On non-invertibilities for Structural Analysis

Vincent de Flaugergues 1,2, Vincent Cocquempot 1, Mireille Bayart 1, and Marco Pengov 2

1 Laboratoire d’Automatique, Génie Informatique et Signal, FRE CNRS 3303 – Université Lille 1: Sciences et Technologies, 59655 Villeneuve d’Ascq Cedex, France. vincent.cocquempot@univ-lille1.fr mireille.bayart@univ-lille1.fr
2 PSA Peugeot Citroën, VV 1404, Route de Gisy, 78943 Velizy-Villacoublay, France. vincent.deflaugergues@mpsa.com marco.pengov@mpsa.com

ABSTRACT

This article deals with structural analysis, which is a simple but efficient method in the field of Fault Detection and Isolation (FDI), to determine systems properties, such as observability, fault detectability or diagnosability. Moreover, it allows to determine subsets of the model equations which may or may not yield fault indicators, namely residuals. Because some residuals are obtained by inverting parts of the model, the notion of constraint invertibility is used to assess the possibility of building a residual. Invertibilities are often considered a posteriori, after that the structural analysis has been performed, in order to keep the computable residuals. Taking into account these invertibility constraints in all steps of the structural method would allow, firstly, to provide directly computable residuals, and secondly, to reduce the complexity of structural analysis algorithms. Two types of non-invertibilities may be distinguished: those which are defined according to the nature of the functions, and those which are due to the structure of the model. Two algorithms are proposed for determining the latter ones. Integration of the two kinds of invertibilities from the first step of the structural analysis is the objective of this paper.

1 INTRODUCTION

Structural analysis is an efficient and well-known method to analyze systems monitorability and to search for fault indicators, also called residuals, which are used for Fault Detection and Isolation (FDI). A residual is a function of known variables (measures and inputs, called observations in the following) that is computed on-line to test the consistency of these variables with the model of the system. An inconsistency between the model and the observations informs on the appearance of a fault and on its localization. Different formalizations of structural models can be found in the literature. In this paper, the structural model is a bipartite graph that represents the links between functions (or constraints) and variables (known or unknown). This bipartite graph represents the structure of the model, that is to say the way the physical variables of the model are interconnected (or constrained) by the model relations. The model relations can be non-linear, qualitative, or described by a lookup table. No precise knowledge, neither on the type of the relations, nor on the values of the parameters, is required to build the bipartite graph. A structural analysis can thus be performed at the early stages of the design of a diagnosis system, to provide structural properties of the model as for instance observability, controllability, fault detectability and isolability. It may also help the designer in identifying potential residual generators (Maquin et al., 1997) and locations of sensors with the objective to enhance the system monitorability (Frisk and Krysander, 2007), (Conrard et al., 2009), (Rosich et al., 2007), (Carpentier et al., 1997).

Classically, a structural analysis involves the following steps (Svärd and Wassén, 2006):

1. consider a behavioral model of the system to monitor, the parameters of which may not necessarily be identified;
2. extract a structural model, in the form of a bipartite graph;
3. perform the Dulmage-Mendelsohn decomposition (Dulmage and Mendelsohn, 1958) of the graph, in order to determine the monitorable subsystem, on which it is possible to generate residuals;
4. search for potential residual generators, in an exhaustive way (Dustegor et al., 2004), (Krysander et al., 2008), (Travé-Massuyès et al., 2006), (Pulido and Gonzalez, 2004), (Armengol et al., 2009)). Indeed, a structural analysis allows to determine sets of constraints from which a residual can – but not necessarily will – be derived. Residual generators correspond to a subgraph of the structural model, called ARR structure (ARR : Analytical Redundancy Relation) or MSO sets (MSO: Minimal structurally overdetermined – see section 2.3);
5. select the implementable or realizable residual generators, that is to say discard those for which a residual cannot be practically generated. There are many methods for generating residuals. The particular method we discuss in this article is the method where a computation sequence of unknown variables is determined by following paths on the bipartite graph. For this method, a residual cannot be generated when there are non-invertible constraints that should be inverted when following the computation sequence.

Motivations. Industrial models we consider in this collaborative work between PSA-Peugeot Citroën and LAGIS Laboratory, are MATLAB/SIMULINK simulation models composed of thousands of relations, among which lookup tables, cartographies, logical or conditional relations... Difficulties arising from analysing such models concern in the first place the complexity of algorithms performing the exhaustive search of potential residual generators. Current algorithms, though very efficient, cannot deal with certain models. Model simplifications can be a solution to this. To guide the search could be another one. A third possibility is to integrate feasibility criterions for residuals generation. We discuss the third possibility in this article, keeping in mind that for the most complex models, a solution would combine many ideas.

In order to take into account invertibilities, not a posteriori but at each step of the analysis, it is necessary to define them initially in the model structure. There are two types of non-invertibilities: those which are defined according to the nature of the functions, and those which are due to the structure of the model. This paper studies two particular cases of non-invertibilities related to the structure of the model, and provides two algorithms to indicate these non-invertibilities in the initial structural model:

- the first algorithm deals with differential-algebraic loops, for which integral causality is mandatory;
- the second one deals with algebraic loops, which can be detected and therefore avoided by a specific orientation of the graph.

These two kinds of non-invertibilities are thus not due to the mathematical form, or the nature of the constraints, but result from the structure of the system.

The rest of the paper is structured as follows. Section 2 recalls briefly the main concepts of structural analysis. The notion of constraint invertibility is discussed in section 3. An algorithm for defining the causality of dynamical relations, and another for avoiding algebraic loops are proposed in section 4. An academical example is presented in section 5 and illustrates the method.

2 STRUCTURAL ANALYSIS FOR FAULT DETECTION AND ISOLATION

2.1 Bipartite Graph

Consider a behavioral model of a physical system:
\[
\begin{align*}
C(x_1, \ldots, x_m, z_1, \ldots, z_k) &= 0 \\
\ldots \\
C_n(x_1, \ldots, x_m, z_1, \ldots, z_k) &= 0
\end{align*}
\]

The model relations \(\{c_i\}_{i=1}^n\) are called constraints: they link the variables, which may be known (\(\{z_i\}_{i=1}^k\)) or unknown (\(\{x_i\}_{i=1}^m\)). These constraints may be dynamic or static, expressed analytically or numerically...

The structural model of this behavioral model is defined by the graph \(G = (C, V = X \cup Z, \Gamma)\), where:
- \(C\) is a set of vertices, representing the set of constraints \(\{c_i\}_{i=1}^n\);
- \(V\) is a set of vertices, representing the set of variables \(\{v_i\}_{i=1}^{m+k}\).
- \(\Gamma\) is the set of edges.

The incidence matrix of the graph \(G\), noted \(S\), is a boolean matrix, the rows of which correspond to \(C\), and the column of which correspond to \(V\). It is defined as follows:
\[
S = \{s_{ij} \mid s_{ij} = 1 \text{ if } (c_i, v_j) \in \Gamma, 0 \text{ otherwise}\}. \tag{2}
\]

In the following, it is considered that the vertices of \(Z\) have been deleted in the structural graph, so that only the constraints and unknown variables remain.

2.2 Canonical Decomposition

The Dulmage-Mendelsohn algorithm decomposes the graph in three parts:
- \(S^+\) is the over-constrained, monitorable, part of the system.
- \(S^0 = \bigcup_{i=1}^p S^0_i\) is the just-constrained, or observable, part of the system.
- \(S^-\) is the under-constrained part of the system.

2.3 MSO, ARR structure

Definition 1. A set \(M\) of equations is structurally overdetermined if \(M\) has more equations than unknowns.

Definition 2. A structurally overdetermined set \(M\) is a proper structurally overdetermined (PSO) set if \(M = M^+\).

Definition 3. A structurally overdetermined set is a minimal structurally overdetermined (MSO) set if no proper subset is a structurally overdetermined set.
A MSO set $M$ verifies $M^+ = M$ and $|M| - |\text{var}_X(M)| = 1$ (var$_X(M)$ represents the set of unknown variables appearing in the relations of the set $M$, and $|x|$ represents the cardinal of $x$). Such a set is also called an ARR structure. The term ‘Possible Conflict’ is also used in (Pulido and Gonzalez, 2004).

MSO sets are over-constrained subsystems and are thus used to generate residuals. Different MSO-based approaches for residual generation may be found in the literature. The following list provides some of them, yet note that it is not exhaustive. Note also that two methods may result in the same solution:

- Loop-less MSO sets, which are returned by the ‘Ranking Algorithm’ described in (Blanke et al., 2006), may be used to provide a computation sequence which can be followed to generate Analytical Redundancy Relations;
- MSO sets corresponding to a state-space form (3) are considered in (Sv˚ ard and Wass´ en, 2006), where residuals are generated either in simulation (open-loop), or by using observer techniques (closed-loop). We will mention here works detailed in (˚ Aslund and Frisk, 2006), for systems under the following generic form (3);

$$
\begin{align*}
\dot{x} &= f(x, z) \\
0 &= h(x, z)
\end{align*}
$$

- Generation of residuals by simulation is performed in (Pulido and Gonzalez, 2004). (Pulido et al., 2008) deals with initial condition problems by designing an initial condition observer, placed upstream from simulated residual generators. (Calder´ on-Espinoza et al., 2007) considers limited initial conditions and interval techniques;
- Elimination methods have been proposed in (Guinez et al., 1997), (Frisk, 2000), for algebraic systems. When the MSO is not algebraic, differentiating equations ((Krysander, 2006)) may be a solution to derive an algebraic system.

Anyway, it is possible, to help residual generation, to determine an rearrangement of the equations of the MSO set which eases the residual generator design. This is done by using matchings over the structural model of the MSO set.

2.4 Matching

The following definitions can be found in (Blanke et al., 2006).

**Definition 4** (Matching). A matching is a set of non-adjacent edges, i.e. without a common vertex. The graph being bipartite, a matching corresponds to a set of couples (constraint, variable).

**Definition 5** (Maximal matching). A maximal matching is a maximal-sized matching.

**Definition 6** (Complete matching). A complete matching on a set of vertices $V$ is a matching such that all vertices of $V$ are covered, that is to say that all vertices in $V$ are matched.

Choosing a complete matching on unknown variables of the MSO set $M$ allows to rearrange the equations of $M$ and yields a rewriting of the equations in $M$. This rewriting will be useful to design the residual generator. Matching $x_j$ to $c_i(x_j, x_{k \neq j})$ is interpreted in the following way: $x_j$ is deduced from $c_i$, all remaining variables $x_k$ being supposed known. The only equation in $M$ which hasn’t been matched is said to be redundant, and acts as a consistency test. A residual may thus be generated. Graphically, a matching induces an orientation of the graph associated to the MSO set:

- if the edge $(c_i, x_j)$ belongs to the chosen matching set, it is directed from $c_i$ to $x_j$;
- otherwise, the edge is directed from $x_j$ to $c_i$.

The directed graph of the MSO takes known variables (measured and control variables) as inputs and provides a residual as output. This orientation may induce loops, which are strong connected components (SCC), the size of which is greater than 1 ((Dustegor et al., 2004)): a loop represents a system of equations which must be solved as a whole. Loops are not necessarily solvable, as it will be discussed in subsections 3.2 and 3.3. All matchings cannot lead to a residual, because the encountered SCC have to be solvable. This matter is obviously linked with non-invertibilities and causality constraints, and is discussed in the following section (3).

3 NON-INVERTIBILITIES AND SOLVABILITY OF SCC

The three following subsections present a state of the art on the solvability of SCC. Hypotheses made by the authors will be, as much as possible, stated.

3.1 SCC of size 1

For a SCC of size 1, non-invertibility considerations are generally used to evaluate its solvability.

**Definition 7.** Let a constraint $c_i(\ldots, x_j, \ldots) = 0$, if $x_j$ can be uniquely determined using $c_i$ – other variables of $c_i$ being known – then $c_i$ is said to be invertible with respect to $x_j$. In the incidence matrix, $s_{i,j} = 1$ is noted if the edge is invertible, $s_{i,j} = -1$ otherwise.

Classical non invertible contraints are tables, maps, non-linear functions, hysteresis, some logical functions, conditional functions, or functions modelling logical sensors, such as detectors. Consequently, the notion of non-invertibility allows to discard residual generators implying the inversion of such relations.

3.2 Solvability of differential algebraic SCC of size $n > 1$

Works dealing with the structural solvability of differential algebraic loops, in a numerical resolution context, have been published. In (Pulido et al., 2008), it is stated that:

- integral causality is compulsory in loops;
- differential loops are not loops but spirals, because they involve different temporal indices ((Dressler and Freitag, 1994)).
Setting integral causality comes down to writing a semi-explicit form of the differential algebraic equation (DAE). Implicit forms of a differential-algebraic equation are barely considered in the automatic control literature (should one decide to use FDI tools after the structural analysis), or by numerical solving tools (should one decide to simulate the residuals, open-loop). In the contrary, semi-explicit and explicit forms are well-documented (indeed, state-space forms are derivated from an explicit form).

As far as semi-explicit forms are concerned, two families of methods for their numerical computation exist: ‘direct’ methods and ‘ODE’ methods ([Shampine et al., 1999]). ‘ODE’ methods consist in transforming the implicit equation in an ordinary differential equation. In (Svärd and Nyberg, 2008), the reader should find explanations on:

- implicit and semi-explicit forms of differential algebraic equations;
- the differential index of a semi-explicit DAE, which indicates the difficulty of the resolution;
- solvability conditions by an ‘ODE’ method, that is to say:
  - possibility of writing a semi-explicit form;
  - possibility of solving the algebraic part, and thus write an ordinary differential equation;
  - initial condition consistency.

These criteria are stated in a simulation context, but one should note that the scope exceeds numerical resolution: indeed, a state-space form is obtained, and classic automatic techniques can be applied afterwards. In that case, structural analysis is used as a pre-analysis of the system and can be complemented with FDI techniques.

On a side note, we will also mention here links between well-posedness and the solvability of differential equations ([Vidyasagar, 1980] for instance). The criterion we will use for assessing the solvability of differential loops is that, when removing differential constraints, the resulting graph must be loop-less. Invertibilities are then considered in the (loop-less) remaining graph. Such loops are structurally equivalent to ODEs. This criterion is strongly suggested by the aforementioned references; it is also more restrictive (DAEs may not verify this criterion and still easily be solved), but, in the other hand, is structural in nature. More formally:

**Criterion 1.** Let a DAE with algebraic constraints \( C_a \) and differential constraints \( C_d \) linking variables \( \dot{x}_d, x_d, \) and \( x_a \). Here, \( x_a \) are algebraic variables, that is to say that their derivatives do not appear in the loop. \( x_d \) and \( x_d \) are differentiated variables. Let \( G_a \) the graph obtained by removing \( C_d \) and \( x_d \); this corresponds to setting integral causality. \( G_a \) is therefore defined by constraints \( C_a \) restricted to variables \( \dot{x}_d \cup x_a \). The solvability of such DAE can be assessed if:

1. \( G_a \) is loop-less;
2. the complete matching in \( G_a \) only uses invertible edges.

Indeed, let an implicit DAE \( F(\dot{x}_d, x_d, x_a, z) = 0 \), if the above criterion is verified, then it is possible to rewrite \( F \) as \( \dot{x}_d = f(x_d, z), x_a = g(x_d, z) \).

### 3.3 Solvability of algebraic SCC of size \( n > 1 \)

A structural solvability condition for algebraic loops does not exist, therefore, studying these loops must be done on a case by case basis. However, ([Rosich et al., 2009]) considers that only linear algebraic loops are structurally solvable, and, to that end, introduces a symbol \( l \) in the incidence matrix, pointing linear dependencies. In the same vein, another interesting reference is (Murota, 1987)'s Combinatorial Canonical Form decomposition. As far as we are concerned, we consider in this article that algebraic loops are not solvable.

### 4 ALGORITHMS TO DEFINE NON-INVERTIBILITIES

#### 4.1 Objectives

The general scope of this paper is the integration of non-invertibilities in the various stages of structural analysis, in order to reduce complexity of the algorithms. This work echoes some of our previous works, previously mentionned. We will focus here on the definition of the (non) invertibility constraints in the structural graph. As previously explained, non-invertibilities may be intrinsic to the constraints (and may be indicated directly on the structural model), but may also result from the structure of the system. In this section, two algorithms are presented to define some non-invertibilities relatively to the structure. These two algorithms are not based on the same hypotheses, so it is not always possible to use both, but they share the same principles.

1. the first algorithm makes the assumption of a mixed-causality approach ([Svärd and Nyberg, 2008]), therefore no causality is a priori preferred. The purpose of this algorithm is to determine which differential constraints will always appear in a loop, and therefore, set them in integral causality;
2. for the second algorithm, we must first note that the inversion of an algebraic constraint may always lead to an algebraic loop. In this case, forbidding the inversion will constrain the graph and will imply that the loop will not be taken. However algebraic loops are difficult to identify when they appear in a DAE. That’s why this second algorithm makes the assumption of a full integral causality approach, in which case algebraic loops can be identified. To sum up, the output of this algorithm is the definition of non-invertibility constraints so that the graph orientation does not lead to algebraic loops.

#### 4.2 Introduction examples

The example in figure 1 illustrates the first algorithm, whereas the example in figure 2 illustrates the second algorithm. In the first example, we will draw the reader’s attention on the following items:

- an analytical redundancy relation may be derived by solving the algebraic loop made of \( C_1 \) and \( C_2 \), then by using \( I \) as a redundant constraint, to test the consistency. In that case, \( I \) is not used in a matching;
• matching \( I \) to \( \dot{x} \) imply that \( x \) is matched to \( C_1 \) or \( C_2 \). In either case, a differential loop containing \( I \) is implied by the matching.

Since the invertibility of redundant constraints has no relevance in our criterions (non-invertibilities are used when building a matching, not when testing the consistency), we will always assume that it is not possible to take the integrator in differential causality in a matching, for it will always appear in a loop, and say that \( I \) is not invertible with respect to \( \dot{x} \). The reason, on which our algorithms are based and which we will develop further in the following, is that it is not possible to 'reach' \( x \) without using the integrator and \( \dot{x} \). If it were possible to 'reach' \( x \) without using the integrator and \( \dot{x} \), then deriving \( x \) to compute \( \dot{x} \) would be possible, i.e. it would be possible to match \( I \) to \( \dot{x} \), which would correspond to a SCC of size 1.

![Figure 1: Non-invertible differential loop](image)

The second example shows that if one try to match \( C_2 \) to \( x_1 \), then one must match \( x_2 \) to \( C_3 \), which creates an algebraic loop. Our second algorithm's purpose is to indicate that \( C_2 \) is not invertible with respect to \( x_1 \), from the beginning, to prevent matching them. Detection of the algebraic loop is done in the same way, by testing if it is possible to 'reach' \( x_2 \) without using \( C_2 \) and \( x_1 \). If it were possible, then inverting \( C_2 \) to compute \( x_1 \) would be possible and would correspond to a SCC of size 1.

![Figure 2: Structural model holding an algebraic loop](image)

Since we have considered that algebraic loops are not solvable and have given a criterion of solvability for DAEs accordingly, our notion of reachability should ideally be that there exist a path without algebraic loops, such that SCC of size 1 are invertible, and such that DAEs verify criterion 1. For practical purposes, our notion of reachability of \( x \) is less evolved, but it includes our ideal notion: it means that there is a directed path, consistent with non-invertibilities, leading to \( x \). We will see that because of this, we are only able to provide sufficient conditions in lemmas 2 and 3.

4.3 Definitions

Definition 8 (Just-constrained subgraph). A subgraph \((M, X, \Gamma)\) is just-constrained if

1. \(|M| = |X|\);
2. the set of relations \(M\) verify \(M^0 = M\).

Definition 9 (Directable just-constrained subgraph). A directable just-constrained subgraph is a just-constrained subgraph which has a complete matching using only invertible edges.

Definition 10 (Reachable variable). A variable \( x \) is said to be reachable if there exists a directable just-constrained subgraph \((M, X, \Gamma)\) containing \( x \), i.e. \( x \in X \).

To assess the reachability of a variable, we use the algorithm described in (de Flaugergues et al., 2009), which is an alternative canonical decomposition \((S^+, S^0, S^-)\), taking invertibilities into account.

Lemma 1. An unknown variable is reachable if it appears in \(S^+\) or \(S^0\).

4.4 Defining the causality

In this section, no hypothesis is made a priori on the orientation of differential constraints and it is assumed that the two orientations are possible (mixed-causality). The only restriction is that integral causality is compulsory inside loops. The key idea for identifying differential constraints which will always appear in a loop is the reachability of variables.

The following lemma may be used to determine whether a differential constraint can appear in differential causality in a strongly connected component of size 1, in which case differential causality is allowed. If not, differential causality is forbidden. More over, lemma 2 is an implication: as pointed before, it provides a sufficient but not necessary condition to forbid differential causality.

Lemma 2. A dynamical constraint \( I \) linking an integrated variable \( x_i \) and a derivated variable \( x_d \): \( I(x_i, x_d) \) cannot be taken in differential causality, in a matching, if \( x_i \) is not reachable on the bipartite graph, when the initial graph is modified as follows:

- Remove \( I \);
- Place non-invertibilities on every edge linked to \( x_d \); on the incidence matrix \( S \), the column \( x_d \) is set to \(-1\): \( S(:, x_d) = -[S(:, x_d)]\).

Proof. If it is not possible to reach \( x_i \) while \( x_d \) is not reachable, and without using \( I \), then it is not possible to use \( I \) to calculate \( x_d \). Two cases are possible: either non-invertibilities prevent to reach \( x_i \); or \( x_d \) and/or \( I \) are needed to compute \( x_i \). In either case, \( I \) cannot be in a loopless path in a matching, and the causality will always be integral in a matching.

The algorithm tests the reachability of derivated variables one after the other; it must be noted that placing an integral causality may imply that a variable which has precedentely been evaluated as reachable is no longer reachable. It is therefore necessary to reevaluate the reachability of derivated variables as long as there is no more modification.
Lemma 3. An algebraic constraint $C$ linking variables $x_1, x_p$, matched to $x_m$, will always imply an algebraic loop if one of the $x_k, k \neq m$, is not reachable on the bipartite graph, when the initial graph is modified as follows:

- Replace every differential constraint $I(x_i, x_d)$ by a measurement function on $x_i$. In other terms, set the edge between $x_d$ and $I$ to 0, and the edge between $x_i$ and $I$ to 1 (integral causality);
- Remove $C$;
- Place of non-invertibilities on every edge linked to $x_m$: on the incidence matrix $S$, the column $x_m$ is set to $-1$: $S(:, x_m) = -S(:,\{x_m\})$.

Proof. If it is not possible to reach one of the $x_k, k \neq m$, while $x_m$ is not reachable, and without using $C$, then it is not possible to use $C$ to calculate $x_m$. Two cases are possible: either the $x_k$ are not reachable due to invertibilities, or $x_m$ and/or $C$ are needed to compute $x_m$ and matching $C$ to $x_m$ implies a loop. This loop is algebraic since differential constraints have been removed (forced to integral causality).

Algorithm 1 Defining causalities

1: Identify the set of differential constraints $I$ on the graph $S$
2: Initialize $MODIF = 1$
3: while $MODIF == 1$ do
4:   $MODIF = 0$
5:   for all $I_k \in I$ for which differential causality is allowed do
6:     Identify $I_k$’s integrated variable $x_i$ and derivated one $x_d$
7:     if $I_k$ doesn’t verify lemma 2 then
8:       $I_k$ is not invertible with respect to $x_d$: modify $S$
9:     end if
10:   end if
11: end for
12: end while
13: Return $S$

4.5 Avoiding algebraic loops

In this section, we assume that a full integral causality approach is adopted. In order to detect algebraic loops, differential constraints are removed so that only algebraic constraints remain.

Lemma 4. On the bipartite graph, modified by replacing every differential constraint $I : x_i = I(x_d)$ by a measurement equation on $x_i$ (integral causality), the set of constraints which will always appear in algebraic loops is given by the ‘Ranking’ algorithm.

Proof. The ‘Ranking’ algorithm not only finds loopless ARR structures, but also identifies the loopless part of the graph (Blanke et al., 2006).

Algorithm 2 Avoiding algebraic loops v0

1: Replace differential constraints $I_k$ on the graph $S$ by sensors on the integrated variable
2: Initialize $MODIF = 1$
3: while $MODIF == 1$ do
4:   $MODIF = 0$
5:   for all invertible edge $C x_m$ do
6:       if $C$ and $x_m$ don’t verify lemma 3 then
7:         $C$ is not invertible with respect to $x_m$: modify $S$
8:         $MODIF = 1$
9:       end if
10:   end if
11: end while
12: Return $S$

Algorithm 3 Avoiding algebraic loops

1: Identify the set of differential constraints $I$ in $S$
2: Replace each differential constraint $I_k \in I$ by a sensor-like constraint $I_k': I_k'(x_i)$. Name $S'$ this new matrix.
3: $AL = \{\text{constraints in } S'\}$
4: $J = \{\text{constraints in } S' \text{ containing one unknown variable only}\}$
5: $K = \text{var}_X(J)$
6: while $J \neq \emptyset$ do
7:   $AL = AL \setminus J$
8:   Modify $S'$ by setting rows $J$ and columns $K$ to 0 (deletion of edges)
9:   $J = \{\text{constraints in } S' \text{ containing one unknown variable only}\}$
10: $K = \text{var}_X(J)$
11: end while
12: $J = \{\text{constraints in } S' \text{ which don’t contain any variable}\}$
13: $AL = AL \setminus J$
14: In $S$, set every link to $AL$ to $-1$
15: $S = \text{Avoiding Algebraic Loops v0}(S)$
16: Return $S$

Improvements

Improvements can be done thanks to a pseudo-‘Ranking Algorithm’, which identifies constraints which will always appear in algebraic loops. Many edges can thus be identified as non-invertible with a low-cost algorithm.
5 EXAMPLE

In the following example (figure 3), it is assumed that a few functions are not invertible; they are represented with an arrow indicating in which direction they must be used. Thus, $M_3$ is not invertible w.r.t. $x_1$, and $C_3$ is not invertible w.r.t. $x_2$. At first glance, there are three MSO: one linking $u$ and $y_1$, one linking $u$ and $y_2$, and the last linking $y_1$ and $y_2$. We will show that the subpart of the system between $x_1$ and $y_2$ must be taken with integral causality, and that, therefore, it is not possible to generate the residual linking $y_1$ and $y_2$ simply by following paths on the graph, which is not obvious from the beginning, since there is only one non-invertibility defined on that subpart.

Our first algorithm will proceed as follows:

- the first integrator $I_1$ is examined: its output is reachable from $y_2$. Indeed, the subgraph between $y_2$ and $x_1$ is directly accessible. No modification is made;

- the second integrator $I_2$ is examined: matching $I_2$ to $x_2$ is impossible, because this matching will always induce a differential loop. This is identified by testing that it is not possible to reach $x_2$ without using $I_2$ and $x_2$. Consequently, integral causality is compulsory;

- the first integrator is re-examined: due to non-invertibilities on $I_2$ and $C_3$, its output cannot be reached ($x_2$ cannot be matched using an invertible edge), therefore, it will always be used with integral causality.

Since full integral causality is compulsory, the second algorithm can be run: it will proceed as indicated in table 1, and will detect the algebraic loop made of constraints $C_5$ and $C_6$. We have not indicated in table 1 the tests of obvious invertible edges, i.e. $(C_1, x_1)$, $(C_2, x_2)$, $(C_3, x_3)$, $(C_4, x_4)$, $(C_5, x_5)$, $(C_6, x_6)$.

On this example, our algorithm will thus return that constraints must be used in the way indicated by the arrows on the figure 4. In the end, the definition of these non-invertibilities let us see more clearly and more obviously that it is not possible to generate the residual linking $y_1$ and $y_2$ by following paths on the graph.

6 CONCLUSION

The objective of this paper is to analyze a structural model and determine if there are equations in the model that cannot be used to compute unknown variables, when generating residuals. When these equations are found, non-invertibilities are defined to prevent them to be used. This paper eventually aims at integrating non-invertibility notions in a structural analysis – which the complexity of our models leads us to – therefore it is important to well define invertibilities in the first place. Focus has been put on defining invertibilities regarding the structure of the system, that is to say:

- differential constraints for which we can assure that they will always appear in a loop are defined with the integral causality;

- orientations of the graph which imply algebraic loops are forbidden.

This second aspect may be a bit restrictive, but our complex industrial models have led us to think that it is needed to organize the MSO sets into a hierarchy, and get intermediate or partial results, corresponding to restrictive or intermediate hypotheses like this one. Improvements could be done regarding this hypothesis (that algebraic loops are forbidden), with a view to allowing linear algebraic loops for instance.

**ACKNOWLEDGEMENTS**

This work was partially supported by the Association Nationale de la Recherche Technique (ANRT), and PSA Peugeot Citroën.
REFERENCES


