solving Polynomial Systems. In *2021 International Symposium on Symbolic and Algebraic Computation.* Saint Petersburg, Russia, 51–58.

[6] A. F. Brouwer, R. Meza, and M. C. Eisenberg. 2015. Transmission heterogeneity and autoinoculation in a multisite infection model of HPV. *Mathematical Biosciences* 270 (2015), 115–125.

[7] B. Buchberger. 1976. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.* 10, 3 (1976), 19–29.

[8] M. A. Capistrán, M. A. Moreles, and B. Lara. 2009. Parameter estimation of some epidemic models. The case of recurrent epidemics caused by respiratory syncytial virus. *Bulletin of Mathematical Biology* 71, 8 (2009), 1890–1901.

[9] E. Dankwa, C. Donnelly, A. Brouwer, R. Zhao, M. Montgomery, M. Weng, and N. Martin. 2021. Estimating vaccination threshold and impact in the 2017–2019 hepatitis A virus outbreak among persons experiencing homelessness or who use drugs in Louisville, Kentucky, United States. *Vaccine* 39, 49 (2021), 7182–7190.

[10] S. Demignot and D. Domurado. 1987. Effect of prosthetic sugar groups on the pharmacokinetics of glucose-oxidase. *Drug design and delivery* 1, 4 (1987), 333–348.

[11] R. Dong, C. Goodbrake, H. A. Harrington, and G. Pogudin. 2023. Differential elimination for dynamical models via projections with applications to structural identifiability. *SIAM Journal on Applied Algebra and Geometry* 7, 1 (2023).

[12] C. Eder and T. Hofmann. 2021. Efficient Gröbner bases computation over principal ideal rings. *Journal of Symbolic Computation* 103 (2021), 1–13.

[13] J. Egea, D. Henriques, T. Cokelaer, A. Villaverde, A. MacNamara, D. Danciu, J. Banga, and J. Saez-Rodriguez. 2014. MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC Bioinformatics* 15, 1 (2014), 136.

[14] M. England. 2020. Real quantifier elimination by cylindrical algebraic decomposition, and improvements by machine learning. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation (ISSAC '20).* ACM.

[15] J.-C. Faugère. 1999. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra* 139, 1 (1999), 61–88.

[16] J.-C. Faugère. 2002. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation.* 75–83.

[17] J.-C. Faugère, M. Safey El Din, and T. Verron. 2013. On the complexity of computing Gröbner bases for quasi-homogeneous systems. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation.* 189–196.

[18] J.-C. Faugère, M. Safey El Din, and T. Verron. 2016. On the complexity of computing Gröbner bases for weighted homogeneous systems. *Journal of Symbolic Computation* 76 (2016), 107–141.

[19] D. Florescu and M. England. 2020. A Machine Learning Based Software Pipeline to Pick the Variable Ordering for Algorithms with Polynomial Inputs. In *Mathematical Software – ICMS 2020 (Lecture Notes in Computer Science, Vol. 12097).* Springer, 302–311.

[20] B. C. Goodwin. 1965. Oscillatory behavior in enzymatic control processes. *Advances in enzyme regulation* 3 (1965), 425–437.

[21] H. Hong, A. Ovchinnikov, G. Pogudin, and C. Yap. 2019. SIAN: software for structural identifiability analysis of ODE models. *Bioinformatics* 35, 16 (2019), 2873–2874.

[22] H. Hong, A. Ovchinnikov, G. Pogudin, and C. Yap. 2020. Global identifiability of differential models. *Communications on Pure and Applied Mathematics* 73, 9 (2020), 1831–1879.

[23] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. 2006. COPASI—a COmplex PAthway SImulator. *Bioinformatics* 22, 24 (2006), 3067–3074.

[24] I. Ilmer, A. Ovchinnikov, and G. Pogudin. 2021. Web-based Structural Identifiability Analyzer. In *Computational Methods in Systems Biology (Lecture Notes in Computer Science, Vol. 12881).* 254–265.

[25] M. Kauers and J. Moosbauer. 2020. Good Pivots for Small Sparse Matrices. In *Computer Algebra in Scientific Computing. CASC 2020 (Lecture Notes in Computer Science, Vol. 12291).* Springer, 358–367.

[26] Z. Li, M. Osborne, and T. Prvan. 2005. Parameter estimation of ordinary differential equations. *IMA J. Numer. Anal.* 25, 2 (2005), 264–285.

[27] T. Lipniacki, P. Paszek, A. Brasier, B. Luxon, and M. Kimmel. 2004. Mathematical model of NF-kB regulatory module. *Journal of Theoretical Biology* 228, 2 (2004), 195–215.

[28] M. Locke, G. Lythe, M. López-García, C. Muñoz-Fontela, M. Carroll, and C. Molina-París. 2021. Quantification of Type I Interferon Inhibition by Viral Proteins: Ebola Virus as a Case Study. *Viruses* 13, 12 (2021), 2441.

[29] G. Massonis, J. R. Banga, and A. F. Villaverde. 2021. Structural Identifiability and Observability of Compartmental Models of the COVID-19 Pandemic. *Annual Reviews in Control* 51 (2021), 441–459.

[30] F. Ollivier. 2023. *Effective formal resolution of systems of algebraic differential equations.* Habilitation à diriger des recherches. Institut Polytechnique de Paris. https://hal.science/tel-04098759

[31] A. Ovchinnikov. 2021. SIAN-Julia: Structural Identifiability Analyzer. https://github.com/alexeyovchinnikov/SIAN-Julia

[32] A. Ovchinnikov, A. Pillay, G. Pogudin, and T. Scanlon. 2021. Computing all identifiable functions of parameters for ODE models. *Systems & Control Letters* 157 (2021), 105030.

[33] A. Ovchinnikov, G. Pogudin, and P. Thompson. 2023. Input-output equations and identifiability of linear ODE models. *IEEE Trans. Automat. Control* 68 (2023), 812–824.

[34] D. Peifer, M. Stillman, and D. Halpern-Leistner. 2020. Learning selection strategies in Buchberger's algorithm. In *International Conference on Machine Learning.* PMLR, 7575–7585.

[35] L. Pickering, T. del Río Almajano, M. England, and K. Cohen. 2024. Explainable AI Insights for Symbolic Computation: A case study on selecting the variable ordering for cylindrical algebraic decomposition. *Journal of Symbolic Computation*

123 (2024), 102276.
[36] A. Raue, B. Steiert, M. Schelker, C. Kreutz, T. Maiwald, H. Hass, J. Vanlier, C. Tönsing, L. Adlung, R. Engesser, W. Mader, T. Heinemann, J. Hasenauer, M. Schilling, T. Höfer, E. Klipp, F. Theis, U. Klingmüller, B. Schöberl, and J. Timmer. 2015. Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. *Bioinformatics* 31, 21 (2015), 3558–3560.
[37] X. Rey Barreiro and A. Villaverde. 2023. Benchmarking tools for a priori identifiability analysis. *Bioinformatics* 39, 2 (2023).
[38] H. Schmidt and M. Jirstrand. 2005. Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics* 22, 4 (2005), 514–515.
[39] A. Sedoglavic. 2002. A probabilistic algorithm to test local algebraic observability in polynomial time. *Journal of Symbolic Computation* 33, 5 (2002), 735–755.
[40] A. Tran, M. A. Al-Radhawi, I. Kareva, J. Wu, D. Waxman, and E. Sontag. 2020. Delicate Balances in Cancer Chemotherapy: Modeling Immune Recruitment and Emergence of Systemic Drug Resistance. *Frontiers in Immunology* (2020).
[41] T. Verron. 2016. *Régularisation du calcul de bases de Gröbner pour des systèmes avec poids et déterminantiels, et application en imagerie médicale.* Ph. D. Dissertation. Paris 6.
[42] S. Zhang, J. Ponce, Z. Zhang, G. Lin, and G. Karniadakis. 2021. An integrated framework for building trustworthy data-driven epidemiological models: Application to the COVID-19 outbreak in New York City. *PLoS Computational Biology* 17, 9 (2021).
[43] A. I. Zobnin. 2009. One-element differential standard bases with respect to inverse lexicographical orderings. *Journal of Mathematical Sciences* 163, 5 (2009), 523–533.

7 SYSTEMS AND WEIGHTS

In this section, we present details about models considered. Specifically, we will describe the differential equations and the resulting weights of the models used in the analysis of this paper.

7.1 Goodwin oscillator

This model is presented in (19) and comes from [20] and describes time periodicity in cell behavior. This example has 4 state variables $x_{1,2,3,4}$ and 6 parameters. Below is the Goodwin oscillator model and the weight assignment for it (identity weights are not displayed):

$$\begin{cases} \dot{x}_1 = -b\,x_1 + \frac{1}{(c+x_4)}, & \\ \dot{x}_2 = \alpha\,x_1 - \beta\,x_2, & x_2 \implies 3 \\ \dot{x}_3 = \gamma\,x_2 - \delta\,x_3, & x_3 \implies 3 \\ \dot{x}_4 = \frac{\sigma\,x_4\,(\gamma\,x_2 - \delta\,x_3)}{x_3}, & x_4 \implies 2 \\ y = x_1 & \beta \implies 4 \end{cases} \tag{19}$$

SIAN uses an auxiliary variable z_{aux} to account for the presence of denominators in the right-hand side of the original input ODE system. We observe that giving a weight of at most 3 to this variable does not decrease performance.

7.2 SEIRP model

This is a biomedical model applied to COVID-19 in [29]. The outputs were changed to make the system more of a computational challenge to SIAN. Below is the SEIRP model and the weight assignment for it (identity weights are not displayed):

$$\begin{cases} \dot{S} = -\alpha_e\,S\,E - \alpha_i\,S\,I, & \\ \dot{E} = \alpha_e\,S\,E + \alpha_i\,S\,I - \kappa\,E - \rho\,E, & \\ \dot{I} = \kappa\,E - \beta\,I - \mu\,I, & E \implies 2 \\ \dot{R} = \beta\,I + \rho\,E, & \rho \implies 3 \\ \dot{P} = \mu\,I, & \\ y_1 = I + S & \end{cases} \tag{20}$$

7.3 SEIR COVID-19 model

Next we consider a SEIR-model of epidemics from [29, table 2, ID=14]. The example originally had 3 output functions. We reduced it to 1 to create more of a computational challenge for our program. We also use the term $\mu\,i\,s$ instead of $\mu\,i\mu\,s$ in the third equation. The state-space form of the model and the weight assignment are presented in (21):

$$\begin{cases} \dot{S} = \mu\,N - \alpha\,S - \beta\,S\,I\,N - \mu\,S, & \\ \dot{E} = \beta\,S\,I\,N - \mu\,E - \gamma\,E, & \\ \dot{I} = \gamma\,E - \delta\,I - \mu\,I\,S, & S \implies 2 \\ \dot{Q} = \delta\,I - \lambda\,Q - \kappa\,Q - \mu\,Q, & I \implies 3 \\ \dot{R} = \lambda\,Q - \mu\,S, & E \implies 4 \\ \dot{D} = \kappa\,Q, & \gamma \implies 4 \\ \dot{C} = \alpha\,S - \mu\,C - \tau\,C, & \delta \implies 4 \\ y = C & \end{cases} \tag{21}$$

7.4 SIR model with forcing term

The following model was presented in [8]. This is a SIR-model with an oscillating forcing term given by equations for $x_1,\ x_2$. We also give our weight assignment.

$$\begin{cases} \dot{S} = \mu - \mu\,S - b_0\,(1 + b_1\,x_1)\,I\,S + g\,R, & \\ \dot{I} = b_0\,(1 + b_1\,x_1)\,I\,S - (\nu + \mu)\,I, & \\ \dot{R} = \nu\,I - (\mu + g)\,R, & x_1 \implies 2 \\ \dot{x}_1 = -M\,x_2, & x_2 \implies 3 \\ \dot{x}_2 = M\,x_1, & M \implies 3 \\ y_1 = I,\ y_2 = R. & \end{cases} \tag{22}$$

7.5 A different SEIR-like COVID-19 model

The following model also comes from [29]. We also provide our weight assignment.

$$\begin{cases} \dot{S}_d = -\epsilon_s\,\beta_a\,(A_n + \epsilon_a\,A_d)\,S_d - h_1\,S_d + h_2\,S_n - \epsilon_s\,\beta_i\,S_d\,I_n, \\ \dot{S}_n = -\beta_i\,S_n\,I_n - \beta_a\,(A_n + \epsilon_a\,A_d)\,S_n + h_1\,S_d - h_2\,S_n, \\ \dot{A}_d = \epsilon_s\beta_i S_d I_n + \epsilon_s\beta_a(A_n + \epsilon_a A_d)S_n + h_2 A_n - \gamma_{ai} A_d - h_1 A_d, \\ \dot{A}_n = \beta_i\,S_n\,I_n + \beta_a\,(A_n + \epsilon_a\,A_d)\,S_n + h_1\,A_d - \gamma_{ai}\,A_n - h_2\,A_n, \\ \dot{I}_n = f\,\gamma_{ai}\,(A_d + A_n) - \delta\,I_n - \gamma_{ir}\,I_n, \\ \dot{R} = (1 - f)\,\gamma_{ai}\,(A_d + A_n) + \gamma_{ir}\,I_n, \\ y_1 = S_d,\ \ y_2 = I_n \end{cases}$$

$$\begin{array}{ll} A_d & \implies 2 \\ A_n & \implies 2 \\ S_n & \implies 2 \\ \beta_{a,i} & \implies 2 \\ h_{1,2} & \implies 2 \\ \gamma_{ai} & \implies 2 \\ f & \implies 2 \\ \epsilon_{a,s} & \implies 2 \end{array}$$

In this model, the computation without weights has not finished in reasonable time, consuming all available memory.

7.6 Pharmacokinetics model

This model comes from [10] describing pharmacokinetics of glucose-oxidase. We make one modification setting $a_1 = a_2$. The model is small but presents a significant computational challenge

for global identifiability, that is, it is very difficult to compute Gröbner basis of this model's polynomial system in SIAN. We also provide our weight assignment:

$$\begin{cases} \dot{x}_1 = a_1\,(x_2 - x_1) - \frac{(k_a\,n\,x_1)}{(k_c\,k_a + k_c\,x_3 + k_a\,x_1)}, \\ \dot{x}_2 = a_1\,(x_1 - x_2), \\ \dot{x}_3 = b_1\,(x_4 - x_3) - \frac{(k_c\,n\,x_3)}{(k_c\,k_a + k_c\,x_3 + k_a\,x_1)}, \\ \dot{x}_4 = b_2\,(x_3 - x_4), \\ y_1 = x_1 \end{cases} \quad \begin{array}{l} x_2 \implies 2 \\ x_3 \implies 3 \\ x_4 \implies 3 \\ b_2 \implies 4 \end{array} \tag{23}$$

7.7 NF-κB model

This model comes from [27] and was used for identifiability analysis in [2]. The ODE system consists of 15 equations, (24),

$$\begin{cases} \dot{x}_1 = k_p - k_d\,x_1 - k_1\,x_1\,u, \\ \dot{x}_2 = -k_3\,x_2 - k_d\,x_2 - a_2\,x_2\,x_{10} + \\ +t_1\,x_4 - a_3\,x_2\,x_{13} + t_2\,x_5 + (k_1\,x_1 - k_2\,x_2\,x_8)\,u, \\ \dot{x}_3 = k_3\,x_2 - k_d\,x_3 + k_2\,x_2\,x_8\,u, \\ \dot{x}_4 = a_2\,x_2\,x_{10} - t_1\,x_4, \\ \dot{x}_5 = a_3\,x_2\,x_{13} - t_2\,x_5, \\ \dot{x}_6 = c_{6a}\,x_{13} - a_1\,x_6\,x_{10} + t_2\,x_5 - i_1\,x_6, \\ \dot{x}_7 = i_1\,k_v\,x_6 - a_1\,x_{11}\,x_7, \\ \dot{x}_8 = c_4\,x_9 - c_5\,x_8, \\ \dot{x}_9 = c_2 + c_1\,x_7 - c_3\,x_9, \\ \dot{x}_{10} = -a_2\,x_2\,x_{10} - a_1\,x_{10}\,x_6 + c_{4a}x_{12} - \\ -c_{5a}x_{10} - i_{1a}x_{10} + e_{1a}x_{11}, \\ \dot{x}_{11} = -a_1\,x_{11}\,x_7 + i_{1a}\,k_v\,x_{10} - e_{1a}\,k_v\,x_{11}, \\ \dot{x}_{12} = c_{2a} + c_{1a}\,x_7 - c_{3a}\,x_{12}, \\ \dot{x}_{13} = a_1\,x_{10}\,x_6 - c_{6a}\,x_{13} - a_3\,x_2\,x_{13} + e_{2a}\,x_{14}, \\ \dot{x}_{14} = a_1\,x_{11}\,x_7 - e_{2a}\,k_v\,x_{14}, \\ \dot{x}_{15} = c_{2c} + c_{1c}\,x_7 - c_{3c}\,x_{15} \end{cases} \tag{24}$$

and the outputs, (25):

$$\begin{cases} y_1 = x_2, \quad y_2 = x_{10} + x_{13}, \quad y_3 = x_9, \\ y_4 = x_1 + x_2 + x_3, \quad y_5 = x_7, \quad y_6 = x_{12}, \end{cases} \tag{25}$$

We use the values of these parameters from [27] to reduce the number of target identifiability candidates: $a_1, a_2, a_3, c_{1a}, c_{5a}, c_{1c}, c_{3c}, c_{2c}, c_1, c_2, c_3, c_4, e_{1a}, k_v$. The output functions of (24) yields these weights (not listed states get weight of 1): $c_5 \implies 3,\ x_4, x_5, x_6, x_8, x_{11}, x_{14} \implies 2$.

7.8 Two SEIR epidemiological models

The following two SEIR models were presented in [29, Examples 34 and 16]. Example 34 is presented in (26), while example 16 is given by (27). We also provide our weight assignments.

$$\begin{cases} \dot{S} = \Lambda - r\,\beta\,S\,I/N - \mu\,S, \\ \dot{E} = \beta\,S\,I/N - \epsilon\,E - \mu\,e, \\ \dot{I} = \epsilon\,E - \gamma\,I - \mu\,I, \\ \dot{R} = \gamma\,I - \mu\,R, \\ y = I + R. \end{cases} \quad \begin{array}{l} E \implies 2 \\ S \implies 3 \\ \gamma \implies 4 \end{array} \tag{26}$$

$$\begin{cases} \dot{S} = -\beta\,S\,I, \\ \dot{E} = \beta\,S\,I - \epsilon\,E, \\ \dot{I} = \epsilon\,E - (\rho + \mu)\,I, \\ \dot{R} = \rho\,I - d\,R, \\ y = I + R \end{cases} \quad \begin{array}{l} E \implies 2 \\ S \implies 3 \\ \beta \implies 3 \\ \rho \implies 3 \\ \gamma \implies 4 \end{array} \tag{27}$$

The output functions for both examples are structurally similar. They are different from those in the original paper to increase the computational difficulty for SIAN's Gröbner basis routine.

7.9 HPV models

We considered two HPV models studied in [6]. The model itself is given by (28) with indices $i, j \in \{F, M\}$. We present the Gröbner basis computation timings for two cases of outputs given in equations (30) and (29). The outputs in (30) result in the weight of 2 assigned to the following parameters and states provided in (32) where $i, j \in \{M, F\}$. Everything else gets weight 1. The output collection from (29) results in (31).

$$\begin{cases} \dot{S}_i &= \frac{\mu}{2} + \gamma_i^G\,I_i^G + \gamma_i^O\,I_i^O - S_i\,\mu - S_i\,(\beta_{ji}^{OO}(I_i^O + I_i^{OG}) \\ &\quad +\beta_{ji}^{GO}(I_i^G + I_i^{OG})) - S_i(\beta_{ji}^{OG}(I_i^O + I_i^{OG}) + \\ &\quad +\beta_{ji}^{GG}(I_i^G + I_i^{OG})), \\ \dot{I}_i^O &= S_i\,(\beta_{ji}^{OO}\,(I_i^O + I_i^{OG}) + \beta_{ji}^{GO}\,(I_i^G + I_i^{OG})) + \gamma_i^G\,I_i^{OG} \\ &\quad -I_i^O\,(\nu_M^{OG} + \gamma_i^O + \mu + \beta_{ji}^{OG}\,(I_i^O + I_i^{OG}) + \\ &\quad +\beta_{ji}^{GG}\,(I_i^G + I_i^{OG})), \\ \dot{I}_i^G &= S_i\,(\beta_{ji}^{OG}\,(I_i^O + I_i^{OG}) + \beta_{ji}^{GG}\,(I_i^G + I_i^{OG})) + \gamma_i^O\,I_i^{OG} \\ &\quad -I_i^G\,(\nu_M^{GO} + \gamma_i^G + \mu + \beta_{ji}^{OO}\,(I_i^O + I_i^{OG}) + \\ &\quad +\beta_{ji}^{GO}\,(I_i^G + I_i^{OG})), \\ \dot{I}_i^{OG} &= I_i^O\,(\nu_M^{OG} + \beta_{ji}^{OG}\,(I_i^O + I_i^{OG}) + \beta_{ji}^{GG}\,(I_i^G + I_i^{OG})) \\ &\quad +I_i^G\,(\nu_M^{GO} + \beta_{ji}^{OO}\,(I_i^O + I_i^{OG}) + \beta_{ji}^{GO}\,(I_i^G + I_i^{OG})) \\ &\quad -I_i^{OG}\,(\gamma_i^O + \gamma_i^G + \mu), \end{cases} \tag{28}$$

$$\text{Output set 1: } \begin{cases} y_1 = I_M^G + I_M^{OG}, \quad y_2 = I_M^O + I_M^{OG}, \\ y_3 = I_M^{OG} + I_F^{OG} \end{cases} \tag{29}$$

$$\text{Output set 2: } \begin{cases} y_1 = I_M^G + I_M^{OG}, \quad y_2 = I_M^O + I_M^{OG}, \\ y_3 = I_F^G + I_F^{OG}, \quad y_4 = I_F^O + I_F^{OG} \end{cases} \tag{30}$$

$$\text{Output set 1 weights: } \begin{cases} I_F^G, I_F^O, I_F^{OG}, S_M \Rightarrow 2 \\ S_F, \gamma_F^G, \gamma_F^O, \nu_F^{GO}, \nu_F^{OG}, \\ \beta_{FM}^{GG}, \beta_{MF}^{GO}, \beta_{MF}^{OG}, \beta_{MF}^{OO} \Rightarrow 3 \end{cases} \tag{31}$$

$$\text{Output set 2 weights: } \begin{cases} S_i, \gamma_i^G, \gamma_i^O, \mu, \nu_i^{GO}, \nu_i^{OG}, \beta_{ij}^{GG}, \beta_{ji}^{GG}, \beta_{ij}^{GO}, \\ \beta_{ji}^{GO}, \beta_{ij}^{OG}, \beta_{ji}^{OG}, \beta_{ij}^{OO}, \beta_{ji}^{OO} \end{cases} \tag{32}$$

7.10 Example with slowdown: a SIR-model

In (33) from [29, Table 1, ID 26], we present an example in which the weight assignment generated by our algorithm *increases* the

running time of F4. In Maple, we observed an increase in CPU time from around 12 to 50 minutes while memory usage slightly decreases from 11.5 to 10.8 GB. In Magma, this system shows a larger increase in memory from 5.6 to 18.8 GB with an increase in CPU time from around 7 to 32 minutes.

$$\begin{cases} \dot{S} = b\,N - S\,(I\,\lambda + \lambda\,Q\,\epsilon_a\,\epsilon_q + \lambda\,\epsilon_a\,A + \lambda\,\epsilon_j\,J + d + 1), \\ \dot{I} = k_1\,A - (g_1 + \mu_2 + d_2)\,I, \\ \dot{R} = g_1\,In + g_2\,J - d_3\,R, \\ \dot{A} = S\,(I\,\lambda + \lambda\,Q\,\epsilon_a\,\epsilon_q + \lambda\,\epsilon_a\,A + \lambda\,\epsilon_j\,J) - (k_1 + \mu_1 + d_4)\,A, \\ \dot{Q} = \mu_1\,A - (k_2 + d_5)\,Q, \\ \dot{J} = k_2\,Q + \mu_2\,I - (g_2 + d_6)\,J, \\ y_1 = Q, \\ y_2 = J \end{cases} \tag{33}$$

Model information			Time (min)			Memory (GB)			Primes	
Model name	num. polys.	num. vars.	old SIAN ordering	differential degrevlex	our final ordering	speedup	old SIAN ordering	differential degrevlex	our final ordering	reduction
HPV, (28), (30)	79	75	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
COVID Model 2, (7.5)	49	48	N/A	3762.7	∞	N/A	23.5	∞	N/A	84
Pharmacokinetics, (23)	48	47	N/A	102.5	∞	N/A	8.0	∞	N/A	212
HPV, (28), (29)	97	92	N/A	179.0	∞	N/A	4.0	∞	N/A	82
Goodwin Oscillator (19)	42	43	148.6	7.8	19.0	11.0	0.8	13.7	103	103
SEIR-1, (26)	44	45	10.2	0.8	12.6	3.3	0.1	24.7	68	68
COVID Model 1, (21)	51	50	1346.9	5.0	173.9	15.7	0.3	45.7	120	120
NF-κB, (24)	120	109	40.6	13.7	3.0	6.3	3.2	1.9	88	89
SEIRP, (20)	50	42	8.3	3.5	2.4	1.7	0.3	6.3	32	32
SEIR-2, (27)	44	43	2.9	1.7	1.8	0.8	0.2	4.6	38	38

Table 4: Weighted ordering applied to the Gröbner basis computation step of SIAN with zero characteristic using Maple 2021.2. "N/A" stands for the Maple error "`Error, (in Groebner:-F4:-GroebnerBasis) numeric exception: division by zero`" without a clear direct cause.

Model information			Time (min)			Memory (GB)			Primes	
Model name	num. polys.	num. vars.	old SIAN ordering	differential degrevlex	our final ordering	speedup	old SIAN ordering	differential degrevlex	our final ordering	reduction
HPV, (28), (30)	79	75	31.2	85.2	0.4	2.8	2.7	1.0	8	74
COVID Model 2, (7.5)	49	48	20722.2	6784.6	3.1	38.6	24.1	1.6	36	93
Pharmacokinetics, (23)	48	47	1181.6	202.7	5.8	9.4	10.8	0.9	17	275
HPV, (28), (29)	97	92	1229.2	457.4	2.7	10.6	18.8	0.6	167	185
Goodwin Oscillator (19)	42	43	112.5	3.5	31.8	3.1	0.5	6.0	109	169
SEIR-1, (26)	44	45	18.5	0.3	73.2	2.1	0.2	9.9	84	87
COVID Model 1, (21)	51	50	7626.9	3.1	2468.9	8.8	1.8	4.9	237	260
NF-κB, (24)	120	109	78.7	12.5	6.3	2.8	0.7	4.8	9	107
SEIRP, (20)	50	42	41.5	3.3	12.6	1.3	1.6	0.8	62	41
SEIR-2, (27)	44	43	7.6	1.1	6.9	1.4	0.7	1.7	80	78

Table 5: Weighted ordering applied to the Gröbner basis computation step of SIAN with zero characteristic using Magma V2.26-8.

8 INVERTED WEIGHTS

The weight assignment we discussed above is not unique. In fact, we can even find an alternative assignment given the same weight generation procedure as described earlier. Instead of simply using

Model information			Time (min)			Memory (GB)		
Model name	num. polys.	num. vars.	eq. (18) order	eq. (18), inv. weights	speedup	eq. (18) order	eq. (18), inv. weights	reduction
HPV, (28), (30)	79	75	N/A	N/A	-	N/A	N/A	-
COVID Model 2, (7.5)	49	48	N/A	607.1	∞	N/A	33.6	∞
Pharmacokinetics, (23)	48	47	N/A	127.0	∞	N/A	6.02	∞
HPV, (28), (29)	97	92	N/A	19.1	∞	N/A	3.4	∞
Goodwin Oscillator (19)	42	43	29.8	0.6	72.6	10.6	0.1	21.7
SEIR-1, (26)	44	45	2.2	0.23	14.9	3.3	0.1	16.2
COVID Model 1, (21)	51	50	321.9	148.2	2.2	15.2	3.4	4.4
NF-κB, (24)	120	109	7.1	5.3	1.3	6.1	1.9	3.2
SEIRP, (20)	50	42	2.0	4.5	0.5	1.6	0.8	2.1
SEIR-2, (27)	44	43	0.8	0.7	1.2	0.7	0.1	6.7

Table 6: Results of applying the *inverted* weighted ordering to only Gröbner basis computation step of SIAN with characteristic $p = 11863279$ using Maple 2021.2. "N/A" stands for the Maple error "`Error, (in Groebner:-F4:-GroebnerBasis) numeric exception: division by zero`" without a clear direct cause.

the rule of higher level, as in (7), we can generate the following assignment:

$$\text{Weight(x)} := M - \text{Level}(x) + 1,$$

where $M = \max_x \text{Level}(x)$ is a maximal possible level of each state. With this strategy, we can also see improvement. In fact, for certain systems, such as (19), this assignment is more beneficial in reducing the runtime. At the same time, in case of (28), (30), we observe a similar error message in Maple as if weights were not present. The results of this new assignment are presented in Table 6. It shows that there is still room for improvement in finding a weight assignment rule.

One would expect that many of the same phenomena from the previous weight assignment would happen with these models as well. In particular, it should attempt to force the variables of *low* level to appear in the leading monomials of the corresponding equations, thus making the pairs involving these equations to be of higher formal degree, which would, once again, avoid selecting too many pairs at once and steer the computation towards first working with the variables and equations at a *high* level only. This heuristic is similar to how one would solve a "simple" ODE system by integration. In particular, if one would try to solve the equations

$$x_2' = a \tag{34}$$

$$x_1' = x_2 \tag{35}$$

$$y = x_1, \tag{36}$$

one would first solve for x_2 by integrating the constant a in (34) to get $x_2 = at + x_2(0)$, then would integrate x_1 in (35) to get $x_1 = at^2 + x_2(0)t + x_1(0)$, and finally after substituting for y in (36) we get $y(t) = at^2 + x_2(0)t + x_1(0)$. The inverted weight assignment heuristic models the way a human would naturally solve such an ODE system. While we expect the regular weight assignment to be useful more often, we also expect there to be ODE systems that have a nice structure in the equations coming from higher derivatives that the inverted weights select first before other equations.