A new algorithm for finding the input-output equation of differential models

Ruiwen Dong
joint work with:
Gleb Pogudin, Heather Harrington, Christian Goodbrake

October 2020

Introduction : identifiability

Identifiability: property of a differential model with parameters that allows for the parameters to be determined uniquely from the model equations, noiseless data and sufficiently exciting inputs.

Classical example: The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = -cx_2 + dx_1x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{1}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are unknown parameters to be identified. We can observe the output $y = x_1$.

Identifiability: property of a differential model with parameters that allows for the parameters to be determined uniquely from the model equations, noiseless data and sufficiently exciting inputs.

Classical example: The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = -cx_2 + dx_1x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{1}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are unknown parameters to be identified. We can observe the output $y = x_1$.

Known result:
a, c, d are identifiable, but b is not.

Introduction : identifiability

Identifiability: property of a differential model with parameters that allows for the parameters to be determined uniquely from the model equations, noiseless data and sufficiently exciting inputs.

Classical example: The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = -cx_2 + dx_1x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{1}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are unknown parameters to be identified. We can observe the output $y = x_1$.

Known result:
a, c, d are identifiable, but b is not.

Importance of assessing identifiability: evaluate or reparametrize models before experiments.

Identifiability computation: example

Consider the following ODE system:

$$\Sigma \colon \begin{cases} \dot{x}_1 = a x_2 \\ \dot{x}_2 = b x_1 \\ y = x_1 \end{cases} \qquad (2)$$

where a, b are parameters to be determined, y is the output.
How to find out which parameters are identifiable?

Identifiability computation: example

Consider the following ODE system:

$$\Sigma\colon \begin{cases} \dot{x}_1 = ax_2 \\ \dot{x}_2 = bx_1 \\ y = x_1 \end{cases} \tag{2}$$

where a, b are parameters to be determined, y is the output.
How to find out which parameters are identifiable?

Immediate consequence of Σ :

$$\ddot{y} - aby = 0$$

called the **input-output equation**.

Identifiability computation: example

Consider the following ODE system:

$$\Sigma\colon \begin{cases} \dot{x}_1 = ax_2 \\ \dot{x}_2 = bx_1 \\ y = x_1 \end{cases} \tag{2}$$

where a, b are parameters to be determined, y is the output.
How to find out which parameters are identifiable?

Immediate consequence of Σ :

$$\ddot{y} - aby = 0$$

called the **input-output equation**.

Result: ab is identifiable from knowing y, but not a or b.

Identifiability computation: example

Consider the following ODE system:

$$\Sigma\colon \begin{cases} \dot{x}_1 = a x_2 \\ \dot{x}_2 = b x_1 \\ y = x_1 \end{cases} \tag{2}$$

where a, b are parameters to be determined, y is the output.
How to find out which parameters are identifiable?

Immediate consequence of Σ :

$$\ddot{y} - aby = 0$$

called the **input-output equation**.

Result: ab is identifiable from knowing y, but not a or b.

Input-output equations: "minimal" equations that depend only on the input and output variables and parameters.

Introduction : input-output equations

Two different kinds of identifiability:

- Single-experiment identifiability: what we can identify from a single experiment.
- Multi-experiment identifiability: what we can identify from a sufficiently (finite) many experiments.

Under some assumptions, **single-experiment identifiability = multi-experiment identifiability**
Input-output equations $\longrightarrow$ **Multi-experiment Identifiability**:

Introduction : input-output equations

Two different kinds of identifiability:

- Single-experiment identifiability: what we can identify from a single experiment.
- Multi-experiment identifiability: what we can identify from a sufficiently (finite) many experiments.

Under some assumptions, **single-experiment identifiability = multi-experiment identifiability**
Input-output equations $\longrightarrow$ **Multi-experiment Identifiability**:

Proposition

Note y the output of an ODE system, u its inputs, and $\boldsymbol{\theta}$ the vector of all its parameters.
Consider input-output equations as monic polynomials in y, u and their derivatives over the field $\mathbb{C}(\boldsymbol{\theta})$. A rational function of parameters $p \in \mathbb{C}(\boldsymbol{\theta})$ is multi-experiment identifiable if and only if it is in the field generated by the coefficients of the input-output equations.

Example: $\ddot{y} - aby = 0 \Longrightarrow \mathbb{C}(ab)$ is everything we can identify.

Immediate example

The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = a x_1 - b x_1 x_2 \\ \dot{x}_2 = -c x_2 + d x_1 x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{3}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are parameters we want to identify. We can observe the output $y = x_1$.

Immediate example

The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = -cx_2 + dx_1x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{3}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are parameters we want to identify. We can observe the output $y = x_1$.

Input-output equation:

$$ady^3 - acy^2 - dy^2\dot{y} + cy\dot{y} - \dot{y}^2 + y\ddot{y} = 0$$

Immediate example

The predator-prey model

$$\Sigma: \quad \begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = -cx_2 + dx_1x_2 \\ \text{output:} \quad y = x_1 \end{cases} \tag{3}$$

where x_1 is the number of prey, x_2 is the number of predator. a, b, c, d are parameters we want to identify. We can observe the output $y = x_1$.

Input-output equation:

$$ady^3 - acy^2 - dy^2\dot{y} + cy\dot{y} - \dot{y}^2 + y\ddot{y} = 0$$

Identifiability consequence:

$$\begin{aligned} &\text{The field of identifiable functions} \\ =&\text{The field that the coefficients of input-output equation generate} \\ =&\mathbb{C}(ac, ad, c, d) = \mathbb{C}(a, c, d) \end{aligned}$$

Problem

Setup:
Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ y = g(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{4}$$

where:

- $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of state variables;
- $\mathbf{u} = (u_1, \ldots, u_m)$ is a vector of input (control) variables to be chosen by an experimenter;
- y is the output variable (<u>scalar</u>: we limit ourselves to single output case);
- $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ is a vector of unknown (constant) parameters to be identified;
- $\mathbf{f} = (f_1, \ldots, f_n)$, where $f_i \in \mathbb{C}(\mathbf{x}, u, \boldsymbol{\theta})$ are rational functions;
- $g \in \mathbb{C}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta})$ is a rational function.

Problem

Setup:
Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ y = g(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{5}$$

Goal: Find a "minimal" consequence of Σ depending only on input, output, and parameters, also known as an *input-output equation*:

$$\phi(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \mathbf{u}^{(2)}, \ldots, y, y', y^{(2)} \ldots, y^{(h)}) = 0,$$

This means:

1. ϕ vanishes on every solution of the system Σ.
2. ϕ is an irreducible polynomial.
3. h is as small as possible.

Problem

Setup:
Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ y = g(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{5}$$

Goal: Find a "minimal" consequence of Σ depending only on input, output, and parameters, also known as an *input-output equation*:

$$\phi(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \mathbf{u}^{(2)}, \ldots, y, y', y^{(2)} \ldots, y^{(h)}) = 0,$$

This means:

1. ϕ vanishes on every solution of the system Σ.
2. ϕ is an irreducible polynomial.
3. h is as small as possible.

Proposition

The input-output equation exists and is unique.

State of art

Various existing software for checking *identifiability* (all of them support multiple outputs):

1. SIAN: Implemented on MAPLE, checks identifiability without computing the input-output equations.
2. DAISY: Written in REDUCE, checks identifiability by computing the input-output equations.
3. RosenfeldGroebner implemented in MAPLE: do differential elimination on the system of differential equations.
4. COMBOS: web-based application, checks multi-experiment identifiability by calculating the input-output equations.

Performance

SIWR model:

$$\begin{cases} \dot{s} = \mu - \beta_i si - \beta_w sw - \mu s + \alpha r, \\ \dot{i} = \beta_w sw + \beta_i si - (\gamma + \mu) i, \\ \dot{w} = \xi(i - w), \\ \dot{r} = \gamma i - (\mu + \alpha) r, \\ y = \kappa i \end{cases}$$

Pharmacokinetics model:

$$\begin{cases} \dot{x}_0 = a_1(x_1 - x_0) - \frac{k_a n x_0}{k_c k_a + k_c x_2 + k_a x_0}, \\ \dot{x}_1 = a_2(x_0 - x_1), \\ \dot{x}_2 = b_1(x_3 - x_2) - \frac{k_c n x_2}{k_c k_a + k_c x_2 + k_a x_0}, \\ \dot{x}_3 = b_2(x_2 - x_3), \\ y = x_0 \end{cases}$$

Hyperchaotic QWWC system:

$$\begin{cases} \dot{x} = a(y - x) + yz, \\ \dot{y} = b(x + y) - xz, \\ \dot{z} = -cz - dw + xy, \\ \dot{w} = ez - fw + xy, \\ o = x \end{cases}$$

Extended SEIR model:

$$\begin{cases} \dot{s} = -\beta s(i + j + qa), \\ \dot{e} = \beta s(i + j + qa) - ke, \\ \dot{a} = k(1 - \rho)e - \gamma_1 a, \\ \dot{i} = k\rho e - (\alpha + \gamma_1) i, \\ \dot{j} = \alpha i - \gamma_2 j, \\ \dot{c} = \alpha i, \\ y = c \end{cases}$$

Performance

Model	DAISY	RG	SIAN	**Our implementation** **
SIWR	> 5 h.	> 5 h.	> 5 h.	9 s. + 9 s. = 18 s.
Extended SEIR	OOM*	OOM*	> 5 h.	22 s. + 37 s. = 69 s.
Pharmacokinetics	> 5 h.	OOM*	> 5 h.	20 s. + 45 s. = 65 s.
QWWC	> 5 h.	OOM*	> 5 h.	236 s. + 246 s. = 482 s.

Table: Performance comparison

* OOM = out of memory

** Time for computing IO-equation + Time for identifiability

Model	io-equation size (N. terms)	identifiable
SIWR	209349	all
Extended SEIR	927131	$\beta, k, \gamma_1, \gamma_2, \alpha$
Pharmacokinetics	1062553	all
QWWC	6853210	a, b

Table: Results

First attempt: implicitization of Lie derivatives

Find Lie derivatives:

Original system:

$$\Sigma\colon \begin{cases} \dot{x}_1 = ax_1 + x_2^2 \\ \dot{x}_2 = bx_1^2 + x_2 \\ y = x_1 \end{cases}$$

$$\Longrightarrow$$

Lie derivatives:

$$\Pi\colon \begin{cases} y = x_1 \\ y' = \dot{x}_1 = ax_1 + x_2^2 \\ y'' = \ldots = a^2 x_1 + (a+2)x_2^2 + 2bx_2x_1^2 \end{cases}$$

First attempt: implicitization of Lie derivatives

Find Lie derivatives:

Original system:

$$\Sigma\colon \begin{cases} \dot{x}_1 = ax_1 + x_2^2 \\ \dot{x}_2 = bx_1^2 + x_2 \\ y = x_1 \end{cases} \implies$$

Lie derivatives:

$$\Pi\colon \begin{cases} y = x_1 \\ y' = \dot{x}_1 = ax_1 + x_2^2 \\ y'' = \ldots = a^2 x_1 + (a+2)x_2^2 + 2bx_2x_1^2 \end{cases}$$

Implicitization of hypersurface on $\mathbb{C}(a,b)^3$:

Parametric description:

$$\Pi\colon \begin{cases} y = g(x_1, x_2) \\ y' = g_1(x_1, x_2) \\ y'' = g_2(x_1, x_2) \end{cases} \implies$$

Implicit description:

$$\phi(y, y', y'') = 0$$

Methods: Groebner Basis, <u>Repeated resultants</u>, Macaulay resultant, Sparse resultant, Interpolation, ...

First attempt: efficiency problems

$$\Sigma \colon \begin{cases} \dot{x}_1 = a x_1 + x_2^2 \\ \dot{x}_2 = b x_1^2 + x_2 \\ y = x_1 \end{cases}$$

First attempt: efficiency problems

$$\Sigma \colon \begin{cases} \dot{x}_1 = ax_1 + x_2^2 \\ \dot{x}_2 = bx_1^2 + x_2 \\ y = x_1 \end{cases}$$

Calculate Lie derivatives recursively:

$$\Pi \colon \begin{cases} y = x_1 \\ y' = a x_1 + x_2^2 \\ y'' = a\dot{x}_1 + 2x_2\dot{x}_2 = a(ax_1 + x_2^2) + 2x_2(bx_1^2 + x_2) \end{cases}$$

First attempt: efficiency problems

$$\Sigma\colon \begin{cases} \dot{x}_1 = ax_1 + x_2^2 \\ \dot{x}_2 = bx_1^2 + x_2 \\ y = x_1 \end{cases}$$

Calculate Lie derivatives recursively:

$$\Pi\colon \begin{cases} y = x_1 \\ y' = ax_1 + x_2^2 \\ y'' = a\dot{x}_1 + 2x_2\dot{x}_2 = a(ax_1 + x_2^2) + 2x_2(bx_1^2 + x_2) \end{cases}$$

Better way:

$$\tilde{\Pi}\colon \begin{cases} y = x_1 \\ y' = ax_1 + x_2^2 = ay + x_2^2 \\ y'' = ay' + 2x_2\dot{x}_2 = ay' + 2x_2(by^2 + x_2) \end{cases}$$

First attempt: efficiency problems

$$\Sigma\colon \begin{cases} \dot{x}_1 = a x_1 + x_2^2 \\ \dot{x}_2 = b x_1^2 + x_2 \\ y = x_1 \end{cases}$$

Calculate Lie derivatives recursively:

$$\Pi\colon \begin{cases} y = x_1 \\ y' = a x_1 + x_2^2 \\ y'' = a\dot{x}_1 + 2x_2\dot{x}_2 = a(a x_1 + x_2^2) + 2x_2(b x_1^2 + x_2) \end{cases}$$

Better way:

$$\tilde{\Pi}\colon \begin{cases} y = x_1 \\ y' = a x_1 + x_2^2 = a y + x_2^2 \\ y'' = a y' + 2x_2\dot{x}_2 = a y' + 2x_2(b y^2 + x_2) \end{cases}$$

Idea of our algorithm: Eliminate state variables as soon as we can.

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Initialiate system:

$$\Sigma_0\colon \begin{cases} y - x_1 = 0 & (R_0) \\ \dot{x}_1 - (ax_1 + x_2^2) = 0 & (S_1) \\ \dot{x}_2 - (bx_1^2 + x_2) = 0 & (S_2) \end{cases}$$

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Initialiate system:

$$\Sigma_0 : \begin{cases} y - x_1 = 0 & (R_0) \\ \dot{x}_1 - (a x_1 + x_2^2) = 0 & (S_1) \\ \dot{x}_2 - (b x_1^2 + x_2) = 0 & (S_2) \end{cases}$$

Choose state variable to eliminate: x_1. Criteria: lowest degree.

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Initialiate system:

$$\Sigma_0\colon \begin{cases} y - x_1 = 0 & (R_0) \\ \dot{x}_1 - (ax_1 + x_2^2) = 0 & (S_1) \\ \dot{x}_2 - (bx_1^2 + x_2) = 0 & (S_2) \end{cases}$$

Choose state variable to eliminate: x_1. Criteria: lowest degree.
Elimination of x_1:

Differentiate R_0:

$$\begin{cases} y' - \dot{x}_1 = 0 & (R_0') \\ \dot{x}_1 - (ax_1 + x_2^2) = 0 & (S_1) \\ \dot{x}_2 - (bx_1^2 + x_2) = 0 & (S_2) \end{cases} \Longrightarrow$$

Eliminate $\dot{\mathbf{x}}$ from R_0':

$$\begin{cases} y' - (ax_1 + x_2^2) = 0 & (\langle R_0' \rangle + \langle S_1 \rangle) \\ \dot{x}_2 - (b{x_1}^2 + x_2) = 0 & (S_2) \end{cases}$$

Eliminate x_1 using R_0:

$$\Longrightarrow \Sigma_1\colon \begin{cases} y' - (ay + x_2^2) = 0 & (R_1) \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases}$$

Our algorithm: example

Idea: Eliminate $x_1, x_2, \dots, x_n$ one by one in a dynamically defined order.

Previous system:

$$\Sigma_1\colon \begin{cases} y' - ay - x_2^2 = 0 & (R_1) \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases}$$

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Previous system:

$$\Sigma_1 \colon \begin{cases} y' - ay - x_2^2 = 0 & (R_1) \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases}$$

Choose state variable to eliminate: x_2.

Our algorithm: example

Idea: Eliminate $x_1, x_2, \dots, x_n$ one by one in a dynamically defined order.

Previous system:

$$\Sigma_1 : \begin{cases} y' - ay - x_2^2 = 0 & (R_1) \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases}$$

Choose state variable to eliminate: x_2.
Elimination of x_2**:**

Differentiate R_1:

$$\begin{cases} y'' - ay' - 2x_2\dot{x}_2 = 0 & (R_1') \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases} \Longrightarrow$$

Eliminate $\dot{\mathbf{x}}$ from R_1:

$$y'' - ay' - 2x_2(by^2 + x_2) = 0$$

$$(\langle R_1' \rangle + \langle S_2 \rangle)$$

Eliminate x_2 using R_1:

$$\Longrightarrow \mathrm{Res}_{x_2}(y' - ay - x_2{}^2, y'' - ay' - 2x_2(by^2 + x_2)) = 0 \qquad (R_2)$$

Our algorithm: example

Idea: Eliminate $x_1, x_2, \ldots, x_n$ one by one in a dynamically defined order.

Previous system:

$$\Sigma_1 \colon \begin{cases} y' - ay - x_2^2 = 0 & (R_1) \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases}$$

Choose state variable to eliminate: x_2.
Elimination of x_2:

Differentiate R_1:

$$\begin{cases} y'' - ay' - 2x_2\dot{x}_2 = 0 & (R_1') \\ \dot{x}_2 - (by^2 + x_2) = 0 & (S_2) \end{cases} \Longrightarrow$$

Eliminate $\dot{\mathbf{x}}$ from R_1:

$$y'' - ay' - 2x_2(by^2 + x_2) = 0$$
$$(\langle R_1' \rangle + \langle S_2 \rangle)$$

Eliminate x_2 using R_1:

$$\Longrightarrow \mathrm{Res}_{x_2}(y' - ay - x_2^2, y'' - ay' - 2x_2(by^2 + x_2)) = 0 \qquad (R_2)$$

Input-output equation is irreducible: **factorization**

Factorize R_2 and choose the correct factor with a plug-in.

Our algorithm: extraneous factors

We are using repeated univariate resultant: $(R_0), (R_1)$ are “pivots”. This causes **extraneous factors**.

Smallest nontrivial example:
Given $f, g, h \in \mathbb{C}[x, y, z]$, find $\langle f, g, h \rangle \cap \mathbb{C}[z]$.

Our algorithm: extraneous factors

We are using repeated univariate resultant: $(R_0), (R_1)$ are "pivots". This causes **extraneous factors**.

Smallest nontrivial example:
Given $f, g, h \in \mathbb{C}[x, y, z]$, find $\langle f, g, h\rangle \cap \mathbb{C}[z]$.

Repeated resultant approach:
Find $\mathrm{Res}_x(\mathrm{Res}_y(f, g), \mathrm{Res}_y(f, h))$, then factorize.

Our algorithm: extraneous factors

We are using repeated univariate resultant: $(R_0),(R_1)$ are "pivots". This causes **extraneous factors**.

Smallest nontrivial example:
Given $f,g,h \in \mathbb{C}[x,y,z]$, find $\langle f,g,h\rangle \cap \mathbb{C}[z]$.

Repeated resultant approach:
Find $\mathrm{Res}_x(\mathrm{Res}_y(f,g),\mathrm{Res}_y(f,h))$, then factorize.

Justification (L. Busé, B. Mourrain):

$$\mathrm{Res}_x(\mathrm{Res}_y(f,g),\mathrm{Res}_y(f,h)) = \underbrace{\mathrm{Res}_{x,y}(f,g,h)}_{\langle f,g,h\rangle\cap\mathbb{C}[z]}\underbrace{\mathrm{Res}_{x,y,y'}(f,\delta_{y,y'}f,g(y),h(y'))}_{\text{Extraneous factor}}$$

$(\delta_{y,y'}f = \frac{f(y)-f(y')}{y-y'})$. Acceptable if f has low degree in y: justifies the choice of variable to eliminate by lowest degree.

Eliminate extraneous factor before computation?

Optimization 1: early detection of extraneous factors

We use the *Bézout matrix* to compute resultant.

Bézout matrix:
Let $f(z) = \sum_{i=0}^{n} u_i z^i, g(z) = \sum_{i=0}^{n} v_i z^i$,

$$\frac{f(x)g(y) - f(y)g(x)}{x - y} = \sum_{i,j=0}^{n-1} b_{ij} x^i y^j$$

$$B_n(f, g) = (b_{ij})_{i,j=0,\ldots,n-1}$$

$$\det B_n(f, g) = \text{Res}(f, g)$$

Optimization 1: early detection of extraneous factors

Example:

$$\begin{aligned}
&\mathrm{Res}_x(ax^2 + yx + 2(b+e),\, ax^2 + cx + (b+e)) \\
=&\det\begin{vmatrix} a(c-y) & -a(b+e) \\ -a(b+e) & (b+e)(y-2c) \end{vmatrix} \\
=&a(b+e)\underbrace{(3cy - y^2 - 2c^2 - ab - ae)}_{\text{We want this}}
\end{aligned}$$

Optimization 1: early detection of extraneous factors

Example:

$$\begin{aligned}
&\mathrm{Res}_x(ax^2 + yx + 2(b+e), ax^2 + cx + (b+e)) \\
&= \det \begin{vmatrix} a(c-y) & -a(b+e) \\ -a(b+e) & (b+e)(y-2c) \end{vmatrix} \\
&= a(b+e)\underbrace{(3cy - y^2 - 2c^2 - ab - ae)}_{\text{We want this}}
\end{aligned}$$

Actual computation:

$$\det \begin{vmatrix} (c-y) & -1 \\ -a(b+e) & (y-2c) \end{vmatrix}$$

Essential simplification as this happens often (equations not generic)

Optimization 2: variable change

Optimization 1 is straightforward.

However, **in most cases, we cannot always detect extra factors in matrix:** we can do it when the extra factor divides the constant term or the leading term.

$$\mathrm{Res}_x(ax^2 + yx + 2(b+e), ax^2 + cx + (b+e)) = \det \begin{vmatrix} a(c-y) & -a(b+e) \\ -a(b+e) & (b+e)(y-2c) \end{vmatrix}$$

Optimization 2: variable change

Optimization 1 is straightforward.

However, **in most cases, we cannot always detect extra factors in matrix:** we can do it when the extra factor divides the constant term or the leading term.

$$\mathrm{Res}_x(ax^2 + yx + 2(b+e), ax^2 + cx + (b+e)) = \det \begin{vmatrix} a(c-y) & -a(b+e) \\ -a(b+e) & (b+e)(y-2c) \end{vmatrix}$$

Counter-example:

$$\begin{aligned} &\mathrm{Res}_x(x^2 + x(y+2) + y + 1 + 2b, x^2 + x(c+2) + b + c + 1) \\ &= \det \begin{vmatrix} c-y & c-y-b \\ c-y-b & yb-y-2bc+c \end{vmatrix} \\ &= b\underbrace{(3cy - y^2 - 2c^2 - b)}_{\text{We want this}} \end{aligned}$$

Optimization 2: variable change

However, there is one trick:

If an extraneous factor $p(y) | \text{Res}_x(f(x, y), g(x, y))$, then

$$\text{Res}_x(f(x, y), g(x, y)) \equiv 0 \mod p(y)$$

so f, g share some common root $x_0 \mod p(y)$:

$$\exists x_0, f(x_0, y) = g(x_0, y) = 0 \mod p(y)$$

Optimization 2: variable change

However, there is one trick:

If an extraneous factor $p(y) | \text{Res}_x(f(x,y), g(x,y))$, then

$$\text{Res}_x(f(x,y), g(x,y)) \equiv 0 \mod p(y)$$

so f, g share some common root $x_0 \mod p(y)$:

$$\exists x_0, f(x_0,y) = g(x_0,y) = 0 \mod p(y)$$

If we consider f, g as polynomials in $x - x_0$, then $p(y)$ divides their constant terms:

$$\begin{cases} f = a_d(y)(x-x_0)^d + \ldots + a_1(y)(x-x_0) + p(y)q_a(y) \\ g = b_d(y)(x-x_0)^d + \ldots + b_1(y)(x-x_0) + p(y)q_b(y) \end{cases}$$

Optimization 2: variable change

However, there is one trick:

If an extraneous factor $p(y) | \mathrm{Res}_x(f(x,y), g(x,y))$, then

$$\mathrm{Res}_x(f(x,y), g(x,y)) \equiv 0 \mod p(y)$$

so f, g share some common root $x_0 \mod p(y)$:

$$\exists x_0, f(x_0, y) = g(x_0, y) = 0 \mod p(y)$$

If we consider f, g as polynomials in $x - x_0$, then $p(y)$ divides their constant terms:

$$\begin{cases} f = a_d(y)(x - x_0)^d + \ldots + a_1(y)(x - x_0) + p(y)q_a(y) \\ g = b_d(y)(x - x_0)^d + \ldots + b_1(y)(x - x_0) + p(y)q_b(y) \end{cases}$$

Do variable change $x - x_0 \longrightarrow x$:

$$\begin{cases} f = a_d(y)x^d + \ldots + a_1(y)x + p(y)q_a(y) \\ g = b_d(y)x^d + \ldots + b_1(y)x + p(y)q_b(y) \end{cases}$$

$$\mathrm{Res}_x(f, g) = \det \begin{vmatrix} \ldots & p(y)q_a(y) \\ \vdots & \vdots \\ \ldots & p(y)q_b(y) \end{vmatrix}$$

Optimization 2: performance

Example: when we know *a priori* that $x - y$ is an extra factor.

$$\begin{vmatrix} x & y \\ y & x \end{vmatrix} \longrightarrow \begin{vmatrix} x & -(x-y) \\ y & x-y \end{vmatrix} \longrightarrow (x+y)(x-y)$$

Equivalence with some "optimal" elimination order in sparse resultant?

Good acceleration in practice:

Model	Without Opt.2	With Opt.2
SIWR	47s	9s
Extended SEIR	>5h	22s
Pharmacokinetics	227s	20s
QWWC	1020s	236s

Table: Time comparison for computing io-equation

Summary

Our algorithm:

1. Eliminate $x_1, \ldots, x_n$ one by one in a dynamically chosen order (always do low degree variable first). Eliminations are done using resultant (Bézout matrix).
   We can regard it as a gradual ordering change (replacing a state variable x_i with a higher order of y).
2. Factorize intermediate results as often as possible, eliminate extraneous factors with plug-ins.
3. "Reveal" extraneous factors in Bézout matrix with variable change.
4. Eliminate extraneous factors in resultants before determinant computation.

Multiple output case: work in progress

Setup:

Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{6}$$

Now, $\mathbf{y} = (y_1, \ldots, y_m)$ is a vector of outputs, $\mathbf{g} = (g_1, \ldots, g_m)$ is a vector of rational functions.

Multiple output case: work in progress

Setup:

Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{6}$$

Now, $\mathbf{y} = (y_1, \ldots, y_m)$ is a vector of outputs, $\mathbf{g} = (g_1, \ldots, g_m)$ is a vector of rational functions.

Goal: Find a set of "minimal" consequences of Σ depending only on input, output, and parameters, also known as *input-output equations*:

$$\begin{cases} \phi_1(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1+1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \phi_2(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2+1)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \vdots \\ \phi_m(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m+1)}) = 0 \end{cases}$$

This means:

1. $\phi_i, i = 1, \ldots, m$ vanish on every solution of the system Σ.
2. $\phi_i, i = 1, \ldots, m$ are irreducible polynomials.
3. The sum $h_1 + h_2 + \ldots + h_m$ is as small as possible.

Multiple output case: work in progress

Setup:

Given a system Σ of the form

$$\Sigma\colon \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \end{cases} \tag{6}$$

Now, $\mathbf{y} = (y_1, \ldots, y_m)$ is a vector of outputs, $\mathbf{g} = (g_1, \ldots, g_m)$ is a vector of rational functions.

Goal: Find a set of "minimal" consequences of Σ depending only on input, output, and parameters, also known as *input-output equations*:

$$\begin{cases} \phi_1(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1+1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \phi_2(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2+1)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \vdots \\ \phi_m(\boldsymbol{\theta}, \mathbf{u}, \mathbf{u}', \ldots, y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m+1)}) = 0 \end{cases}$$

This means:

1. $\phi_i, i = 1, \ldots, m$ vanish on every solution of the system Σ.
2. $\phi_i, i = 1, \ldots, m$ are irreducible polynomials.
3. The sum $h_1 + h_2 + \ldots + h_m$ is as small as possible.
4. Moreover, we want the set $\{\phi_1, \ldots, \phi_m\}$ to have some special structure.

Multiple output case: example

$$\Sigma\colon \begin{cases} \dot{x}_1 = x_2 + x_3 \\ \dot{x}_2 = a x_3 \\ \dot{x}_3 = b x_1 \\ y_1 = x_1 & (R_1) \\ y_2 = x_2 & (R_2) \end{cases} \tag{7}$$

Good sets of IO-equations:

$$\Phi_1\colon \begin{cases} y_1'' = b y_1 + a y_1' - a y_2 & (R_1) \\ y_2' = a y_1' - a y_2 & (R_2) \end{cases} \qquad \Phi_2\colon \begin{cases} y_1' = y_2 + \frac{y_2'}{a} & (R_1) \\ y_2'' = a b y_1 & (R_2) \end{cases}$$

More than one possible set: multiple choice of differentiation.

$$R_1, R_2, R_2 \longrightarrow \Phi_1$$

$$R_1, R_2, R_1 \longrightarrow \Phi_2$$

Multiple output case: search tree

Find the "most simple" IO-equation set! Heuristics: degree, balance, etc.

$$\begin{cases} y_1 = x_1 & (R_1) \\ y_2 = x_2 & (R_2) \end{cases}$$

$$\Big\downarrow R_1 \text{ pivot}$$

$$\begin{cases} y_1' - x_2 + x_3 = 0 & (R_1) \\ y_2 - x_2 = 0 & (R_2) \end{cases}$$

$$\Big\downarrow R_2 \text{ pivot}$$

$$\begin{cases} y_1' - y_2 + x_3 = 0 & (R_1) \\ y_2' - a x_3 = 0 & (R_2) \end{cases}$$

$$\Big\downarrow R_1 \text{ pivot} \qquad \qquad \Big\downarrow R_2 \text{ pivot}$$

$$\Phi_1\colon \begin{cases} y_1'' = b y_1 + a y_1' - a y_2 & (R_1) \\ y_2' = a y_1' - a y_2 & (R_2) \end{cases} \qquad \Phi_2\colon \begin{cases} y_1' = y_2 + \frac{y_2'}{a} & (R_1) \\ y_2'' = a b y_1 & (R_2) \end{cases}$$

Special structure of input-output equation set

Generalizing our algorithm on elimination, we get a set of polynomials:

$$\begin{cases} \phi_1(y_1, y_1', \ldots, y_1^{(h_1+1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \phi_2(y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2+1)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \vdots \\ \phi_m(y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m+1)}) = 0 \end{cases}$$

Moreover, we want the set $\{\phi_1, \ldots, \phi_m\}$ to have some special structure.

Proposition

If $\{\phi_1, \ldots, \phi_m\}$ *is a characteristic set of the ideal of input-output equations, then their coefficients generate the field of all identifiable functions.*

Special structure of input-output equation set

Generalizing our algorithm on elimination, we get a set of polynomials:

$$\begin{cases} \phi_1(y_1, y_1', \ldots, y_1^{(h_1+1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \phi_2(y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2+1)}, \ldots, y_m, \ldots, y_m^{(h_m)}) = 0 \\ \vdots \\ \phi_m(y_1, y_1', \ldots, y_1^{(h_1)}, y_2, \ldots, y_2^{(h_2)}, \ldots, y_m, \ldots, y_m^{(h_m+1)}) = 0 \end{cases}$$

Moreover, we want the set $\{\phi_1, \ldots, \phi_m\}$ to have some special structure.

Proposition

If $\{\phi_1, \ldots, \phi_m\}$ is a characteristic set of the ideal of input-output equations, then their coefficients generate the field of all identifiable functions.

Proposition

Note q_i the leading coefficient of ϕ_i with respect to the variable y_i,
$Q = q_1 \cdot \ldots \cdot q_m$.
If $\langle \phi_1, \ldots, \phi_m \rangle : Q^\infty$ is prime, then their coefficients generate the field of all identifiable functions.

Consequence: if we can verify primality of $\langle \phi_1, \ldots, \phi_m \rangle : Q^\infty$, then we can use the coefficients of $\{\phi_1, \ldots, \phi_m\}$.

Bad IO-equation set

Not all minimal IO-equation sets can be used to assess identifiability:

$$\Sigma \colon \begin{cases} \dot{x}_1 = \frac{1+{x_1}^2}{2} \\ \dot{x}_2 = \frac{1-{x_1}^2}{1+{x_1}^2} \\ y_1 = \frac{2x_1}{b(1+{x_1}^2)} & (R_1) \\ y_2 = x_2 & (R_2) \end{cases}$$

$$\Phi \colon \begin{cases} b^2 {y_1'}^2 + b^2 {y_1}^2 - 1 = 0 \\ {y_2'}^2 + b^2 {y_1}^2 - 1 = 0 \end{cases}$$

However, b is in fact identifiable because

$$b y_1' - y_2' = 0$$

is a consequence of Σ.

Bad IO-equation set

Not all minimal IO-equation sets can be used to assess identifiability:

$$\Sigma\colon \begin{cases} \dot{x}_1 = \frac{1+{x_1}^2}{2} \\ \dot{x}_2 = \frac{1-{x_1}^2}{1+{x_1}^2} \\ y_1 = \frac{2x_1}{b(1+{x_1}^2)} & (R_1) \\ y_2 = x_2 & (R_2) \end{cases}$$

$$\Phi\colon \begin{cases} b^2 {y_1'}^2 + b^2 {y_1}^2 - 1 = 0 \\ {y_2'}^2 + b^2 {y_1}^2 - 1 = 0 \end{cases}$$

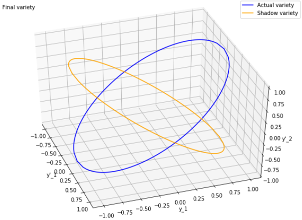
However, b is in fact identifiable because

$$by_1' - y_2' = 0$$

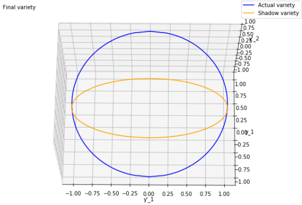
is a consequence of Σ.

Solution: Pose $z = y_1' + y_2'$ as a new variable ("dummy output") and compute a new IO-equation.

Extra projection: illustration



(a) Reconstruction from projections



(b) Extra projection

Figure: Illustration of bad IO-equation set:
Blue variety: actual ODE solution
Orange variety: extraneous variety seen from projections
An extra projection can distinguish the actual solution.

Other possible improvements

- Alternative methods to extra projection?
- Works well when most eliminations are linear, otherwise huge extraneous factors.
- Better/more concise representation for input-output equations?