Discrete Mathematics 225 (2000) 173–191

DISCRETE MATHEMATICS

On algebraic identification of causal functionals

C. Hespel[a], G. Jacob[b, *]

[a]LANS, INSA, 20 Avenue Buttes de Coësmes, 35 043 Rennes Cedex, France
[b]LIFL (URA 369 CNRS), Université Lille I, 59024 Villeneuve d'Ascq Cedex, France

Received 30 August 1998; revised 27 April 1999; accepted 28 April 2000

Abstract

We present here a second step in solving the *Algebraic Identification Problem* for the causal analytic functionals in the sense of Fliess. These functionals are symbolically represented by noncommutative formal power series $G=\sum_{w\in\mathscr{X}^\star}\langle G\,|\,w\rangle w$, where w is a word on a finite-encoding alphabet $\mathscr{X}$. The problem consists in computing the coefficients $\langle G\,|\,w\rangle$ from the choice of a finite set of informations on the input/output behaviour of the functional. In a previous work, we already presented a first step: we showed that one can compute the contributions of G relative to a family of noncommutative polynomials $\boldsymbol{g}_\mu$ with integer coefficients, indexed by the set of partitions. Hence it remains to *inverse these relations* by computing the words w as linear combinations of the $\boldsymbol{g}_\mu$. An answer could be found in two ways: firstly by providing an *identification computation tool*, secondly by solving the '*Identifiability Problem*': is the previous identification effectively computable at any order? A computational tool is here presented, in the form of a concise Maple package IDENTALG that computes the polynomials $\boldsymbol{g}_\mu$ by a block recursive matrix implementation, and allows then to *test the identification* (*when possible*) *at any order* by matrix inversion. It requires a combinatorial study of the differential monomials on the inputs. The computation of a test set covering the identification of 2048 words is presented. This package is given in the widely significant case of functionals depending on 'a single input with drift part'. It can be used without change in case of 'two inputs without drift'. It could be extended very easily to the case of 'several inputs with drift part'. Finally, we discuss the *Identifiability Problem*: we summarize the current state of our results, and we conclude with a conjecture in a weak form and in a strong form. © 2000 Published by Elsevier Science B.V. All rights reserved.

1. Introduction

The causal functionals (see [3]) are the time input/output functionals that can be described symbolically by some noncommutative formal power series. The corresponding functional is recovered via Chen iterated integrals (see [1]). We present first the background, as can be found also in [12], and the definition of the Chen series of an

* Corresponding author.
E-mail address: jacob@lifl.fr (G. Jacob).

input. The precise definition of the ‘Algebraic Identification Problem’ is then given. The iterated derivatives of the output are then computed by means of the iterated derivatives of the Chen series. This allows to state the ‘generic equation’, that summarizes all differential properties of causal functionals.

1.1. Causal functionals

Let us consider the n-dimensional dynamical system

$$(\Sigma)\quad \begin{cases} \dot{q} = f_0(q) + \sum_{j=1,\dots,m} f_j(q)a_j(t), \\ y(t) = h(q(t)) \end{cases}$$

- $\boldsymbol{a}(t) = (a_j(t))_{j=1,\dots,m}$ is the real m-dimensional input,
- $q(t) \in \mathscr{V}$ is the current state, with $\mathscr{V}$ a real differentiable manifold,
- $\{f_j(q)\}_{j=1,\dots,m}$ is a family of smooth vector fields on $\mathscr{V}$,
- $h : \mathscr{V} \mapsto \mathbb{R}$ is a smooth function called the ‘observation map’.
- $y(t) \in \mathbb{R}$ is the output.

The functional that associates the input $\boldsymbol{a}(t)$ with the resulting output $y(t)$ of the system (*initialized in a state* $q(0) = q_0$) is called the input/output behaviour.

The vector field f_0 is called *the drift* of (Σ). It is currently studied by introducing a fictitious input $a_0(t) \equiv 1$. A system without drift ($f_0 \equiv 0$) is also called *homogeneous in the inputs*. With the $m+1$ vector fields $f_i(q)$ we associate an alphabet $\mathscr{Z} = \{z_0, z_1, \dots, z_m\}$ of $(m+1)$ letters. Each word $w = z_{i_1}z_{i_2}\cdots z_{i_k}$ on $\mathscr{Z}$ can be viewed as a multi-index. It belongs to the free monoid $\mathscr{Z}^\star$ on $\mathscr{Z}$. The *empty word* on $\mathscr{Z}$ is denoted ε.

Let n be the dimension of the state variety $\mathscr{V}$. The integration of the system, expressed in local coordinates, can begin as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}h(q) = \sum_{i=1}^{n} \frac{\partial h}{\partial q^i}\frac{\mathrm{d}}{\mathrm{d}t}q_i = \sum_{i=1}^{n}\left[f_0^i(q) + \sum_{j=1}^{m} f_j^i(q)a_j(t)\right]$$

$$\frac{\partial}{\partial q^i}h(q) = \sum_{j=0}^{m}(f_j \circ h)(q)a_j(t),$$

where $f_j \circ h$ is the Lie derivative of h along the vector field f_j. A standard iterative integration process then allows the computation of the output of the system (Σ) as a power series expansion, as shown by Dyson [2] (see also Lappo-Danilevsky, [13]), and also known, in the control theory context, as the ‘Peano–Baker formula’ (see [3, 11, 12]). It can be interpreted as a separation property between input and geometric contributions. This power series expansion is

$$y(t) = \langle G_\Sigma \,||\, \mathscr{C}_{\boldsymbol{a}}(t)\rangle = \sum_{w\in\mathscr{Z}^\star} \langle G_\Sigma \,|\, w\rangle\langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, w\rangle \tag{1}$$

- $\langle G_\Sigma \,|\, w\rangle$ is the *geometric contribution* (independent of input). We have $\langle G_\Sigma \,|\, \varepsilon\rangle = h_{|q0}$, and for any word $w = z_{i_1} z_{i_2} \cdots z_{i_k}$, the coefficient $\langle G_\Sigma \,|\, w\rangle$ is the following iterated Lie derivative of h (evaluated in q_0):

$$\langle G_\Sigma \,|\, z_{i_1} z_{i_2} \cdots z_{i_k}\rangle = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_k} \circ h_{|q0} \tag{2}$$

- $\langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, w\rangle$, the *input contribution* (independent of system), is the iterated integral recursively defined as follows: $\langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, \varepsilon\rangle = 1$, and for any $w = uz_j$, $u \in \mathscr{Z}^\star$, $z_j \in \mathscr{Z}$:

$$\langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, uz_j\rangle = \int_0^t \langle \mathscr{C}_{\boldsymbol{a}}(\tau) \,|\, u\rangle a_j(\tau)\,\mathrm{d}\tau. \tag{3}$$

These two infinite families of coefficients are summarized in two noncommutative formal power series. A *noncommutative power series S* on $\mathscr{Z}$ is any map from $\mathscr{Z}^\star$ to $\mathbb{R}$, and is usually written as a formal sum $S = \sum_{w\in\mathscr{Z}^\star} \langle S \,|\, w\rangle w$. So the geometric contributions can be summarized in the *generating series*, or *Fliess series* $G_\Sigma = \sum_{w\in\mathscr{Z}^\star} \langle G_\Sigma \,|\, w\rangle w$ of the system (Σ). The input's contributions can be summarized in the *Chen series* $\mathscr{C}_{\boldsymbol{a}} = \sum_{w\in\mathscr{Z}^\star} \langle \mathscr{C}_{\boldsymbol{a}} \,|\, w\rangle w$ of the input $\boldsymbol{a}$.

1.2. The identification problem

By the expansion formula (1), the Fliess series appears as a symbolic encoding of the input/output behaviour of the system. More generally, following Fliess [3,5], any formal power series G can be interpreted as the symbolic encoding of the *causal functional* obtained by replacing each word w by the corresponding Chen iterated integral $\langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, w\rangle$. In other words,

$$y(t) = \langle G \,||\, \mathscr{C}_{\boldsymbol{a}}(t)\rangle = \sum_{w\in\mathscr{Z}^\star} \langle G \,|\, w\rangle \langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, w\rangle.$$

A natural question is to decide if the generating series is a '*canonical form*' of the causal functionals. The answer is yes: *two power series define the same causal functional if and only if they are equal.* We know three proofs of this fact, by Fliess (see [4]), by Reutenauer [14] and by Sonntag and Wang [15]. They are discussed in [10]. None of these proofs produces a way to identify the generating series, i.e. to effectively compute the unknown coefficients of the generating series, from information or measures on its input/output behaviour.

Here we deal with a stronger property, designed as the *Algebraic Identification Problem:* can we compute the generating series of a causal functional by an iterative process, that only involves a chosen panel of polynomial inputs and the resulting jet coefficients of the outputs?

1.3. Output derivatives and Chen series

The first derivative of the output of a causal functional G is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} y(t) = \sum_{w\in\mathscr{Z}^\star} \langle G \,|\, w\rangle \left\langle \frac{\mathrm{d}}{\mathrm{d}t} \mathscr{C}_{\boldsymbol{a}}(t) \,|\, w \right\rangle$$

in which the time derivative of the Chen series appears. Hence, we have for any $u \in \mathscr{Z}^\star$ and $z_j \in \mathscr{Z}$, by Eq. (3):

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}t}\mathscr{C}_{\boldsymbol{a}}(t) \,|\, uz_j \right\rangle = \langle \mathscr{C}_{\boldsymbol{a}}(t) \,|\, u\rangle a_j(t). \tag{4}$$

Therefore, the *Chen series* satisfies the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathscr{C}_{\boldsymbol{a}}(t) = \mathscr{C}_{\boldsymbol{a}}(t)\mathscr{L}_{\boldsymbol{a}}(t)$$

with initial condition $\mathscr{C}_{\boldsymbol{a}}(0)=1$, where $\mathscr{L}_{\boldsymbol{a}}(t)$ is the polynomial $\sum_{z_j \in \mathscr{Z}} a_j(t) z_j$.

In the same way, we obtain the nth derivative of the output $y(t)$ by computing the nth derivative of the Chen series. A straightforward computation gives

$$\frac{\mathrm{d}^n}{\mathrm{d}t^n}\mathscr{C}_{\boldsymbol{a}}(t) = \mathscr{C}_{\boldsymbol{a}}(t)\boldsymbol{A}_n(t),$$

where the noncommutative polynomials $\boldsymbol{A}_n$ are recursively defined as follows:

$$\boldsymbol{A}_0(t) = 1, \quad \boldsymbol{A}_{n+1}(t) = \mathscr{L}_{\boldsymbol{a}}(t)\boldsymbol{A}_n(t) + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}_n(t). \tag{5}$$

For instance,

$$\begin{aligned}\boldsymbol{A}_2 = \mathscr{L}_{\boldsymbol{a}}\mathscr{L}_{\boldsymbol{a}} + \frac{\mathrm{d}}{\mathrm{d}t}\mathscr{L}_{\boldsymbol{a}} &= \sum_{i,j} a_i a_j \,.\, z_i z_j + \sum_j \dot{a}_j z_j \\ &= \sum_{i<j} a_i a_j \,.\, (z_i z_j + z_j z_i) + \sum_j a_j^2 \,.\, z_j^2 + \sum_j \dot{a}_j z_j.\end{aligned}$$

We are interested in computing the jets of the output at time $t=0$. Since $\mathscr{C}_{\boldsymbol{a}}(0)=0$, we obtain

$$\frac{\mathrm{d}^n}{\mathrm{d}t^n} y(0) = \sum_{w \in \mathscr{Z}^\star} \langle G \,|\, w\rangle \langle \boldsymbol{A}_n(0) \,|\, w\rangle \tag{6}$$

and consequently, in the current example (second derivative) we have

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} y(0) = \sum_{i<j} a_i(0)a_j(0)\langle G \,|\, z_i z_j + z_j z_i\rangle + \sum_j a_j^2(0)\langle G \,|\, z_j^2\rangle + \sum_j \dot{a}_j(0)\langle G \,|\, z_j\rangle,$$

where we set $\langle G \,|\, P\rangle = \sum_{w \in \mathscr{Z}^*} \langle G \,|\, w\rangle\langle P \,|\, w\rangle$ for any noncommutative polynomial P.

In this example it is clear that we can proceed in two steps. Firstly by an adequate choice of the values $a_j(0)$ and $\dot{a}_j(0)$, we identify the constants $\langle G \,|\, z_j\rangle$, $\langle G \,|\, z_j^2\rangle$ and $\langle G \,|\, z_i z_j + z_j z_i\rangle$ (for $i<j$), which we call the '*contributions*' in G of the noncommutative polynomials z_j, z_j^2 and $z_i z_j + z_j z_i$. Secondly, we have to separate the contributions $\langle G \,|\, z_i z_j\rangle$ and $\langle G \,|\, z_j z_i\rangle$ for $i<j$. But that requires to carry out the calculation to at least the third derivative $(\mathrm{d}^3/\mathrm{d}t^3)y(0)$.

In our approach, we shall proceed by following the same two computation steps.

1.4. The generic equation

Let us summarize the family of polynomials $\boldsymbol{A}_n$ in the infinite graded sum $\mathscr{A} = \sum_{n\in\mathbb{N}} \boldsymbol{A}_n$. Thus, the recurrence relations (5) must be translated into the following *generic equation*:

$$\mathscr{A} = \varepsilon + \mathscr{L}_{\boldsymbol{a}}\mathscr{A} + \frac{\mathrm{d}}{\mathrm{d}t}\mathscr{A} \quad \text{with } \mathscr{L}_{\boldsymbol{a}}(t) = \sum_{z_j\in\mathscr{Z}} a_j(t)z_j. \tag{7}$$

The series $\mathscr{A}$ is a combinatorial object that *contains all differential information on the family of causal functionals*. The nth derivative of the output is obtained by selecting the terms indexed by words of length n, and then by specifying these words as iterated derivatives of the vector fields of the system. $\mathscr{A}$, however, also allows the calculation of the contribution to the output (and its iterated derivatives) of any differential monomial on the inputs. For this purpose some combinatorial tools are required.

2. Differential monomials and coloured partitions

This section consists essentially in rephrasing the ‘generic equation’ in the *combinatorial language of partitions.*

In the case of a system without a drift part, we are led to compute with usual differential monomials. These are encoded in partitions (single input case), and coloured partitions (several inputs).

In the case of a system with a drift part, we are led to use a *special constant differential letter* a_0. So we get a kind of nonhomogeneous (coloured) partitions.

In any case, as it will be shown in the third section, each coloured partition is used as an index for a noncommutative polynomial that produces a contribution in a derivative of the output.

2.1. Partitions and single differential letter

If there is a system with a single input without drift, we become concerned with the *differential monomials* on a single differential letter $a_1 = a$. These take the form

$$a^{\mu} = (a^{(i_1-1)})^{e_1}(a^{(i_2-1)})^{e_2}\dots(a^{(i_q-1)})^{e_q}, \quad 1\leqslant i_1 < i_2 < \cdots < i_q,$$

where $a^{(j)}$ is the jth derivative of a, and $a^{(0)} = a$. The powers e_q are strictly positive integers. Such a monomial a^{μ} is indexed by the *partition*: $\mu = i_1^{e_1}i_2^{e_2}\dots i_q^{e_q}$ (where $i_j \in N^{\star}$). The integers i_j are called the *parts* of the partition μ. The *weight* and the *size* of μ are defined as follows:

$$\mathrm{wgt}(\mu) = \sum_k e_k.i_k, \qquad \mathrm{size}(\mu) = \sum_k e_k.$$

The empty partition is denoted ε. In particular, we get $\mathrm{wgt}(\varepsilon) = \mathrm{size}(\varepsilon) = 0$.

2.2. Coloured partitions and several differential letters

If there is a system with several inputs without drift, the differential monomials on the finite set $\{a_1, a_2, \dots, a_m\}$ take the form

$$\boldsymbol{a}^{\mu} = a_1^{\mu_1} a_2^{\mu_2} \dots a_m^{\mu_m},$$

where each $a_j^{\mu_j}$ is a monomial on the single differential letter a_j. The encoding alphabet $\mathscr{Z} = \{z_1, \dots, z_m\}$ is an *alphabet of colours*, in bijection with the input set $\{a_j\}_{j=1}^m$. Thus, the combinatorial encoding of $\boldsymbol{a}^{\mu}$ is the *coloured partition*

$$\mu = \mu_1 \otimes \mu_2 \otimes \cdots \otimes \mu_m,$$

where each μ_j is a one-colour partition. The tensor notation is justified in order to extend it to linear combinations of coloured partitions. With this notation, the weight and the size of μ are defined as follows:

$$\mathrm{wgt}(\mu) = \sum_{j=1,\dots,m} \mathrm{wgt}(\mu_j), \qquad \mathrm{size}(\mu) = \sum_{j=1,\dots,m} \mathrm{size}(\mu_j).$$

2.2.1. Derivation rule

The time derivative on the differential monomials is reflected in the following *derivation rule D on (coloured) partitions*:

- For a differential letter a, one has $(\mathrm{d}/\mathrm{d}t)(a^{(i_k-1)}) = a^{(i_k)}$, and then $D(i_k) = i_k + 1$.
- We extend D to any one colour partition by the Leibnitz derivation rule and commutative reordering (the result being a linear combination of partitions):

$$D(i_1^{e_1} i_2^{e_2} \dots i_q^{e_q}) = \sum_{k=1,\dots,q} \boldsymbol{e_k} \star (i_1^{e_1} \dots i_k^{e_k - 1} (i_k + 1) i_{k+1}^{e_{k+1}} \dots i_q^{e_q}). \tag{8}$$

- We extend D on coloured partitions by

$$D(\mu_1 \otimes \mu_2 \otimes \cdots \otimes \mu_m) = \sum_{j=1,\dots,m} \mu_1 \otimes \cdots \otimes \mu_{j-1} \otimes D(\mu_j) \otimes \mu_{j+1} \otimes \cdots \otimes \mu_m. \tag{9}$$

For clarity, we adopt the notation $\sum_k \boldsymbol{c_k} \star \mu_k$ for the linear combination of (coloured) partitions μ_k with coefficients $c_k \in \mathbb{R}$. For example, we get

$$D(1^1 2^3 3^2 6^1) = 2^4 3^2 6^1 + \boldsymbol{3} \star 1^1 2^2 3^3 6^1 + \boldsymbol{2} \star 1^1 2^3 3^1 4^1 6^1 + 1^1 2^3 3^2 7^1.$$

Notation 2.1. *For any (coloured) partition μ we define the symbol $\langle \mu \,|\, \tau \rangle$ by the equalities*

$$D\mu = \sum_{\tau \in (coloured)\ partitions} \langle \mu, \tau \rangle \star \tau, \quad \textit{with } \langle \mu \,|\, \tau \rangle \in \mathbb{N}. \tag{10}$$

Lemma 2.1. 1. *For any μ, the symbol $\langle \mu \,|\, \tau \rangle$ is equal to* 0 *for all but a finite set of partitions τ.*

2. *For any* τ, *the symbol* $\langle \mu \mid \tau \rangle$ *is equal to* 0 *for all but a finite set of partitions* μ.
3. *Furthermore*, $\langle \mu \mid \tau \rangle \neq 0$ *implies* $\mathrm{wgt}(\tau) = 1 + \mathrm{wgt}(\mu)$ *and* $\mathrm{size}(\tau) = \mathrm{size}(\mu)$.

2.3. Unhomogeneous coloured partitions

If there is a system with a drift part, in addition to the differential letters $\{a_1, a_2, \ldots, a_m\}$, we get a new nonfree differential letter a_0, that satisfies $a_0 \equiv 1$, and $a_0^{(n)} = 0$ for any $n > 0$. In this way, we are in an *unhomogeneous case*. The differential monomials take the form

$$\mu = 1^p \otimes v, \quad \text{with } v \text{ homogeneous coloured partition.}$$

The integer $p = \mathrm{depth}(\mu)$ is called the *depth* of μ. In addition, we get:

$$\mathrm{wgt}(1^p \otimes v) = p + \mathrm{wgt}(v) \quad \text{and} \quad \mathrm{size}(1^p \otimes v) = p + \mathrm{size}(v).$$

The derivation rule is given by

$$D(1^p \otimes v) = 1^p \otimes D(v).$$

The symbol $\langle \mu \mid \rho \rangle$ is defined exactly as in (10). Lemma 2.1 remains true unchanged.

3. Combinatorial analysis of the generic equation

3.1. Homogeneous case

Let us interpret $\mathscr{A}$ as a series $\mathscr{A} = \sum_\mu \mu \,.\, \boldsymbol{g}_\mu$ in the differential monomials μ, with coefficients $\boldsymbol{g}_\mu$ in the noncommutative polynomial algebra $\mathbb{R}\langle \mathscr{Z} \rangle$. We intend to compute recursively these polynomials. With these notations, the generic equation becomes

$$\sum_\mu \mu \,.\, \boldsymbol{g}_\mu = \varepsilon + \sum_{j=1}^{m} \sum_\sigma a_j \sigma \,.\, z_j \boldsymbol{g}_\sigma + \sum_v D(v) \,.\, \boldsymbol{g}_v. \tag{11}$$

By identifying the factor of the same coloured partition μ in both sides, we obtain the recursive equation

$$\boldsymbol{g}_\mu = \sum_{j=1}^{m} z_j \boldsymbol{g}_{\mu \rhd a_j} + \sum_{v \in (\text{coloured}) \text{ partitions}} \langle v | \mu \rangle \star \boldsymbol{g}_v, \quad \text{with } \boldsymbol{g}_\varepsilon = 1, \tag{12}$$

where the operator $\rhd$ is used as follows:

$$\boldsymbol{g}_{\mu \rhd a_j} = \begin{cases} \boldsymbol{g}_\sigma & \text{if } \mu = a_j \sigma, \text{ (in other words } \mu_j = 1\sigma_j, \text{ and } \mu_k = \sigma_k \text{ for } k \neq j), \\ 0 & \text{in other cases.} \end{cases}$$

The last sum, in accordance with the definition of the symbol $\langle v | \mu \rangle$, is in fact extended to the finite set of the coloured partitions v that are *primitives* of the partition μ.

These equations (12) appear as a linear recursive definition of the polynomials $\boldsymbol{g}_\mu$ involving either the partitions $\mu \rhd a_j$ (of size strictly smaller than $\mathrm{size}(\mu)$), or the primitive partitions of μ (of weight strictly smaller than $\mathrm{wgt}(\mu)$). These also allow an easy proof by recurrence of the result:

Lemma 3.1. *For any coloured partition $\mu = \mu_1 \otimes \cdots \otimes \mu_m$, the polynomial $\boldsymbol{g}_\mu$ is homogeneous of degree* $\operatorname{size}(\mu_j)$ *in the letter z_j, for $j = 1,\ldots,m$.*

In a previous paper (see [10]) we have shown:

Theorem 1. *For any generating series G, each contribution $\langle G \,|\, \boldsymbol{g}_\mu \rangle$ can be computed from an adequate choice of a panel of polynomial inputs, and of a finite set of jet coordinates of the corresponding outputs.*

We actually deduce that the algebraic identification problem can be solved if and only if any word $w \in \mathscr{Z}^\star$ is a linear combination of the polynomials $\boldsymbol{g}_\mu \in \mathbb{R}\langle \mathscr{Z} \rangle$.

3.2. The unhomogeneous case

In the unhomogeneous case (systems with a drift part), the recursive equations (12) become

$$\boldsymbol{g}_{1^p \otimes v} = z_0 \boldsymbol{g}_{1^{p-1} \otimes v} + \sum_{j=1,\ldots,m} z_j \boldsymbol{g}_{1^p \otimes v \triangleright a_j} + \sum_{\sigma \in \text{col. part.}} \langle \sigma | v \rangle \star \boldsymbol{g}_{1^p \otimes \sigma}, \quad \text{with } \boldsymbol{g}_{\varepsilon \otimes \varepsilon} = 1, \tag{13}$$

where it must be supposed that the first term on the right-hand side vanishes if $p = 0$. The jth term of the second sum vanishes if the jth component v_j of v has *no part equal* to 1. The third sum, extended to the primitives σ of v, is finite.

Eq. (13) also allows the easy proof by recurrence of the lemma:

Lemma 3.1 (bis). *For any unhomogeneous coloured partition $1^p \otimes \mu = 1^p \otimes \mu_1 \otimes \cdots \otimes \mu_m$, the polynomial $\boldsymbol{g}_{1^p \otimes \mu}$ is homogeneous of degree p in the letter z_0, and of degree* $\operatorname{size}(\mu_j)$ *in the letter z_j, for $j = 1,\ldots,m$.*

Theorem 1 remains true for functionals with drift part, as follows:

Theorem 1 (bis) (see [10]). *For any generating series G, if there is a drift part, each contribution $\boldsymbol{g}_{1^p \otimes v}$ can be computed from an adequate choice of a panel of polynomial inputs, and of a finite set of jet coordinates of the corresponding output.*

As in the homogeneous case, we deduce that the algebraic identification problem can be solved if and only if any word $w \in \mathscr{Z}^\star$ is a linear combination of the polynomials $\boldsymbol{g}_{1^p \otimes v} \in \mathbb{R}\langle \mathscr{Z} \rangle$.

Remark 3.1. Any solution of the algebraic identification for n inputs with drift provides a simple solution for the same problem in case of $n + 1$ inputs without drift.

4. The recursive building in matrix form

In this section, we restrict ourselves by clarity to the *significant case of one colour partition*, *in the non homogeneous case* (corresponding to a system with a single input $a_1(t) = a(t)$ with a drift part). Our purpose here is to compute explicitely the polynomials $\boldsymbol{g}_{1^p\otimes v}$. We first give a recursive definition of these polynomials. This definition requires the computation of the matrices of 'primitive coefficients'. The same tools then allow us to start a splitting procedure, which is the first step in expressing each word as a linear combination of the polynomials $\boldsymbol{g}_{1^p\otimes v}$.

In this case, the recursive equations satisfied by the polynomials $\boldsymbol{g}_{1^p\otimes v}$ can be written as

$$\boldsymbol{g}_{1^p\otimes v} - \sum_{\sigma} \langle \sigma \mid v \rangle \star \boldsymbol{g}_{1^p\otimes\sigma} = z_1 \boldsymbol{g}_{1^p\otimes v \triangleright a} + z_0 \boldsymbol{g}_{1^{p-1}\otimes v}. \tag{14}$$

The first term on the right-hand side vanishes if v has no part equal to 1, and the second term vanishes if $p = 0$. Recall that ε denotes the empty partition, and that $\boldsymbol{g}_{\varepsilon\otimes\varepsilon} = 1$ (see Eqs. (11)). In particular we obtain $\boldsymbol{g}_{\varepsilon\otimes 1^m} = z_1 \boldsymbol{g}_{\varepsilon\otimes 1^{m-1}} = \cdots = z_1^m$, and $\boldsymbol{g}_{1^p\otimes\varepsilon} = z_0 \boldsymbol{g}_{1^{p-1}\otimes\varepsilon} = \cdots = z_0^p$.

These equations bring to light the fact that *the right-hand side polynomials* split along the first occuring letter (either z_0 or z_1). This would be underlined via the following matrix notation.

4.1. The matrix encoding

Let p and m be two positive integers. Let us note $\mathscr{Z}^{m,p}$ the set of homogeneous words of degree p in z_0, and m in z_1, ordered by the lexicographical ordering with respect to $z_1 < z_0$. For each (one colour) partition v of size m, the polynomial $\boldsymbol{g}_{1^p\otimes v}$ is a linear combination of words in $\mathscr{Z}^{m,p}$. The computation of the polynomials $\boldsymbol{g}_{1^p\otimes v}$ on this basis can be done via a matrix encoding, as follows.

For any positive integers m and k, let $\mathscr{M}_m^k$ be the set of (one colour) partitions $v = j_1 j_2 j_3 \ldots j_m$ of size m that satisfy

$$1 \leqslant j_1 \leqslant j_2 \leqslant j_3 \leqslant \cdots \leqslant j_m \leqslant k+1$$

ordered by the lexicographical ordering. For each fixed positive integer p, we denote by $\boldsymbol{G}_m^k(p)$ the column vector of the polynomials $\boldsymbol{g}_{1^p\otimes v}$ for all $v \in \mathscr{M}_m^k$. All of these polynomials are linear combinations of words in $\mathscr{Z}^{m,k}$. The dimensions of the matrices $\boldsymbol{G}_m^k(p)$ are given by the following lemma:

Lemma 4.1. *The application from $\mathscr{M}_m^k$ in $\mathscr{Z}^{m,k}$ defined by*

$$j_1 j_2 \ldots j_m \mapsto z_0^{j_1-1} z_1 z_0^{j_2-j_1} z_1 z_0^{j_3-j_2} \ldots z_1 z_0^{j_m-j_{m-1}} z_1 z_0^{k+1-j_m}$$

is an ordered bijection. Their common cardinality is equal to $\binom{m+k}{m}$.

Lemma 4.2. *Expanded on the polynomial basis $\mathscr{X}^{m,p}$, the column vector $\boldsymbol{G}_m^k(p)$ appears as a rectangular matrix, and satisfies the following recursive definition*:

$$\boldsymbol{J}_m^k \boldsymbol{G}_m^k(p) = \left(\begin{matrix} \boldsymbol{G}_{m-1}^k(p) \\ 0 \end{matrix} \middle| \, \boldsymbol{G}_m^k(p-1) \right) \tag{15}$$

and in the degenerate cases

$$\boldsymbol{J}_m^k \boldsymbol{G}_m^k(0) = \begin{pmatrix} \boldsymbol{G}_{m-1}^k(0) \\ 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{G}_0^k(p) = (0) \tag{16}$$

with the following notations and conventions:

1. $\boldsymbol{J}_m^k$ is the square matrix indexed by $\mathscr{M}_m^k \times \mathscr{M}_m^k$ the coefficient of which on the row of index σ and on the column v is equal to

   $$1 \quad \text{if } \sigma = v, \qquad -\langle \sigma \,|\, v \rangle \quad \text{if } \sigma \neq v.$$

   So $\boldsymbol{J}_m^k$ is sparse, lower triangular, and with only 1's in the main diagonal.
2. According to Lemma 4.1, the dimension of $\boldsymbol{G}_m^k(p)$ is $\binom{m+k}{m} \times \binom{m+p}{m}$. It is a square matrix if and only if $k = p$. (Each matrix $\boldsymbol{G}_m^k(p)$ is a submatrix of $\boldsymbol{G}_m^{k+1}(p)$, formed by extracting adequate rows.)
3. The vertical block structure expresses the splitting of the right-hand side of Eq. (14) with respect to the first letter (either z_0 or z_1). The '0' left lower block corresponds to the rows indexed by partitions v having no part equal to 1.

If we denote $\boldsymbol{T}_m^k$ as the inverse of the matrix $\boldsymbol{J}_m^k$, we get:

Theorem 2. *The matrices $\boldsymbol{G}_m^k(p)$ can be computed by the following recursive algorithm*:

$$\boldsymbol{G}_m^k(p) = \boldsymbol{T}_m^k \left(\begin{matrix} \boldsymbol{G}_{m-1}^k(p) \\ 0 \end{matrix} \middle| \, \boldsymbol{G}_m^k(p-1) \right) \tag{17}$$

with initializing conditions

$$\boldsymbol{G}_m^k(0) = \boldsymbol{T}_m^k \begin{pmatrix} \boldsymbol{G}_{m-1}^k(0) \\ 0 \end{pmatrix}, \qquad \boldsymbol{G}_0^k(p) = (1). \tag{18}$$

This recursive definition of the matrices $\boldsymbol{G}_m^k(p)$ is implemented in the package IDENTALG by the procedure call Gstar(k, m, p).

4.2. *Computation of the matrix $\boldsymbol{J}_m^k$ and its inverse $\boldsymbol{T}_m^k$*

Let us denote $\boldsymbol{P}_m^k$ as the square matrix of primitives, indexed by $\mathscr{M}_m^k \times \mathscr{M}_m^k$, the coefficient of which on the row of index σ and on the column of index v is equal to $\langle \sigma | v \rangle$.

Definition 4.1. We say that a partition v of $\mathscr{M}_m^k$ has *level* j if it can be written as $v = 1^{m-j}\rho$ where ρ has no part equal to 1, and size$(\rho) = j$.

For $1 \leqslant l \leqslant m$, we denote $\mathscr{M}_m^k(l)$ as the set of partitions of $\mathscr{M}_m^k$ of level up to l, and $\boldsymbol{P}_m^k(l)$ as the restriction of $\boldsymbol{P}_m^k$ to the partitions of $\mathscr{M}_m^k(l)$.

We have clearly $\boldsymbol{P}_m^k = \boldsymbol{P}_m^k(m)$. The aim of this definition is to give a recursive computation of the matrix $\boldsymbol{P}_m^k$.

Proposition 4.1. *The matrices $\boldsymbol{P}_m^k(l)$ satisfy the recursive block description*

$$\boldsymbol{P}_m^k = \boldsymbol{P}_m^k(m) \quad \boldsymbol{P}_m^k(l) = \begin{pmatrix} \boldsymbol{P}_m^k(l-1) & 0 \\ ltp & \boldsymbol{P}_l^{k-1}(l) \end{pmatrix} \tag{19}$$

with initializing conditions

$$\boldsymbol{P}_m^k(0) = \boldsymbol{P}_m^0(l) = (0)$$

where the lower triangular part denoted here 'ltp' is the product of $(m-l+1)$ *and of the identity matrix of dimension*

$$\binom{l+k-2}{l-1} = \dim(\mathscr{M}_{l-1}^{k-1}),$$

prolongated by 0's to the left and to the bottom in order *to produce the convenient matrix dimensions.*

Proof. If $v = 1^{m-j}\rho$ is a partition of level j, since $D(1) = 2$, we have the equality

$$D(1^{m-j}\rho) = (\boldsymbol{m}-\boldsymbol{j})\star 1^{m-j-1}2\rho + 1^{m-j}D\rho.$$

The matrix $\boldsymbol{P}_m^k(l)$ can be computed by a decomposition in four matrix blocks, as we now show.

1. If σ and v have a level smaller than l, then the coefficient $\langle \sigma \mid v \rangle$ is the same in $\boldsymbol{P}_m^k(l)$ as in $\boldsymbol{P}_m^k(l-1)$.
2. If σ has a level smaller than l and if v has level l, then $\langle \sigma | v \rangle = 0$.
3. If σ and v have level l, then we have $\sigma = 1^{m-l}\tau$ and $v = 1^{m-l}\rho$, where the partitions $\tau, \rho \in \mathscr{M}_l^k(l)$ have no part equal to 1, and $\langle \sigma | v \rangle = \langle \tau | \rho \rangle$.
   Let us define $\hat{\tau}$ and $\hat{\rho}$ by replacing in ρ and in τ each part equal to j by the part $j-1$. Then $\langle \tau | \rho \rangle = \langle \hat{\tau} | \hat{\rho} \rangle$, and $\hat{\rho}, \hat{\tau} \in \mathscr{M}_l^{k-1}$ have level at most l. Therefore, the restriction of $\boldsymbol{P}_m^k(l)$ to the partitions of level equal to l is equal to $\boldsymbol{P}_l^{k-1}(l)$.
4. If σ has level l and if v has level smaller than l, then $\langle \sigma | v \rangle \neq 0$ if and only if v has level $l-1$, and

$$\begin{array}{lll} v = 1^{m-l+1}\rho & \text{and} & \sigma = 1^{m-l}\tau, \\ \tau = 2\rho & \text{and} & \langle v | \sigma \rangle = m-l+1. \quad \square \end{array}$$

Theorem 3. *The matrices $\boldsymbol{J}_m^k = Id - \boldsymbol{P}_m^k$ can be computed by the following recursive algorithm*:

$$\boldsymbol{J}_m^k = \boldsymbol{J}_m^k(m) \quad \boldsymbol{J}_m^k(l) = \begin{pmatrix} \boldsymbol{J}_m^k(l-1) & 0 \\ -ltp & \boldsymbol{J}_l^{k-1}(l) \end{pmatrix} \tag{20}$$

(*where* '*ltp*' *is the same as in Eq.* (19)) *with initializing conditions*

$$\boldsymbol{J}_m^k(0) = \boldsymbol{J}_m^0(l) = (1).$$

This recursive definition of the matrices $\boldsymbol{J}_m^k(l)$ is implemented in the package IDENTALG by the procedure call Jstar(k, m, l).

In the same way, we get a recursive definition of the inverse $\boldsymbol{T}_m^k$ of $\boldsymbol{J}_m^k$ that avoids any call to the procedure inverse:

Theorem 4. *The matrices $\boldsymbol{T}_m^k$ can be computed by the following recursive algorithm*:

$$\boldsymbol{T}_m^k = \boldsymbol{T}_m^k(m) \quad \boldsymbol{T}_m^k(l) = \begin{pmatrix} \boldsymbol{T}_m^k(l-1) & 0 \\ \boldsymbol{T}_l^{k-1}(l) * ltp * \boldsymbol{T}_m^k(l-1) & \boldsymbol{T}_l^{k-1}(l) \end{pmatrix} \tag{21}$$

(*where* '*ltp*' *is defined as in Eq.* 19 *and* ' $*$ ' *is the matrix product*), *with initial conditions*

$$\boldsymbol{T}_m^k(0) = \boldsymbol{T}_m^0(l) = (1).$$

This recursive definition of the matrices $\boldsymbol{T}_m^k(l)$ is implemented in the package IDENTALG by the procedure call Tstar(k, m, l). (It is quite more efficient in computation time than a call to matrix inversion.)

4.3. *Splitting analysis*

Our '*splitting algorithm*' consists in computing an inverse (or a left inverse) of the matrix $\boldsymbol{G}_m^k(p)$ in order to express any word of $\mathscr{X}^{m,p}$ as a linear combination of polynomials $\boldsymbol{g}_{1^p \otimes v}$ with size$(v) = m$.

For any integer m, and any $1 \leqslant j \leqslant k$, we denote by $\mathscr{M}_m^{[j,k]}$ the set of partitions of size m that can be written in the form

$$i_1^{j_1} i_2^{j_2} \ldots i_m^{j_m} \quad \text{with } j+1 \leqslant j_1 \leqslant j_2 \leqslant j_3 \leqslant \cdots \leqslant j_m \leqslant k+1.$$

The restriction on $\boldsymbol{G}_m^k(p)$ to the rows on $\mathscr{M}_m^{[j,k]}$ will be denoted by $\boldsymbol{G}_m^{[j,k]}(p)$. In particular, we get $\boldsymbol{G}_m^{[0,k]}(p) = \boldsymbol{G}_m^k(p)$.

The block triangular decomposition given in Eq. (15) shows that, up to left multiplication by the invertible matrix $\boldsymbol{J}_m^k$, the matrix $\boldsymbol{G}_m^k(l)$ is in a clear sense equivalent to the direct sum of the two diagonal blocks:

$$\boldsymbol{G}_m^k(p) \cong \boldsymbol{G}_{m-1}^k(p) \oplus \boldsymbol{G}_m^{[1,k]}(p-1). \tag{22}$$

With a similar notation, set $\boldsymbol{J}_m^{[j,k]}$ the restriction of $\boldsymbol{J}_m^k$ to the partitions $\mu \in \mathscr{M}_m^{[j,k]}$. It can be proved that in fact $\boldsymbol{J}_m^{[j,k]} = \boldsymbol{J}_m^{k-j}$. By a convenient restriction of Eqs. (14), we obtain again an upper block triangular decomposition

$$\boldsymbol{J}_m^{[j,k]}\boldsymbol{G}_m^{[j,k]}(p) = \left(\begin{array}{c|c} \boldsymbol{H}_{m-1}^{[j,k]}(p) & \boldsymbol{G}_m^{[j,k]}(p-1) \\ 0 & \end{array}\right), \tag{23}$$

where $\boldsymbol{H}_{m-1}^{[j,k]}(p)$ is a rectangular matrix of the same dimensions as $\boldsymbol{G}_{m-1}^{[j,k]}(p)$, namely

$$\binom{m-1+k-j}{m-1}\binom{m-1+p-j}{m-1}.$$

It can be easily computed if we know the matrix $\boldsymbol{G}_m^k(p)$.

In other words, we obtain, for $j < k+1$, up to the invertible matrix $\boldsymbol{J}_m^{k-j}$, the direct sum structure

$$\boldsymbol{G}_m^{[j,k]}(p) \cong \boldsymbol{H}_{m-1}^{[j,k]}(p) \oplus \boldsymbol{G}_m^{[j+1,k]}(p-1). \tag{24}$$

Lemma 4.3. *By iterating this process, for $p \leqslant k$, we obtain finally* (*up to restrictions and left multiplications by invertible matrices*):

$$\boldsymbol{G}_m^k(p) \cong \boldsymbol{G}_{m-1}^k(p) \oplus \boldsymbol{H}_{m-1}^{[1,k]}(p-1) \oplus \cdots \oplus \boldsymbol{H}_{m-1}^{[p-1,k]}(1) \oplus \boldsymbol{H}_{m-1}^{[p,k]}(0). \tag{25}$$

The matrix $\boldsymbol{G}_{m-1}^k(p)$ and the matrices $\boldsymbol{H}_{m-1}^{[j,k]}(p-j)$ (for $j = 1,\dots,p$) will be called the 'head matrices' of $\boldsymbol{G}_m^p(p)$.

If $k < p$, the matrix $\boldsymbol{G}_m^k(p)$ defines an underdeterminated equation systems on the unknowns in $\mathscr{X}^{m,k}$.

If $k = p$, the matrix $\boldsymbol{G}_m^p(p)$ and all its head matrices are square matrices. Therefore, the determinant of $\boldsymbol{G}_m^p(p)$ is the product of the determinants of its 'head matrices'. We can suppose already computed the first one $\boldsymbol{G}_{m-1}^k(p)$ by the same splitting process. The other head matrices can be computed by a repeated use of Eq. (23). In all our test sets, these determinants are not equal to 0. Moreover, they are strictly positive integers, and their values increase very strongly when the integer $m + p$ is increasing. (These head matrices could also be more shortly obtained, because they appear as extracted submatrices of $\boldsymbol{G}_m^p(p-1)$. This fact is used in the complete implementation of our package IDENTALG.)

In case that $k > p$, the same computation leads to an overdetermined linear system. The same splitting technique could be explored, and requires the computation of left generalized inverses.

5. Conclusion

The present identification results and tools will be used for computing all the coefficients of a given causal functional up to some word length. We can then either simulate the corresponding polynomials, or simulate a rational series approximant (that is a noncommutative Padé-type approximant), and thereby obtain a bilinear system along the lines of our previous papers [9,7]. Our particular intent is, by using these computation tools, to progress in solving the identifiability problem:

Definition 5.1. We say that the *identifiability problem can be solved* if and only if any word $w \in \mathscr{X}^{m,p}$ (for any positive integers m and p) can be computed as a linear combination of the polynomials $\boldsymbol{g}_{1^p \otimes v}$ with $\mathrm{size}(v) = m$.

Firstly, we recall our previous result on this problem. Secondly, we present the package and discuss its usefulness. Finally, we present two forms of our conjecture concerning the identifiability property.

5.1. *Our previous results and perspectives*

By using the direct sum decomposition (25), it can be shown [8] that

- $\det(\boldsymbol{G}_m^1(1)) = 1$ (for any m, it is the determinant of a binomial matrix).
- $\det(\boldsymbol{G}_m^2(2)) \in \mathbb{N}_+^*$ by a theorem of Gessel and Viennot [6] (it is the determinant of a matrix extracted from a binomial determinant, for some ordered selected sequences of rows and of columns).
- For the depth $p > 2$ the same technique could be used, but becomes very tedious.

5.2. *The package and its use*

We give in Appendix C a self-contained kernel of the package IDENTALG. It contains three recursive procedures. The first one computes the matrix $\boldsymbol{J}_m^k$ by the procedure call J(m, k). The second one computes the inverse $\boldsymbol{T}_m^k$ of the matrix $\boldsymbol{J}_m^k$ by the call T(m, k). The third one computes the matrix $\boldsymbol{G}_m^k(p)$ by the call Gstar(k, m, p). These three procedures only consist of an exact software translation of the recursive definitions (20), (21), (17) and (18).

The restriction operators are computed by calls of the Maple procedure submatrix, the dimensions of which are controlled by the binomial coefficients, given in Lemma 4.1 and in the proof of Eq. (15).

The computation of the determinant of the matrix Gstar(p, m, p) can be carried out by the call of the Maple function det. It can also be done by using the intermediate computation of the ‘head matrices’ described in Eq. (25). But these head matrices can be preferably obtained by selecting them as submatrices of the smaller matrix Gstar(p, m, p − 1). We have used these observations to optimize the time computation, in a small extension of the package IDENTALG given in Appendix C. It allowed us to compute the determinants of all the ‘head matrices’ of $\boldsymbol{G}_m^p(p)$ for $m + p \leqslant 11$ (corresponding to the identification of 2048 words of the generating series). All the computed determinants are strictly positive and quite big integers (see Appendix B).

5.3. *The conjectures*

Our previous results and our study of the structure of the matrices $\boldsymbol{G}_m^k(p)$ via the package IDENTALG lead to the following conjecture:

Conjecture 5.1 (Weak form). For any integers p and m there is a (computable) integer $k \geqslant p$ such that the matrix $\boldsymbol{G}_m^k(p)$ has full rank.

This form would suffice to solve the identifiability problem.

Conjecture 5.2 (Strong form). For any integers p and m the matrix $\boldsymbol{G}_m^p(p)$ is invertible.

This form would imply that the identifiability problem can be solved by chosing a panel of 'small order' jet coordinates for inputs and of corresponding outputs.

Appendix A: Splitting example of the matrix Gstar(4, 2, 4)

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 3 & 4 & 5 & 3 & 4 & 5 & 6 & 5 & 6 & 7 & 7 & 8 & 9 \\
1 & 3 & 6 & 10 & 15 & 4 & 7 & 11 & 16 & 9 & 13 & 18 & 16 & 21 & 25 \\
1 & 4 & 10 & 20 & 35 & 5 & 11 & 21 & 36 & 14 & 24 & 39 & 30 & 45 & 55 \\
1 & 5 & 15 & 35 & 70 & 6 & 16 & 36 & 71 & 20 & 40 & 75 & 50 & 85 & 105 \\
0 & 0 & 0 & 0 & 0 & 3 & 4 & 5 & 6 & 8 & 10 & 12 & 15 & 18 & 24 \\
0 & 0 & 0 & 0 & 0 & 4 & 7 & 11 & 16 & 19 & 28 & 39 & 51 & 69 & 106 \\
0 & 0 & 0 & 0 & 0 & 5 & 11 & 21 & 36 & 29 & 50 & 81 & 94 & 143 & 230 \\
0 & 0 & 0 & 0 & 0 & 6 & 16 & 36 & 71 & 41 & 82 & 153 & 155 & 265 & 435 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 15 & 21 & 45 & 63 & 126 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 26 & 42 & 99 & 154 & 364 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 42 & 78 & 161 & 278 & 680 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 35 & 56 & 224 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56 & 98 & 476 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 126
\end{bmatrix}.$$

The first head matrix of Gstar$(4, 2, 4)$ is a binomial matrix, equal to Gstar$(4, 1, 4)$. Its determinant is equal to 1. The determinants of the four other head matrices are, respectively,

9 84 294 and 126.

Appendix B: Examples of analysis of G(m, k) = Gstar(k, m, k)

We give here the prime decompositions of the determinants of the 'head matrices' of some $G_m^k(k)$, (except for the first one, equal to $G_{m-1}^k(k)$, that should be beforehand computed).

- analysis of Gstar$(9, 2, 9)$ (dimension: rowdim $= 55$).

The first head matrix of $\mathsf{Gstar}(9,2,9)$ is equal to $\mathsf{Gstar}(9,1,9)$ and has a determinant equal to 1. The prime factor decompositions of the nine other head matrices are given below. Their product is equal to the determinant of $\mathsf{Gstar}(9,2,9)$:

primes, (19)
primes, $(3)(17)(19)$
primes, $(2)^2(3)(17)(19)$
primes, $(2)^4(3)(13)(17)^2(19)$
primes, $(2)^3(11)(13)^2(17)^2(19)$
primes, $(2)^2(11)(13)^3(17)^2(19)$
primes, $(2)^2(11)(13)^3(17)^2(19)$
primes, $(2)^2(11)(13)^2(17)^2(19)$
primes, $(2)(11)(13)(17)(19)$
computation time = 9.000

- analysis of $\mathsf{Gstar}(9,3,9)$ (dimension: rowdim = 220).

The first head matrix of is exactly $\mathsf{Gstar}(9,2,9)$ (computed just before). The determinant of $\mathsf{Gstar}(9,3,9)$ is the product of the determinant of $\mathsf{Gstar}(9,2,9)$ and of the determinants of the other nine head matrices, whose prime decomposition is the following:

primes, $(2)^{15}(3)^8(5)^3(7)^2(11)^6(13)^{13}(17)^{14}(19)^9(23)(29)$
primes, $(2)^{20}(3)^8(5)^8(7)^4(11)^8(13)^{14}(17)^{13}(19)^8(23)^3(29)$
primes, $(2)^{17}(3)^{10}(5)^7(7)^6(11)^{10}(13)^{14}(17)^{12}(19)^8(23)^6(29)$
primes, $(2)^{19}(3)^{14}(5)^6(7)^3(11)^{12}(13)^{14}(17)^{10}(19)^8(23)^7(29)$
primes, $(2)^{12}(3)^8(5)^7(7)(11)^{13}(13)^{13}(17)^9(19)^8(23)^8(29)$
primes, $(2)^8(3)^6(5)^7(11)^{11}(13)^{14}(17)^6(19)^6(23)^7(29)$
primes, $(2)^3(3)^5(5)^7(11)^8(13)^{10}(17)^4(19)^4(23)^6(29)$
primes, $(2)^3(3)^2(5)^4(7)(11)^5(13)^6(17)^3(19)^2(23)^3(29)$
primes, $(2)(3)(5)(7)(11)^2(13)^2(17)(19)(23)(29)$
computation time = 236.000

- analysis of $\mathsf{Gstar}(7,4,7)$ (dimension: rowdim = 330).

The first head matrix of $\mathsf{Gstar}(7,4,7)$ is equal to $\mathsf{Gstar}(7,3,7)$, and should be beforehand computed. The prime factor decomposition of the determinants of the other head matrices is the following:

primes, $(2)^{21}(3)^{74}(5)^{32}(7)^{24}(11)^{55}(13)^{44}(17)^{16}(19)^{23}(23)^{10}(29)^2(31)$
primes, $(2)^{35}(3)^{58}(5)^{38}(7)^{25}(11)^{48}(13)^{36}(17)^{13}(19)^{21}(23)^{13}(29)^3(31)$
primes, $(2)^{16}(3)^{54}(5)^{26}(7)^{24}(11)^{39}(13)^{30}(17)^{11}(19)^{18}(23)^{14}(29)^4(31)$
primes, $(2)^{22}(3)^{49}(5)^{17}(7)^{15}(11)^{26}(13)^{22}(17)^8(19)^{13}(23)^{11}(29)^4(31)$
primes, $(2)^{13}(3)^{29}(5)^{12}(7)^5(11)^{17}(13)^{14}(17)^7(19)^8(23)^7(29)^4(31)$
primes, $(2)^7(3)^{17}(5)^7(11)^7(13)^8(17)^3(19)^4(23)^3(29)^3(31)$
primes, $(3)^5(5)^3(11)^2(13)^2(17)(19)(23)(29)(31)$
computation time = 8448.000

The package ‘IDENTALG’ (a short MAPLE file) can be obtained by E-mail address: jacob@lifl.fr.

Appendix C: The kernel of the package IDENTALG

- **Left and right extension by ‘0’ coefficients**

```
>       restart;with(linalg):
>       myext:= proc(cf,h,l,t)
>                     local bd;
>                     bd:= band([cf],t);
>                     extend(concat(matrix(t,l-t,0),bd),h-t,0,0)
>       end;
>       with(linalg):
```

- **Building of the matrices J(k,m)**
  The matrix J_m^k is obtained by the call jstar(k, m, m);

```
>       jstar:= proc(k,m,lev) option remember;
>             local tg,td,tmini,udiag;
>             if k=0 or lev=0 then RETURN(matrix(1,1,1)) fi;
>                   tg:= binomial(lev+k-1,lev-1);
>                   td:= binomial(lev+k-1,lev);
>                   tmini:= binomial(lev+k-2,lev-1);
>                   udiag:= myext(-m+lev-1,td,tg,tmini );
>       stackmatrix(
>                   extend(jstar(k,m,lev-1),0,td,0 ),
>                   concat(udiag,jstar(k-1,lev,lev) ) )
>       end;
>       J:= proc(k,m) option remember; jstar(k,m,m) end;
```

- **The matrix T(k,m), inverse of the matrix J(k,m)**
  The matrix T_m^k is obtained by the call tstar(k, m, m);

```
>        tstar:= proc(k,m,lev) option remember;
>              local tg,td,tmini,udiag;
>              if k=0 or lev = 0 then matrix(1,1,1)
>              else
>                    tg:= binomial(lev+k-1,lev-1);
>                    td:= binomial(lev+k-1,lev);
>                    tmini:= binomial(lev+k-2,lev-1);
>                    udiag:= multiply(tstar(k-1,lev,lev),
>                          myext(m-lev+1,td,tg,tmini ),
>                          tstar(k,m,lev-1) );
>         stackmatrix(extend(tstar(k,m,lev-1),0,td,0 ),
>              concat(udiag,tstar(k-1,lev,lev) ) )
>         fi
>    end;
>
>    T:= proc(k,m) option remember; tstar(k,m,m) end;
```

- **The identification matrices** Gstar(k, m, p)

The matrix $G^k_m(p)$ is obtained by the call Gstar(k, m, p);

```
>  Gstar:= proc(k,m,p)    option remember;
>    if k=0 or m=0 then matrix(1,1,1)
>    elif p=0 then multiply(tstar(k,m,m) ,
>           extend(Gstar(k,m-1,p), binomial(m+k-1,k-1),0,0)
>         )
>    else multiply(tstar(k,m,m),
>      concat( extend(Gstar(k,m-1,p), binomial(m+k-1,k-1),0,0),
>           Gstar(k,m,p-1) )
>         )
>    fi
>  end;
```

In case $k=p$, the square matrix $G^p_m(p)$ is obtained by the procedure call Gstar(p, m, p).

References

[1] K.T. Chen, Iterated path integrals, Bull. 83 (1977) 831–879.

[2] F.J. Dyson, The radiation theories of Tomonaga, Schwinger and Feyman, Phys. Rev. 75 (1949) 486–502.

[3] M. Fliess, Fonctionnelles casuales non linéaires et indéterminées non commutatives, Bull. Soc. Math. France 109 (1981) 3–40.

[4] M. Fliess, On the concept of derivatives and taylor expansions for nonlinear input/output systems, IEEE Conference on Decision and Control, San Antonio, Texas, 1983, pp. 643–648.

[5] M. Fliess, M. Lamnabhi, F. Lamnabhi-Lagarrigue, An algebraic approach to nonlinear functional expansion, IEEE Trans. Circuits Systems CAS-30 (1983) 554-570.

[6] I. Gessel, G. Viennot, Binomial determinants, paths, and hook length formulae, Ad. Math. 58 (1985) 300–321.

[7] C. Hespel, Truncated bilinear approximants: Carleman, finite Volterra, Padé-type, geometrical and structural automata, First European Conference on "Algebraic Computing in Control", Paris France, 1991, pp. 643–646.

[8] C. Hespel, Une étude des séries formelles non commutatives pour l'approximation et l'identification des systémes dynamiques, Technical Report, Université des Sciences et Technologies de Lille, 1998.

[9] C. Hespel, G. Jacob, Approximation of nonlinear dynamic systems by rational series, Theoret. Comput. Sci. 79 (1991) 151–152.

[10] C. Hespel, G. Jacob, First step towards exact algebraic identification, Discrete Math. 180 (1998) 211–219.

[11] G. Jacob, in: Y. Landau (Ed.). Réalisation des systémes réguliers (on bilinéaires) et les séries génératrices non commutatives, Outils et modéles mathématiques pour l' Automatique, l'Analyse des Systeémes et le Traitement du Signal, Vol. 1, CNRS, 1980, pp. 325–357 (Chapter 2.7).

[12] G. Jacob, Algebraic methods and computer algebra for nonlinear system's study, IMACS Symposium MTCS, Lille, Birkhäuser, Basel, 1983, pp. 599–608.
[13] J.A. Lappo-Danilevski, Théorie des systémes des équations différentielles linéaires, London Mathematical Society Monographs, vol. New series-7, Chelsea, New York, 1953.
[14] C. Reutenauer, private communication.
[15] Y. Wang, E.D. Sonntag, On two definitions of observation spaces, Systems Control Lett. 13 (1988) 279–289.