OBTAINING WEIGHTS FOR GRÖBNER BASIS COMPUTATION IN PARAMETER IDENTIFIABILITY PROBLEMS

A PREPRINT

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

Ilia Ilmer  
Ph.D. Program in Computer Science,  
CUNY Graduate Center,  
New York, USA  
iilmer@gradcenter.cuny.edu

Tatiana Konstantinova  
Department of Mathematics,  
CUNY Queens College,  
New York, USA  
tkonstant@bnl.gov

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

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

Pedro Soto  
Ph.D. Program in Computer Science,  
CUNY Graduate Center,  
New York, USA  
psoto@gradcenter.cuny.edu

February 15, 2022

ABSTRACT

We consider a specific class of polynomial systems that arise in parameter identifiability problems of models of ordinary differential equations (ODE) and discover a method for speeding up the Gröbner basis computation by using a weighted ordering. Our method explores the structure of the ODE model to generate weight assignments for each variable. We provide empirical results that show improvement across different symbolic computing frameworks.

Keywords: F4 Algorithm, Weighted Monomial Ordering, Parameter Identifiability, ODE Systems, Mathematical Biology

1 Introduction

Structural parameter identifiability is a property crucial for designing high-quality mathematical models of real-world phenomena using ordinary differential equations. The question of identifiability arises when one seeks a value for a particular parameter of the model. There can be at most finitely many such values (local structural identifiability) or the value is unique (global structural identifiability). For each case, there exist algorithms that provide solutions with high degree of correctness. We address computational challenges of computing Gröbner bases [4] for polynomial ideals that arise in parameter identifiability. This approach, for instance, lies at the core of global identifiability tool SIAN [19]. The computation itself is well-known to be heavy for some polynomial systems, despite such algorithms as F4 [11] and F5 [12]. We developed an approach that uses additional domain-specific information about the polynomial ideal to significantly speed up the process.
Obtaining weights for Gröbner basis computation in parameter identifiability problems

The Gröbner basis (itself and its computation) of a polynomial system can vary based on the order of monomials. The most common and empirically reliable in terms of computing time ordering of monomials is the so-called total-degree-reverse-lexicographic order, or \( \text{td} \text{deg} \) in MAPLE notation. Weighted ordering adds a layer of comparison to monomial orderings where we first compare variables by the weight value multiplied by its degree exponent and then ties are broken according to, for example, \( \text{td} \text{deg} \). Properly chosen weights will have tremendous impact on the computation time. Consider the following motivating example showing benefits of weights in general:

\[
P := \left\{ \begin{array}{l}
 x_1^7 x_1^4 + x_1 x_2 x_3^2 x_2^5 + x_1 x_2 x_3 x_4 x_5 x_7 + x_1 x_2 x_3 x_4 x_6 x_8 + \\
 + x_1 x_2 x_4^2 x_6^2 + x_2 x_4^6, 
 x_1^6, 
 x_2^6
\end{array} \right.
\]

(1)

Computing the Gröbner basis of this system with \( \text{td} \text{deg} \)-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 minutes of total elapsed time as computed in Maple 2021 on MacBook Pro with 16 GB of RAM and 16-core M1 processor. Modifying the system by substituting \( x_8 \rightarrow x_2^8 \) results in approximately 2 seconds of CPU time and only about 1 second of total elapsed time.

The main result of this paper is the rule for assigning weights in parameter identifiability problems based on the data from the input ODE models. That is, given a polynomial system obtained by SIAN from an input ODE model, the rule produces a weight assignment to the variables based on available data from the original ODE that improves the Gröbner basis computation in the vast majority of cases. We provide experimental results showing that improvements in runtime and memory consumption for MAPLE and Magma. We will take advantage of this improvement to accelerate parameter identifiability solving. The implementation of this strategy will be included in an upcoming release of SIAN and SIAN-Julia [28].

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 contains the weight generation algorithm. In Section 4, we show the experimental results and benchmarks with our new weight assignment approach. Finally, in Section 5, we provide concluding remarks of our work.

2 Related Work

2.1 Gröbner basis and weighted ordering

Gröbner basis is the key part of SIAN [19]. Buchberger [4] presented the first algorithm to solve the challenge of finding the basis of polynomial systems. This solution can be resource heavy [27] and depends on many factors, such as selection strategy for polynomials [17]. The improvement of the algorithm itself came from the works of Faugère, whose F4 [11] and later F5 [12] are a tremendous success. The key idea of algorithms such as F4 is to take advantage of mathematical tools from the realm of linear algebra. The work [2] addresses the termination and complexity properties of the F5 algorithm. In general, these algorithms are much more efficient than the default Buchberger and each creates a family of solutions to the Gröbner basis problem. For an excellent taxonomy of F5-based approaches, we refer to [9].

The analysis of connection between weights and homogenization of ideals appeared in [13] and later in more detail in [14]. Homogeneous ideals are an intriguing special case of inputs for a Gröbner basis algorithm. In the mentioned works, weights were used as a homogenization tool. We have observed in the motivating example (1) above that a weighted ordering can break homogenization, offering large benefits. In fact, polynomial systems in SIAN contain non-homogeneous polynomials in most cases by the nature of the input data. For example, an output function (see Definition 1) of the form

\[ y_i = g_i(\ldots), \]

where \( g_i \) is a polynomial, is inevitably transformed into an equation with a free term of degree 0. Therefore, polynomial systems of SIAN always have a non-homogeneous polynomial. By design of the algorithm, the weight of any variable in \( g_i \) will be 1. 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. In this sense, we do not necessarily make polynomials “more homogeneous”.

Machine learning offers other ways of improving computer algebra algorithms (e.g., an algorithm for Gröbner basis computation). In particular, [30] shows how reinforcement learning improves some aspects of Gröbner basis computation, such as the selection strategy. Machine learning has found applications in other areas of computer algebra, such as choosing order for cylindrical algebraic decomposition [16, 21, 15, 16] or choosing pivots in Gaussian elimination [23].
2.2 Parameter identifiability

The problem of parameter identifiability has been of great interest in the modeling community, and with recent computational advances it gained a lot of attention in the broader scientific computing community. SciML organization recently included an implementation of the global identifiability tool from [8]. The identifiability software ŠIAN [19] that we consider in this paper has been recently used, for instance, in [6, 32, 33, 31]. Together with the algorithm from [29], it has been turned into a web-based analyzer package [22] which was applied in, for instance, [25].

3 Main results

In this section, we present the method for obtaining weighted ordering that improves the runtime of the Gröbner basis computation for polynomial systems formed out of ordinary differential equations. Let us define a model in a state-space form which represents the input to the identifiability software.

**Definition 1 (Model in the state-space form).** A model in the state-space form is a system

\[
\Sigma := \begin{cases} 
\dot{x} &= f(x, \mu, u), \\
y &= g(x, \mu, u), \\
x(0) &= x^*, 
\end{cases}
\]

(2)

where \( f = (f_1, \ldots, f_n) \) and \( g = (g_1, \ldots, g_m) \) with \( f_i = f_i(x, \mu, u) \), \( g_i = g_i(x, \mu, u) \) are rational functions over the complex numbers \( \mathbb{C} \). The vector \( x = (x_1, \ldots, x_n) \) represents the time-dependent state variables and \( \dot{x} \) represents the derivative.

The vectors \( u = (u_1, \ldots, u_s) \), \( y = (y_1, \ldots, y_m) \), \( \mu = (\mu_1, \ldots, \mu_\lambda) \), and \( x^* = (x_1^*, \ldots, x_n^*) \) represent the input variables, output variables, parameters, and initial conditions, respectively.

SIAN [19] transforms the input system from the form of Equation (2) to a collection of differential polynomials before assessing global identifiability. To this end, SIAN computes truncated Taylor polynomials of the output functions at time \( t = 0 \). For more details about the truncation bound, we refer to [20, Theorems 3.16, 4.12]. The final global identifiability assessment relies on: sampling \( x^*, \mu \) in the polynomial system, calculating \( y_i \) and its derivatives based on the sample, and using Gröbner basis computation to check uniqueness of \( x^*, \mu \) that provide \( y_i \) values. This procedure makes SIAN a Monte-Carlo algorithm with user specified probability of correctness. Our goal is to find a weight assignment to each variable of the resulting polynomial system.

3.1 Selecting states for weight assignment

3.1.1 The weight assignment rule.

Given a system (2), one can define the Lie derivative \( \mathcal{L}(h) \) of a function \( h \in \mathbb{C}(x, \mu, u, u', \ldots) \) with respect to the system by

\[
\mathcal{L}(h) = \sum_{i=1}^{n} f_i \frac{\partial h}{\partial x_i} + \sum_{j=1}^{s} u_j \frac{\partial h}{\partial u_j}.
\]

(3)

By applying this formula to each output function \( y_i \), we can define, for each state variable or a parameter \( a \in \{x, \mu\} \) a level as

\[
\text{Level}(a) := \min_i \left[ \exists y_j \in y: a \text{ appears in } \mathcal{L}^i(y_j) \right].
\]

(4)

Using that value, we assign weight as follows:

- for a state variable \( x_i \in x \) (and all its derivatives)
  \[
  \text{Weight}(x_i) := \text{Level}(x_i) + 1;
  \]

- for a parameter \( \mu_i \in \mu \):
  \[
  \text{Weight}(\mu_i) := \begin{cases} 
  \text{Level}(\mu) + 1, & \text{if Level}(\mu_i) = \max_{e \in \mu \cup \lambda} \text{Level}(e), \\
  1, & \text{otherwise}.
  \end{cases}
  \]
Remark 1 (How we found it?). Empirically, we observe that lower occurrence frequency of a state variable in the right-hand side of the ODE system (excluding output functions) leads to a more successful weight assignment. By success here we mean the assignment’s ability to reduce the Gröbner basis algorithm runtime. In our original brute-force search for weights we found that if a state variable is present in the right-hand side of the output functions $y$ then assigning it a weight $w > 1$ would decrease performance. Using these observations, we apply a Lie derivative operator $L$ to each output $y_i$. If the differentiation results in no new state variables, we skip the output function $y_i$ in the next iteration. We repeat this until all states (and parameters) are found. At the same time, we assign each variable a “level” value corresponding to a derivative order in which it appears. To assign weights, we follow the principle “deeper level means higher weight”.

Remark 2. In our search for weights, we aim at minimizing the difference in treatment of state variables and constant parameters of the ODE input. Indeed, any constant parameter can be considered a state with zero derivative. The only exception for constant parameters is that we assign weight to those from the maximal possible level. We only consider locally identifiable constant parameters.

3.1.2 Example
Consider the following ODE system

$$\Sigma = \begin{cases} 
\dot{x}_1 = Ax_1 + Bx_2, \\
\dot{x}_2 = Cx_1, \\
y_1 = x_1.
\end{cases}$$

Differentiating once:

$$\dot{y}_1 = \dot{x}_1 = Ax_1 + Bx_2.$$ 

We see that $A, B, x_2$ all occur after the first differentiation and hence will have weight of 2. 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

$$\ddot{y}_1 = A\dot{x}_1 + B\dot{x}_2 = Ax_1 + Bx_2 + Cx_2,$$

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

$$A, B, x_1 \Rightarrow 1, \ x_2 \Rightarrow 2, \ C \Rightarrow 3.$$ 

4 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.

Normally, MAPLE’s implementation of the F4 algorithm picks multiple different large prime numbers to be the characteristic of the underlying field. In our experiments, we specify a single positive characteristic $p = 11863279$ for better highlights of any time and memory consumption difference between weighted and non-weighted cases. 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 method, 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

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

(6)
Obtaining weights for Gröbner basis computation in parameter identifiability problems

Figure 1: Runtime comparison when running MAPLE’s Gröbner basis with weights (red) and without weights (default, blue) on top of order eq. (8).

SIAN produces 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.

$$E = \begin{cases} 
7828371 - x_{1:0} - x_{2:0} - ax_1 + x_{1:1}, \\
bx_1 - cx_2 + x_{2:1} - x_{1:1} - x_{2:1} + 22382588610034, \\
-x_{1:1} + x_{1:1}, bx_{1:1} - cx_{2:1} + x_{2:2}, \\
x_{1:1} - x_{2:2} + 98741152216384012566, \\
-x_{1:1} + x_{1:1}, bx_{1:1} - cx_{2:2} + x_{2:3}, \\
x_{1:3} + x_{2:3} + 538005180363000517510923144, \\
-x_{1:3} + x_{1:4}, bx_{1:3} - cx_{2:3} + x_{2:4}, \\
x_{1:4} - x_{2:4} + 3127015821351630984063850338736 
\end{cases}$$

(7)

the order of variables that will provide the best speed without weights is

$x_{2:4}, x_{1:4}, x_{2:3}, x_{1:3}, x_{2:2}, x_{1:2}, x_{2:1}, x_{1:1}, x_{2:0}, x_{1:0}, a, b, c.$

(8)

That is, we order variables from higher to lower derivative degree grouping the same degree together (all order-4 derivatives, then all order-3, etc.)

In what follows, we apply the weights on top of the default variable ordering eq. (8) 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 1 and 2:

1. Number of polynomials in the polynomial system.
2. Number of variables in the polynomial system.
3. Default (without weights) CPU time in minutes.
4. Default (without weights) memory in GB.
5. CPU time in minutes with weights.
6. Memory in GB with weights.
7. Speedup calculated as $\frac{\text{old time}}{\text{new time}}$.
8. 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.
5 Concluding remarks

We presented an algorithm for generating weighted orderings of variables that we used in Gröbner basis computation for parameter identifiability problems. SIAN [19, 20] generates these polynomial ideals by successive differentiation of output functions. We observed 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 occurrence of parameters and states in the ODE makes a difference for the effect of a weighted ordering. These empirical observations translated into a sequential Lie differentiation of output functions, mimicking a Breadth-First Search algorithm in graph theory. Effectively, this differentiation produces Taylor coefficients of output functions $y$ 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”.

In cases where 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, the memory usage still showed a positive effect, decreasing by around 80%. There has also been a case where the program ran around 44% faster but consumed 30%
Obtaining weights for Gröbner basis computation in parameter identifiability problems

COVID Model 1 eq. (A.2)

COVID Model 2 eq. (A.3)

Pharmacokinetics eq. (A.15)

HPV eqs. (A.8) and (A.9)

HPV eqs. (A.8) and (A.10)

Figure 4: Memory comparison for very heavy systems (some do not terminate without weights) when running MAPLE’s Gröbner basis with weights (red) and without weights (default, blue) on top of order eq. (8).

Table 1: Results of applying the weighted ordering to only Gröbner basis computation step of SIAN with positive characteristic \( p = 11863279 \) using MAPLE 2021.2. “N/A” stands for the following error message returned by MAPLE: “Error, (in Groebner:-F4:-GroebnerBasis) numeric exception: division by zero”

| Model information | Time (min) | Memory (GB) |
|-------------------|------------|-------------|
| Model name        | num. polys. | num. vars. | alphabetical order | eq. (8) order | eq. (8) weighted | speedup eq. (8) | alphabetical order | eq. (8) order | eq. (8) weighted | reduction eq. (8) |
| COVID Model 2, eq. (A.3) | 49 | 48 | N/A | N/A | 602.0 | \( \infty \) | N/A | N/A | 23.2 | \( \infty \) |
| Pharmacokinetics, eq. (A.15) | 48 | 47 | N/A | N/A | 21.0 | \( \infty \) | N/A | N/A | 7.7 | \( \infty \) |
| HPV, eqs. (A.8) and (A.9) | 97 | 92 | N/A | N/A | 13.9 | \( \infty \) | N/A | N/A | 3.7 | \( \infty \) |
| HPV, eqs. (A.8) and (A.10) | 79 | 75 | N/A | N/A | 5.1 | \( \infty \) | N/A | N/A | 11.0 | \( \infty \) |
| COVID Model 1, eq. (A.2) | 51 | 50 | 377.0 | 321.9 | 1.0 | 327.6 | 15.3 | 15.2 | 0.3 | 52.6 |
| Goodwin Oscillator, eq. (A.1) | 42 | 43 | 44.1 | 29.8 | 1.5 | 18.9 | 10.8 | 10.6 | 0.7 | 14.6 |
| SEIR-1, eq. (A.5) | 44 | 45 | 3.5 | 2.2 | 0.1 | 17.4 | 3.3 | 3.3 | 0.1 | 44.8 |
| NF-κB, eq. (A.13) | 120 | 109 | 10.6 | 7.1 | 2.3 | 3.0 | 11.8 | 6.1 | 3.1 | 1.9 |
| SEIR-2, eq. (A.4) | 50 | 42 | 2.6 | 2.0 | 0.8 | 2.5 | 1.0 | 1.6 | 0.2 | 8.5 |
| SEIR-2, eq. (A.6) | 44 | 43 | 1.3 | 0.8 | 0.4 | 2.2 | 0.8 | 0.7 | 0.1 | 6.1 |

more memory. These non-trivial examples constitute a minority of systems. In some cases, a user would not require a weighted ordering because of how fast the computation of Gröbner basis runs without weights.

It is necessary to note that the weight assignment provided here 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 the rule of higher level Equation (4), we can generate the following assignment:

\[
\text{Weight}(x) := M - \text{Level}(x) + 1, \tag{9}
\]

where \( M = \max_x \text{Level}(x) \) is a maximal possible level of each state. With this strategy, we are also able to see improvement, in fact, for certain systems, such as Equation (A.1), this assignment is more beneficial in reducing the runtime. At the same time, in case of Equations (A.8) and (A.10), we observe a similar error message in MAPLE as if weights were not present. The results of this new assignment are presented in Table 3.
Obtaining weights for Gröbner basis computation in parameter identifiability problems

| Model name          | num. polys. | num. vars. | Time (min) | Memory (GB) | Speedup | Reduction |
|---------------------|-------------|------------|------------|-------------|---------|-----------|
| COVID Model 2, eq. (A.3) | 49          | 48         | 4000.6     | 3471.2      | 517.4   | 6.7       |
| Pharmacokinetics, eq. (A.15) | 48          | 47         | 757.6      | 248.3       | 44.5    | 5.6       |
| HPV, eqs. (A.8) and (A.9) | 97          | 92         | 321.7      | 126.6       | 51.5    | 2.4       |
| HPV, eqs. (A.8) and (A.10) | 79          | 75         | 6.8        | 5.9         | 3.2     | 1.4       |
| COVID Model 1, eq. (A.2) | 51          | 50         | 1331.1     | 1272.1      | 0.6     | 2207.9    |
| Goodwin Oscillator eq. (A.1) | 42          | 43         | 26.9       | 22.4        | 0.8     | 28.5      |
| SEIR-1, eq. (A.5) | 44          | 45         | 8.6        | 3.9         | 0.1     | 76.0      |
| NF-κB, eq. (A.13) | 120         | 109        | 14.6       | 9.1         | 1.7     | 5.2       |
| SEIRP, eq. (A.4) | 50          | 42         | 10.0       | 6.8         | 36.5    | 11.2      |
| SEIR-2, eq. (A.6) | 44          | 43         | 3.4        | 1.2         | 0.2     | 7.6       |

Table 2: Results of applying the weighted ordering to only Gröbner basis computation step of SIAN with positive characteristic $p = 11863279$ in Magma 2.26-8.

Machine learning can offer an automated approach for determining weights. Given a feature set of both the ODE model and the polynomial system, a supervised learning algorithm can output a weight vector or a classification of “Slow” and “Fast” with respect to Gröbner basis. The difficulty of this method lies in determining meaningful features and their representation and, for weights output, knowing the target weights in advance.

Reinforcement learning, in which an algorithm is trained via an interactive reward system, is another possible option. This was explored in [30] for selection strategies in Buchberger’s algorithm. For our case, to find best weights, a reinforcement learning agent would sample weight values, compute the basis, aggregate reward if the computation is improved or lose the reward otherwise. So far in our experiments, this approach has not yielded an improving weight combination that would outperform default order. Another concern is poor generalization properties of this method where one would have to retrain an agent for each new input system. This could be addressed by expanding the state space to include the input ODE systems, so that the agent could make a step by not only choosing weights but also adjusting inputs.

Finally, we discuss the possible effect weights have on the F4 algorithm. The F4 algorithm operates by creating a matrix with rows representing monomials up to a certain degree [11]. From the log outputs of MAPLE’s Gröbner basis function, we observed that without weights, the algorithm spends a considerable amount of time on a degree value while with weights there is either no such delay or it is postponed to a much later stage (higher degree). Assume that computation slows down at degree $n$. After we introduce a weight of $w$, whatever monomial caused the delay will no longer be present, since degrees have changed (recall that we apply weight by raising variables to the power $w$). The problematic computation then moves to higher degrees.

Acknowledgements

The authors are grateful to CCiS at CUNY Queens College for the computational resources and to Andrew Brouwer for pointing out the HPV example. This work was partially supported by the National Science Foundation under grants CCF-1563942, CCF-1564132, DMS-1760448, DMS-1853650, and DMS-1853482.

References

[1] E. Balsa-Canto, A. A Alonso, and J. R. Banga. “An iterative identification procedure for dynamic modeling of biochemical networks”. In: BMC Systems Biology 4.1 (2010), pp. 1–18.

[2] M. Bardet, J.-C. Faugère, and B. Salvy. “On the complexity of the F5 Gröbner basis algorithm”. In: Journal of Symbolic Computation 70 (2015), pp. 49–70.
Obtaining weights for Gröbner basis computation in parameter identifiability problems

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

[4] B. Buchberger. “A theoretical basis for the reduction of polynomials to canonical forms”. In: SIGSAM Bull. 10.3 (1976), pp. 19–29.

[5] M. A. Capistrán, M. A. Moreles, and B. Lara. “Parameter estimation of some epidemic models. The case of recurrent epidemics caused by respiratory syncytial virus”. In: Bulletin of Mathematical Biology 71.8 (2009), pp. 1890–1901.

[6] E. Dankwa, C. Donnelly, A. Brouwer, R. Zhao, M. Montgomery, M. Weng, and N. Martin. “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”. In: Vaccine 39.49 (2021), pp. 7182–7190.

[7] S. Demignot and D. Domurado. “Effect of prosthetic sugar groups on the pharmacokinetics of glucose-oxidase”. In: Drug design and delivery 1.4 (1987), pp. 333–348.

[8] R. Dong, C. Goodbrake, H. A. Harrington, and G. Pogudin. Differential elimination for dynamical models via projections with applications to structural identifiability. 2022.

[9] C. Eder and J.-C. Faugère. “A survey on signature-based algorithms for computing Gröbner bases”. In: Journal of Symbolic Computation 80 (2017), pp. 719–784.

[10] M. England and D. Florescu. “Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition”. In: Intelligent Computer Mathematics. CICM 2019. Vol. 11617. Lecture Notes in Computer Science. 2019, pp. 93–108.

[11] J.-C. Faugère. “A new efficient algorithm for computing Gröbner bases (F4)”. In: Journal of Pure and Applied Algebra 139.1 (1999), pp. 61–88.

[12] J.-C. Faugère. “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. 2002, pp. 75–83.

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

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

[15] D. Florescu and M. England. “Algorithmically generating new algebraic features of polynomial systems for machine learning”. In: (2019).

[16] D. Florescu and M. England. “Improved cross-validation for classifiers that make algorithmic choices to minimise runtime without compromising output correctness”. In: (2019).

[17] A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso. “One sugar cube, please” or selection strategies in the Buchberger algorithm”. In: Proceedings of the 1991 international symposium on Symbolic and algebraic computation. 1991, pp. 49–54.

[18] B. C. Goodwin. “Oscillatory behavior in enzymatic control processes”. In: Advances in enzyme regulation 3 (1965), pp. 425–437.

[19] H. Hong, A. Ovchinnikov, G. Pogudin, and C. Yap. “SIAN: software for structural identifiability analysis of ODE models”. In: Bioinformatics 35.16 (2019), pp. 2873–2874.

[20] H. Hong, A. Ovchinnikov, G. Pogudin, and C. Yap. “Global identifiability of differential models”. In: Communications on Pure and Applied Mathematics 73.9 (2020), pp. 1831–1879.

[21] Z. Huang, M. England, D. Wilson, J. Davenport, L. Paulson, and J. Bridge. “Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition”. In: Intelligent Computer Mathematics. CICM 2014. Vol. 8543. Lecture Notes in Computer Science. 2014, pp. 92–107.

[22] I. Ilmer, A. Ovchinnikov, and G. Pogudin. “Web-based Structural Identifiability Analyzer”. In: Computational Methods in Systems Biology. Vol. 12881. Lecture Notes in Computer Science. 2021, pp. 254–265.

[23] M. Kauers and J. Moosbauer. “Good pivots for small sparse matrices”. In: Computer Algebra in Scientific Computing. CASC 2020. Vol. 12291. Lecture Notes in Computer Science. Springer. 2020, pp. 358–367.

[24] T. Lipniacki, P. Paszek, A. Brasier, B. Luxon, and M. Kimmel. “Mathematical model of NF-kB regulatory module”. In: Journal of Theoretical Biology 228.2 (2004), pp. 195–215.

[25] M. Locke, G. Lythe, M. López-García, C. Muñoz-Fontela, M. Carroll, and C. Molina-París. “Quantification of Type I Interferon Inhibition by Viral Proteins: Ebola Virus as a Case Study”. In: Viruses 13.12 (2021), p. 2441.

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

A.1 Goodwin oscillator

The first model we consider is provided in Equation (A.1). The system comes from \cite{18} and describes time periodicity in cell behavior. This example has 4 state variables $x_1, x_2, x_3, x_4$ and 6 parameters.

\[
\begin{align*}
\dot{x}_1 &= -b x_1 + \frac{1}{(c+x_4)}, \\
\dot{x}_2 &= \alpha x_1 - \beta x_2, \\
\dot{x}_3 &= \gamma x_2 - \delta x_3, \\
\dot{x}_4 &= \sigma x_4 (\gamma x_2 - \delta x_3) \\
y &= x_1 
\end{align*}
\]

(A.1)

The differentiation algorithm applied to this model returns the following weight distribution

$x_1 \Rightarrow 1$, $x_4 \Rightarrow 2$, $x_2, x_3 \Rightarrow 3$, $\beta \Rightarrow 4$.

This indicates that, e.g., $x_2$ and all of its derivatives receive a weight of 3. On the other hand, only constant $\beta$ receives weight of 4.

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.

A.2 SEIR COVID-19 model

In this next example, we considered a SEIR-model of epidemics from \cite[26, table 2, ID=14]{26}. 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 $\mu s$ instead of $\mu i s$ in the third equation. The state-space form of the model is presented in Equation (A.2).

\[
\begin{align*}
\dot{S} &= \mu N - \alpha S - \beta S I N - \mu S, \\
\dot{E} &= \beta S I N - \mu E - \gamma E, \\
\dot{I} &= \gamma E - \delta I - \mu I S, \\
\dot{Q} &= \delta I - \lambda Q - \kappa Q - \mu Q, \\
\dot{R} &= \lambda Q - \mu S, \\
\dot{D} &= \kappa Q, \\
\dot{C} &= \alpha S - \mu C - \tau C, \\
y &= C 
\end{align*}
\]

(A.2)

As a result of our weight ordering pick, we assign these weights:

$C \Rightarrow 1$, $\gamma, \delta, E \Rightarrow 4$, $I \Rightarrow 3$, $S \Rightarrow 2$.
A.3 A different SEIR-like COVID-19 model

The following model also comes from [26]

\[
\begin{aligned}
\dot{S}_d &= -\epsilon_s \beta_a (A_n + \epsilon_a A_d) S_d - h_1 S_d + h_2 S_n - \epsilon_s \beta_s S_d I_n, \\
\dot{S}_n &= -\beta_s S_n I_n - \beta_a (A_n + \epsilon_a A_d) S_n + h_1 S_d - h_2 S_n, \\
\dot{A}_d &= \epsilon_s \beta_a (A_n + \epsilon_a A_d) S_n + h_2 A_n - \gamma ai A_d - h_1 A_d, \\
\dot{A}_n &= \beta_i S_n I_n + \beta_a (A_n + \epsilon_a A_d) S_n + h_1 A_d - \gamma ai A_n - h_2 A_n, \\
\dot{I}_n &= f \gamma ai (A_d + A_n) - \delta I_n - \gamma ir I_n, \\
\dot{R} &= (1 - f) \gamma ai (A_d + A_n) + \gamma ir I_n, \\
y_1 &= S_d, \\
y_2 &= I_n.
\end{aligned}
\] (A.3)

In this model, the computation without weights has not finished in reasonable time, consuming all available memory. Our algorithm returns the following weight assignment:

\[A_d, A_n, S_n, \beta_{a,i}, h_{1,2}, \gamma_{ai}, f, \epsilon_a, s \Rightarrow 2, \quad I_n, S_d \Rightarrow 1\]

A.4 SEIRP model

This is a biomedical model applied to COVID-19 present in [26]. The outputs were changed to make the system more of a computational challenge to SIAN.

\[
\begin{aligned}
\dot{S} &= -\alpha_e S E - \alpha_i S I, \\
\dot{E} &= \alpha_e S E + \alpha_i S I - \kappa E - \rho E, \\
\dot{I} &= \kappa E - \beta I - \mu I, \\
\dot{R} &= \beta I + \rho E, \\
\dot{P} &= \mu I, \\
y_1 &= I + S
\end{aligned}
\] (A.4)

The weights for this system were assigned as follows:

\[\rho \Rightarrow 3, \quad E \Rightarrow 2, \quad I, S \Rightarrow 1\]

A.5 Two SEIR epidemiological models

The next two SEIR models were presented in [26] in examples 34 and 16. Example 34 is presented in Equation (A.5)

\[
\begin{aligned}
\dot{S} &= \Lambda - r \beta S I / N - \mu S, \\
\dot{E} &= \beta S I / N - \epsilon E - \mu e, \\
\dot{I} &= \epsilon E - \gamma I - \mu I, \\
\dot{R} &= \gamma I - \mu I, \\
y &= I + R.
\end{aligned}
\] (A.5)

Example 16 is given by Equation (A.6)

\[
\begin{aligned}
\dot{S} &= -\beta S I, \\
\dot{E} &= \beta S I - \epsilon E, \\
\dot{I} &= \epsilon E - (\rho + \mu) I, \\
\dot{R} &= \rho I - d R, \\
y &= I + R
\end{aligned}
\] (A.6)

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. The weights assigned in the case of Equation (A.5) (left) and Equation (A.6) (right) are

\[
\begin{aligned}
E \Rightarrow 2, \quad \gamma \Rightarrow 4, \quad S \Rightarrow 3 & \quad E \Rightarrow 2, \quad \gamma \Rightarrow 4, \quad S, \rho, \beta \Rightarrow 3. \\
I, R \Rightarrow 1. & \quad I, R \Rightarrow 1.
\end{aligned}
\]
A.6 SIR model with forcing term

The following model was presented in [5]. This is a SIR-model with an oscillating forcing term given by equations for \( x_1, x_2 \).

\[
\begin{align*}
\dot{S} &= \mu - \mu S - b_0 (1 + b_1 x_1) IS + g R, \\
\dot{I} &= b_0 (1 + b_1 x_1) IS - (\nu + \mu) I, \\
\dot{R} &= \nu I - (\mu + g) R, \\
\dot{x}_1 &= -M x_2, \quad \dot{x}_2 = M x_1, \\
y_1 &= I, y_2 = R.
\end{align*}
\]
(A.7)

The weight assignment here is as follows:

\[M, x_2 \Rightarrow 3, \quad I, R \Rightarrow 1, \quad S, x_1 \Rightarrow 2,\]

A.7 HPV models

We considered two HPV models studied in [3]. The model itself is given by eq. (A.8) with indices \( i, j \in \{F, M\} \). We present the Gröbner basis computation timings for two cases of outputs given in equations (A.10) and (A.9) below. The outputs in eq. (A.10) result in the weight of 2 assigned to the following parameters and states provided in eq. (A.12) where \( i, j \in \{M, F\} \). Everything else gets weight 1. The output collection from eq. (A.9) results in eq. (A.11).

\[
\begin{align*}
\dot{S}_i &= \dot{G}_i + \gamma_i \dot{I}_i + \gamma_i \dot{O}_i - S_i \mu - S_i \beta_i (\dot{I}_i^O + \dot{I}_i^G) \\
&\quad + \beta_i (\dot{I}_i^G + \dot{I}_i^O), \\
\dot{I}_i^O &= S_i \beta_i (\dot{I}_i^O + \dot{I}_i^G) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) + \gamma_i \dot{I}_i^G \\
&\quad - \dot{I}_i^O (\nu M + \gamma_i + \mu) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) \\
&\quad + \beta_i (\dot{I}_i^G + \dot{I}_i^O), \\
\dot{I}_i^G &= S_i \beta_i (\dot{I}_i^G + \dot{I}_i^O) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) + \gamma_i \dot{I}_i^O \\
&\quad - \dot{I}_i^G (\nu M + \gamma_i + \mu) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) \\
&\quad + \beta_i (\dot{I}_i^G + \dot{I}_i^O), \\
\dot{I}_i^G &= \dot{I}_i^O (\nu M + \gamma_i + \mu) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) \\
&\quad + \dot{I}_i^G (\nu M + \gamma_i + \mu) + \beta_i (\dot{I}_i^G + \dot{I}_i^O) \\
&\quad - \dot{I}_i^G (\nu M + \gamma_i + \mu),
\end{align*}
\]
(A.8)

Output set 1:
\[
\begin{align*}
y_1 &= I^G_M + I^O_M, \\
y_2 &= I^O_M + I^G_M, \\
y_3 &= I^G_M + I^F_M.
\end{align*}
\]
(A.9)

Output set 2:
\[
\begin{align*}
y_1 &= I^G_M + I^M_M, \\
y_2 &= I^O_M + I^M_M, \\
y_3 &= I^F_M + I^G_M.
\end{align*}
\]
(A.10)

Output set 1 weights:
\[
\begin{align*}
I^F_M, I^O_M, I^G_M, S_M &\Rightarrow 2, \\
S_F, \gamma_i, \nu_i, \beta_i, \nu_i^G, \nu_i^G, \beta_i^G &\Rightarrow 2, \\
\beta_i^G, \nu_i^G, \nu_i^G, \beta_i^G &\Rightarrow 3.
\end{align*}
\]
(A.11)

Output set 2 weights:
\[
\begin{align*}
S_i^G, \gamma_i^G, \nu_i^G, \nu_i^G, \beta_i^G, \nu_i^G, \beta_i^G, \beta_i^G, \beta_i^G &\Rightarrow 2, \\
\beta_i^G, \nu_i^G, \nu_i^G, \beta_i^G, \beta_i^G, \beta_i^G, \beta_i^G &\Rightarrow 3.
\end{align*}
\]
(A.12)
A.8 NF-$\kappa$B model

This model comes from [24] and was used for identifiability analysis in [1]. The ODE system consists of 15 equations, eq. (A.13), and the outputs, eq. (A.14).

\[
\begin{align*}
\dot{x}_1 &= k_p - k_d x_1 - k_1 x_1 u, \quad \dot{x}_2 = -k_3 x_2 - k_d x_2 - a_2 x_2 x_{10} + t_1 x_4 - a_3 x_2 x_{13} + t_2 x_5 + (k_1 x_1 - k_2 x_2 x_8) u, \\
\dot{x}_3 &= k_3 x_2 - k_d x_3 + k_2 x_2 x_8 u, \\
\dot{x}_4 &= a_2 x_2 x_{10} - t_1 x_4, \quad \dot{x}_5 = a_3 x_2 x_{13} - t_2 x_5, \\
\dot{x}_6 &= c_{6a} x_{13} - a_1 x_6 x_{10} + t_2 x_5 - i_1 x_6, \\
\dot{x}_7 &= i_1 k_v x_6 - a_1 x_1 x_7, \\
\dot{x}_8 &= c_4 x_9 - c_5 x_8, \quad \dot{x}_9 = c_2 + c_1 x_7 - c_3 x_9, \\
\dot{x}_{10} &= -a_2 x_2 x_{10} - a_1 x_10 x_6 + c_{4a} x_{12} - c_{5a} x_{10} - i_{1a} x_{10} + c_{1a} x_{11}, \\
\dot{x}_{11} &= a_1 x_11 x_7 + i_{1a} k_v x_{10} - c_{1a} k_v x_{11}, \\
\dot{x}_{12} &= c_{2a} + c_{1a} x_7 - c_{3a} x_{12}, \\
\dot{x}_{13} &= a_1 x_{10} x_6 - c_{6a} x_{13} - a_3 x_2 x_{13} + c_{2a} x_{14}, \\
\dot{x}_{14} &= a_1 x_{11} x_7 - c_{2a} k_v x_{14}. \quad \dot{x}_{15} = c_{2c} + c_{1c} x_7 - c_{3c} x_{15}
\end{align*}
\]

\[
\begin{align*}
y_1 &= x_2, \\
y_2 &= x_{10} + x_{13}, \\
y_3 &= x_9, \\
y_4 &= x_1 + x_2 + x_3, \\
y_5 &= x_7, \\
y_6 &= x_{12},
\end{align*}
\]

We specify values of the following parameters based on the known information from [24] to reduce the number of target identifiability candidates

\[a_1, a_2, a_3, c_{1a}, c_{5a}, c_{1c}, c_{3c}, c_{2c}, c_1, c_2, c_3, c_4, c_{1a}, k_v,\]

The output functions of eq. (A.13) yields the weights as below (states not listed below get weight of 1)

\[c_5 \Rightarrow 3, \quad x_4, x_5, x_6, x_8, x_{11}, x_{14} \Rightarrow 2\]

A.9 Pharmacokinetics model

This model comes from [7] 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.

\[
\begin{align*}
\dot{x}_1 &= a_1 (x_2 - x_1) - \frac{(k_u x_1)}{(k_u + k_v x_3 + k_u x_1)}, \\
\dot{x}_2 &= a_1 (x_1 - x_2), \\
\dot{x}_3 &= b_1 (x_4 - x_3) - \frac{(k_u x_3)}{(k_u + k_v x_3 + k_u x_1)}, \\
\dot{x}_4 &= b_2 (x_3 - x_4), \\
y_1 &= x_1
\end{align*}
\]

The weight assignment for this system is

\[b_2 \Rightarrow 4, x_1 \Rightarrow 1, x_2 \Rightarrow 2, \quad x_3, x_4 \Rightarrow 3\]

A.10 Example with slowdown: a SIR-model

In eq. (A.16), we present an example in which the weight assignment generated by our algorithm increases the running time of F4. This example comes from [26, Table 1, ID 26]. While running it in MAPLE, we observed an increase in CPU time from around 12 to 50 minutes. The 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.

\[
\begin{align*}
\dot{S} &= b N - S (I \lambda + \lambda Q \epsilon_a \epsilon_q + \lambda \epsilon_j A + \lambda \epsilon_j J + d + 1), \\
\dot{I} &= k_1 A - (g_1 + \mu_2 + d_2) I, \\
\dot{R} &= g_1 I n + g_2 J - d_3 R, \\
\dot{A} &= S (I \lambda + \lambda Q \epsilon_a \epsilon_q + \lambda \epsilon_a A + \lambda \epsilon_j J) - (k_1 + \mu_1 + d_4) A, \\
\dot{Q} &= \mu_1 A - (k_2 + d_5) Q, \\
\dot{J} &= k_2 Q + \mu_2 I - (g_2 + d_6) J, \\
y_1 &= Q, y_2 = J
\end{align*}
\]  

(A.16)

B Additional tables

B.1 Inverted weights

In Table 3, we provide MAPLE benchmarks for the weights given by Equation (9). We see that for some systems, such as eq. (A.1), there is a considerable speedup, even compared to the main weight assignment of this paper. The eqs. (A.8) and (A.10) model, however, does not finish. Overall, this illustrates that while this may be inferior to the weight assignment by Equation (4), there are multiple ways to create an efficient weight distribution. The difference in improvement factors is presented in Figures 5 and 6.

| Model name                  | num. polys. | num. vars. | eq. (8) order | inv. weights | speedup | eq. (8) order | eq. (8), inv. weights | reduction |
|-----------------------------|-------------|------------|---------------|--------------|---------|---------------|-----------------------|-----------|
| HPV, egs. (A.8) and (A.10) | 79          | 75         | N/A           | N/A          | -       | N/A           | N/A                   | -         |
| COVID Model 2, eq. (A.3)    | 49          | 48         | N/A           | 607.1        | \infty  | N/A           | 33.6                  | \infty    |
| Pharmacokinetics, eq. (A.15)| 48          | 47         | N/A           | 127.0        | \infty  | N/A           | 6.02                  | \infty    |
| HPV, egs. (A.8) and (A.9)   | 97          | 92         | N/A           | 19.1         | \infty  | N/A           | 3.4                   | \infty    |
| Goodwin Oscillator, eq. (A.1)| 42          | 43         | 29.8          | 0.6          | 72.6    | 10.6          | 0.1                   | 21.7      |
| SEIR-1, eq. (A.5)           | 44          | 45         | 2.2           | 0.23         | 14.9    | 3.3           | 0.1                   | 16.2      |
| COVID Model 1, eq. (A.2)    | 51          | 50         | 321.9         | 148.2        | 2.2     | 15.2          | 3.4                   | 4.4       |
| NF-κB, eq. (A.13)           | 120         | 109        | 7.1           | 5.3          | 1.3     | 6.1           | 1.9                   | 3.2       |
| SEIRP, eq. (A.4)            | 50          | 42         | 2.0           | 4.5          | 0.5     | 1.6           | 0.8                   | 2.1       |
| SEIR-2, eq. (A.6)           | 44          | 43         | 0.8           | 0.7          | 1.2     | 0.7           | 0.1                   | 6.7       |

Table 3: Results of applying the inverted weighted ordering to only Gröbner basis computation step of SIAN with positive characteristic $p = 11863279$ using MAPLE 2021.2. “N/A” stands for the following error message returned by MAPLE: “Error, (in Groebner:-F4:-GroebnerBasis) numeric exception: division by zero”
Obtaining weights for Gröbner basis computation in parameter identifiability problems

Figure 5: Time improvement difference between inverted (eq. (9)) and regular weight assignments when running MAPLE’s Gröbner basis.

Figure 6: Memory improvement difference between inverted (eq. (9)) and regular weight assignments when running MAPLE’s Gröbner basis.