



HAL
open science

LISA: a LInear Structured system Analysis program

Sinuhé Martinez-Martinez, Theodor Mader, Taha Boukhobza, Frédéric Hamelin

► **To cite this version:**

Sinuhé Martinez-Martinez, Theodor Mader, Taha Boukhobza, Frédéric Hamelin. LISA: a LInear Structured system Analysis program. Nov 2006, pp.CDROM. hal-00121017

HAL Id: hal-00121017

<https://hal.science/hal-00121017>

Submitted on 19 Dec 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LISA: A LINEAR STRUCTURED SYSTEM ANALYSIS PROGRAM

**S. Martinez-Martinez, T. Mader, T. Boukhobza, and
F. Hamelin**

*Research Center in Automatic Control (CRAN - CNRS UMR
7039), Nancy University, BP 239, 54506 Vandœuvre Cedex,
Nancy, France, Phone: 33 383 684 464, Fax: 33 383 684 462,
email: sinuhe.martinez@cran.uhp-nancy.fr*

Abstract: In this paper the program LISA is presented. LISA is a flexible and portable program which has been developed to analyse structural properties of large scale linear and bilinear structured systems. More precisely, the program LISA contains programmed algorithms which allow us to study the most recent results in the analysis of the structured systems.

Keywords: Structured systems, graph theory, observability, isolability.

1. INTRODUCTION

Structured systems have received much attention since the beginning of the 70's. Based on the work of (Lin 1974), where graphic conditions for the structural controllability are given, (Reinschke 1988) and (Murota 1987) proposes theoretic algorithms for studying the structural properties of multi-input linear systems, like controllability, observability. The main results concerning graph theoretic approach are summarized in (Dion et al. 2003).

Recently, some algorithms have been proposed for the analysis of structured linear systems. In (Hovelaque et al. 1996), for example, the primal-dual algorithm is proposed to derive the infinite structure of a structured system. Later, the authors have implemented the algorithm to analyse the solvability of disturbance decoupling problem (Hovelaque et al. 1997).

In this context (Blanke et Lorentzen 2006) present a Matlab toolbox called SaTool, in which are implemented some results concerning structural analysis theory like reachability, controllability and fault detectability. SaTool uses mainly the bipartite graph to represent structured systems.

In this article, a new tool for the analysis of structured linear and bilinear systems is presented. This tool is based on the representation of structured systems by directed graphs or digraphs. In fact, it is possible to transform graphic conditions given in terms of digraphs into flow graph conditions. Implementation of flow graphs is relatively easy and mainly efficient (lower computational burden) because there already exist many optimized algorithms useful to analyse structured system properties. The first version of LISA program deals with some new results concerning generic unknown input and state observability and fault isolability of structured linear systems and observability of structured bilinear systems. In order to deal with such complex problems, basic have been developed, this basic tools can be used to solve many other problems.

The paper is organized as follows: in section 2, structured linear and bilinear systems are presented, as well as their graphical representation. In section 3, the LISA program is presented, different useful algorithms are summarized and explained. Estimation of their complexity orders is given. Finally, in section 4, we give the conclusion of the paper.

2. STRUCTURED LINEAR AND BILINEAR SYSTEMS

Before presenting LISA program, an introduction to structured systems and their graphical representation is exposed in the following section.

2.1 Structured linear systems

Let us consider structured linear system noted (Σ_{Λ}^l) :

$$(\Sigma_{\Lambda}^l) : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + E_1w(t) + F_1f(t) \\ y(t) = Cx(t) + Du(t) + E_2w(t) + F_2f(t) \end{cases} \quad (1)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $w(t) \in \mathbb{R}^r$, $f(t) \in \mathbb{R}^q$ and $y(t) \in \mathbb{R}^p$ are respectively the state, control input, disturbance, fault and the measured output vector and $A, B, C, D, E_1, E_2, F_1$ and F_2 are constant structured matrices of appropriate dimensions and each of their elements is either fixed to zero or a free non-zero parameter.

A structured linear system (Σ_{Λ}^l) can be represented through digraph $\mathcal{G}(\Sigma_{\Lambda}^l)$. The later is constituted by a vertex set \mathcal{V} and an edge set \mathcal{E} i.e. $\mathcal{G}(\Sigma_{\Lambda}^l) = (\mathcal{V}, \mathcal{E})$. The total number of vertices are denoted by N and the total number of edges by M . The vertices are associated to the state \mathcal{X} , controlled input \mathcal{U} , disturbance \mathcal{W} , measured output \mathcal{Y} and fault \mathcal{F} of (Σ_{Λ}^l) and the edges represent links between these variables. More precisely, $\mathcal{V} = \mathcal{X} \cup \mathcal{U} \cup \mathcal{W} \cup \mathcal{F} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, \dots, x_n\}$ is the set of state vertices, $\mathcal{U} = \{u_1, \dots, u_m\}$ is the set of control input vertices, $\mathcal{W} = \{w_1, \dots, w_r\}$ is the set of disturbance vertices, $\mathcal{F} = \{f_1, \dots, f_q\}$ is the set of fault vertices and $\mathcal{Y} = \{y_1, \dots, y_p\}$ is the set of measured output vertices. Hence, \mathcal{V} consists of $n + m + r + q + p$ vertices.

The edge set is $\mathcal{E} = A\text{-edges} \cup B\text{-edges} \cup C\text{-edges} \cup D\text{-edges} \cup E_1\text{-edges} \cup E_2\text{-edges} \cup F_1\text{-edges} \cup F_2\text{-edges}$, where

$$\begin{aligned} A\text{-edges} &= \{(x_j, x_i) \mid A(i, j) \neq 0\}, \\ B\text{-edges} &= \{(u_j, x_i) \mid B(i, j) \neq 0\}, \\ C\text{-edges} &= \{(x_j, y_i) \mid C(i, j) \neq 0\}, \\ D\text{-edges} &= \{(u_j, y_i) \mid D(i, j) \neq 0\}, \\ E_1\text{-edges} &= \{(w_j, x_i) \mid E_1(i, j) \neq 0\}, \\ E_2\text{-edges} &= \{(w_j, y_i) \mid E_2(i, j) \neq 0\}, \\ F_1\text{-edges} &= \{(f_j, x_i) \mid F_1(i, j) \neq 0\}, \\ F_2\text{-edges} &= \{(f_j, y_i) \mid F_2(i, j) \neq 0\}. \end{aligned}$$

Here $M^\lambda(i, j)$ is the (i, j) th element of matrix M^λ and (v_1, v_2) denotes a directed edge from vertex $v_1 \in \mathcal{V}$ to vertex $v_2 \in \mathcal{V}$.

Example 1. In Figure 1 is represented the digraph associated to the following structured linear system:

$$A = \begin{pmatrix} \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & \lambda_4 & \lambda_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_6 & 0 \\ 0 & 0 & 0 & 0 & \lambda_7 & 0 & \lambda_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_9 \\ 0 & 0 & 0 & 0 & \lambda_{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \lambda_{11} \\ 0 \end{pmatrix},$$

$$F_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \lambda_{12} \end{pmatrix} \text{ and } C = \begin{pmatrix} \lambda_{13} & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_{14} & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{15} & 0 & 0 & 0 \end{pmatrix}.$$

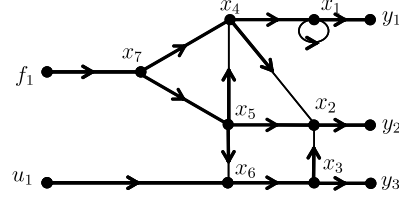


Fig. 1. Digraph for example 1

2.2 Structured bilinear systems

In this part, we consider structured bilinear system Σ_{Λ}^b of the form:

$$\begin{aligned} \dot{x}(t) &= A_0x(t) + \sum_{i=1}^m u_i(t)A_ix(t) + Bu(t) + E_1w(t) + F_1f(t) \\ y(t) &= C_0x(t) + Du(t) + E_2w(t) + F_2f(t) \end{aligned} \quad (2)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $w(t) \in \mathbb{R}^r$, $f(t) \in \mathbb{R}^q$ and $y(t) \in \mathbb{R}^p$ are respectively the state, control input, disturbance, fault and the measured output vectors. For $i = 0, \dots, m$, $A_i, B, C, D, E_1, E_2, F_1$ and F_2 are matrices of appropriated dimensions.

As in the linear case, the digraph associated to (Σ_{Λ}^b) is noted $\mathcal{G}(\Sigma_{\Lambda}^b)$ and is constituted by a vertex set \mathcal{V} and an edge set \mathcal{E} i.e. $\mathcal{G}(\Sigma_{\Lambda}^b) = (\mathcal{V}, \mathcal{E})$. Vertex set is defined identically as in the linear case. Edge set is $\mathcal{E} = \bigcup_{i=0}^m A_i\text{-edges} \cup B\text{-edges} \cup C\text{-edges} \cup D\text{-edges} \cup E_1\text{-edges} \cup E_2\text{-edges} \cup F_1\text{-edges} \cup F_2\text{-edges}$. Note that, we indicate the number i under each A_i -edge in order to preserve the information about the belonging of the edges in the digraph representation. Moreover, we define the following vertex edge subset $\mathcal{X}' = \{x'_{i,j}\}$ where $0 \leq i \leq n$ and $1 \leq j \leq m$ and the edge sets $A'_i\text{-edges} = \{(x_i, x'_{j,k}) \mid A_i(k, i) \neq 0\}$

Example 2. In Figure 2 is represented the digraph associated to the following structured bilinear system:

$$\begin{aligned} A_0 &= \begin{pmatrix} \lambda_1 & \lambda_2 & \lambda_3 & 0 & 0 \\ \lambda_4 & 0 & \lambda_5 & 0 & 0 \\ 0 & 0 & \lambda_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_7 \\ 0 & 0 & 0 & \lambda_8 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} 0 & \lambda_9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{11} & \lambda_{12} & 0 \end{pmatrix}, \\ A_2 &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{13} & 0 \\ 0 & \lambda_{14} & 0 & 0 & 0 \end{pmatrix} \text{ and } C = \begin{pmatrix} \lambda_{15} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{16} & 0 \end{pmatrix}. \end{aligned}$$

Theoretic properties of the realization (Σ_{Λ}^l) or (Σ_{Λ}^b) can be studied according to the values of λ_i .

We say that a property is true generically (Van

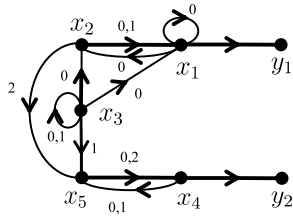


Fig. 2. Digraph for example 2

der Woude 1999) if it is true for almost all the realizations of structured system (Σ_A) .

According to the graphic representation of structured linear system by means of digraph, we will present in the next section the different tools and functions which are programmed in LISA.

3. LISA PROGRAM

LISA was developed in C++, a high level programming language, in order to reduce the computational burden. For the graphical user interface (gui), the library QT 4.0 was chosen. All these features make for LISA a flexible and portable program which is suitable for Windows as well as Linux environment.

Roughly speaking, the program is divided into two parts: the user interface and the storages and calculation classes (*Graph*, *StateRecorder*, *MatrixExporter*). For the purpose of this paper, we will concentrate in only two classes: *Graph* and *MatrixExporter*. A very practical group of functions are programmed in *MatrixExporter* class. This class permits to the user the exportation of graph into a state space representation, which can be read and used by Maple (version 10).

Most of the algorithms programmed in LISA are based on the flow graph approach. Digraph representation is suitable for the visual interpretation of some theoretical conditions which have to be verified by the system to ensure its observability, controllability, etc. However, it is important to note that from a computational point of view, flow graph representation is more suitable than digraph representation. In LISA a translation algorithm is used to convert digraphs into flow graphs and/or bipartite graphs. *FlowGraph* class contains different routines useful to these transformations. Namely, several algorithms are dedicated to find the maximum flow, solve the minimum cost - maximum flow problem and find essential vertices in a flow graph.

In this part, we detail some of the most important algorithms programmed in LISA. For this presentation, algorithms are divided into two main classes: basic graph properties (disjoint paths, essential vertices, etc) and structured system properties (input and state observability and fault isolability). The complexity order of every algorithm is provided in function of the total number of vertices N and the total number of edges M .

3.1 Algorithms for basic graph properties

Successors (predecessors)

This algorithm returns the successor (predecessor) vertices of a given vertex subset. Note that, if directed edge (v_1, v_2) belongs to \mathcal{E} then, v_1 is a predecessor of v_2 and v_2 is a successor of v_1 . The algorithm used to accomplish this task is directly derived from the Boost Library (Boost C++ Libraries 2005).

Finding predecessors in a graph is equivalent to determine successors in the reversed graph. According to the documentation, the complexity of the used algorithm is $O(M)$.

disjoint paths

This algorithm returns the maximal number of disjoint paths between two selected vertex subsets. To cope with this problem, the digraph has to be transformed into a flow graph. This idea is inspired by the next two theorems (Jensen 2002):

- ▷ **Menger's theorem** : the maximum number of edge disjoint paths in a graph is equal to the size of the minimum edge cut in that graph, and
- ▷ **max flow min cut theorem** : the size of the minimum edge cut is equal to the value of the maximum flow in a network where all edges have unit capacity.

Then, the maximum flow is equal to the maximum number of edges disjoint paths in a graph. However, vertex disjoint paths instead of edge disjoint paths are searched. A transformation of a digraph into a flow graph, allows us to convert the min-vertex cut problem to the min-edge cut problem. Transformation of the original digraph \mathcal{G} carries out in two steps: vertex splitting and residual network procedure. Both steps are taking by routines in the *FlowGraph* class. Once the transformation is carried out, that is, a residual network is created, an algorithm to solve the minimum cost - maximum flow problem is implemented. This consists essentially in the Ford Fulkerson algorithm (Bang-Jensen 2002) with a shortest path search at each step. When all the paths are tested, the algorithm stops and the maximum flow is given. There can be a maximum of $O(N)$ disjoint paths (=path searches), the complexity order of each search is $O(N\sqrt{M})$. Hence the overall complexity order is $O(N^2\sqrt{M})$.

essential vertices

This algorithm calculates the essential vertices in disjoint paths between two selected vertex subsets (source and sink). Essential vertices are all the vertices which are included in every set of maximal disjoint paths or maximum linking (Van der Woude 1999). The removal of an essential vertex causes a reduction in the number of disjoint paths. To determine the essential vertices in a digraph, each vertex v is removed and the maximum flow between source and sink is calculated. If the new flow is smaller than

the original flow, then \mathbf{v} is essential. The algorithm requires $O(N)$ flow calculations. As we have seen, the complexity order of every flow calculation is $O(N^2\sqrt{M})$. Then, the overall complexity order is $O(N^3\sqrt{M})$.



separators

This algorithm returns input and output separators. A separator is a vertex subset which contains at least one vertex in every path of a linking. There are at least two vertices belonging to a separator subset in every disjoint path (Murota 1987). The set of separators can be found as follows: First the maximal set of disjoint paths is calculated. Next, the set of essential vertices is calculated. Hence, to find the input (output) separator it is sufficient to take the essential vertex closer to the source (sink) subset in each disjoint path. Essential vertices are found in $O(N^3\sqrt{M})$. Finding the vertices on each path which are also essential vertices is done in $O(N \log_2(N))$, hence the overall asymptotic complexity order is $O(N^3\sqrt{M})$.



maximal matching:

The algorithm determines the size of the maximal matching between two selected vertex subsets (source and sink). A matching is a set of disjoint edges in a bipartite graph. For this algorithm, the equivalence between the maximum flow and the maximal matching in a bipartite graph is considered (Bang-Jensen 2002, Murota 1987). To this aim, an algorithm dedicated to the construction of a bipartite flow graph from a directed graph is used. Once the bipartite flow graph is built, the maximum flow algorithm is run over this bipartite flow graph. The conversion of digraph into bipartite flow graph has a complexity equal to $O(M + N)$ for linear case and $O(M + MN)$ for bilinear case. The complexity order of the computation of the maximum flow is $O(N^2\sqrt{M})$, the overall complexity order is $O(N^2\sqrt{M})$.

3.2 Algorithms for properties of structured system

Generic properties of structured systems related control problems are considered in section 3.1.

Based on the works of (Boukhobza et al. 2006), we recall in a first time the graphic conditions for the generic input and state observability of structured linear systems. Next, according to the works of (Boukhobza and Hamelin 2006a), observability of structured bilinear systems is treated. Finally, FPRG (Fundamental Problem of Residual Generation) of structured linear systems are introduced. We will briefly discuss algorithms for checking these properties.

3.2.1. Input and state observability of structured linear systems In this paragraph, the conditions for the input and state observability of structured linear systems are given and they are translated to an algorithm-based language.

Input and state observability of structured linear system Σ_{Λ}^l requires the definition of the following subsets:


- \mathcal{X}_0 is defined as all vertices \mathbf{x}_i of \mathcal{X} such that the number of disjoint paths from $\{\mathbf{x}_i \cup W \cup F$ to \mathcal{Y} and $W \cup F$ to \mathcal{Y} are equal. According to this, \mathcal{X}_0 is determined with function **disjoint paths** presented in section 3.1.
- \mathcal{X}_s contains all vertices of type \mathcal{X} of all output separators between $\mathcal{F} \cup \mathcal{W}$ and \mathcal{Y} calculated with function **separators**.
- Ω_0 contains all vertices \mathbf{v}_i of $\mathcal{F} \cup \mathcal{W}$ such that the maximal matching from \mathbf{v}_i to $\mathcal{X} \setminus (\mathcal{X}_s \cup \mathcal{X}_0)$ equals zero. This subset is defined by means of the maximal matching function of 3.1.
- \mathcal{Y}_0 consists in all essential vertices between $\mathcal{W} \cup \mathcal{F} \cup \mathcal{Y}$ in \mathcal{Y} . \mathcal{Y}_0 is then determined with **essential vertices function** of 3.1.

According to these definitions, input and state observability of structured linear systems can be verified if next three conditions are satisfied (Boukhobza et al. 2006):

Cond1: each vertex in $\mathcal{X} \cup \mathcal{F} \cup \mathcal{W}$ is predecessor of vertex set \mathcal{Y} .

Cond2: all vertices in \mathcal{X}_0 and Ω_0 are essential for the linkings between Ω_0 and $\mathcal{X}_s \cup \mathcal{Y}_0$.

Cond3: the maximal matching in the bipartite graph $\mathcal{X} \cup \mathcal{F} \cup \mathcal{W} \rightarrow \mathcal{X} \cup \mathcal{Y}$ is equal to $n + q + r$.

Command  **Observable** checks conditions **Cond1**, **Cond2** and **Cond3** as follows:

To verify condition **Cond1** algorithm **predecessors** is used and then all vertices of \mathcal{X} , \mathcal{F} and \mathcal{W} are searched in the result list of predecessors of \mathcal{Y}_0 .

According to the complexity orders associated to the basic graph functions, it results that the complexity order on calculating **Cond1** is $O(M + N \log_2(N))$.

The main steps for the verification of **Cond2** are recapitulated as follows:

- (1) Essential vertices between Ω_0 and $\mathcal{X}_s \cup \mathcal{Y}_0$ are calculated,
- (2) Finally, if all vertices in \mathcal{X}_s and Ω_0 are in the previously calculated set, then **Cond2** is satisfied.

According to the complexity orders associated to the basic graph functions, it results that the complexity order on calculating **Cond2** is $O(N^3\sqrt{M})$.

To check condition **Cond3** it is sufficient to compute the maximal matching between $\mathcal{X} \cup \mathcal{F} \cup \mathcal{W}$ and $\mathcal{X} \cup \mathcal{Y}$. That is, the complexity order of **Cond3** is $O(N^2\sqrt{M})$.

3.2.2. *Observability for structured bilinear systems*
 Graphic conditions for the observability of structured bilinear systems can be found in (Boukhobza and Hamelin 2006a). We can summarize these conditions as follows:

CondA: Each vertex in \mathcal{X} is a predecessor of vertex set \mathcal{Y} , and

CondB: There exists a maximal matching in the bipartite graph $\mathcal{X} \rightarrow \mathcal{Y}' \cup \mathcal{X}'$

For condition **CondA**, function **predecessors** implemented in section 3.1 can be used.


To check condition **CondB** it is sufficient to compute the maximal matching between \mathcal{X} and $\mathcal{X}' \cup \mathcal{Y}$. Thus, the complexity order of **CondB** is $O(NM^{3/2})$.

3.2.3. *Fault Isolability of structured linear systems*

An important property in diagnosis context is the fault isolability. LISA allows us to verify this property for the multifault case as well as for the monofault case. Conditions of The FPRG solvability are treated in detail in (Commault et al. 1999, Commault et al. 2002).

Only one condition has to be verified for ensuring fault isolability:

CondI: a subset of faults \mathcal{F}_0 belonging to \mathcal{F} set, is isolable if the number of disjoint paths between $\mathcal{F} \cup \mathcal{W}$ and \mathcal{Y} is equal to the number of disjoint paths between $\mathcal{F} \setminus (\mathcal{F}_0 \cup \mathcal{W})$ and \mathcal{Y} plus $\text{card}\{\mathcal{F}_0\}$.

Command  **Isolability** check condition **CondI** using the function **disjoint paths** presented in section 3.1. Thus, the overall complexity order is $O(N^2\sqrt{M})$.

In this section, an exhaustive description of the algorithms developed in LISA was done. Structural properties of structured systems as observability and fault isolability are calculated by this application. In order to check these properties, basic algorithms have been programmed. As it is known, some other structural properties can be studied using these graphic algorithms (Van der Woude and Murota 1995, Van der Woude 1999, Dion et al. 2003). So, the LISA program is intended to expand its capabilities to verify other structural properties as controllability, disturbance rejection, etc.

4. CONCLUSION AND PERSPECTIVES

This paper has described. LISA is a program to display and manipulate linear/bilinear control systems. The systems are displayed in terms of graphs, which can be manipulated by the user. Concepts such as observability, fault isolability, etc. have been translated into graph theoretic terms, and were implemented with different graph algorithms.

Since many of the problems that needed to be solve here are rather expensive in terms of time complexity (i.e. finding disjoint paths) it was crucial to use a programming language with little overhead, close to the machine code level, C++ was chosen to benefit from a much grater flexibility (e.g. classes, templates), with only little sacrifice of performance. For the graphical user interface (gui) the library QT 4.0 was chosen, a very flexible, portable and easy to use library for creating graphical user interfaces.

REFERENCES

- Bang-Jensen, J. and G. Gutin (2002). *DIGRAPHS Theory, Algorithms and Application*, Springer-Verlag, London, England.
- Blanke, M. and T. Lorentzen (2006). 'SaTool - A software Tool for Structural Analysis of Complex Automation Systems', In: *6th IFAC SAFEPROCESS' 2006*, Beijing, P. R. China.
- Boost C++ Libraries (2005): Version 1.0 'http://www.boost.org'.
- Boukhobza T. and F. Hamelin (2006a). 'Observability analysis for structured bilinear systems: a graph theoretic approach', *Automatica*, provisionally accepted.
- Boukhobza, T., F. Hamelin, and S. Martinez-Martinez (2006). 'State and input observability analysis for structured linear systems: a graph theoretic approach', *Automatica*, provisionally accepted.
- Commault, C., J.M. Dion, O. Sename, and J.C. Avila Vilchis (1999). 'Fault detection and isolation: a graph approach', *Proc. of the European Control Conference*, Karlsruhe, Germany.
- Commault, C., J.M. Dion, O. Sename, and R. Motyeian (2002). 'Observed-based fault detection and isolation for structured systems', *IEEE Transactions on Automatic Control*, AC-47, 2074-2079.
- Dion, J.-M., C. Commault and J. Van der Woude (2003). 'Generic properties and control of linear structured systems: A survey', *Automatica*, 39(7), 1125-1144.
- Hovelaque, V., C. Commault and J.M. Dion (1996). 'Analysis of linear structured systems using a primal-dual algorithm', *Systems and Control Letters*, 27, 73-85.
- Hovelaque, V., N. Djidi, C. Commault and J.M. Dion (1997). 'Decoupling Problem for Structured Systems Via a Primal Dual Algorithm', *4th Europea Cont. Conf.*, Brussel, Belgium.
- Lin, C. T. (1974). 'Structural controllability', *IEEE Transactions on Automatic Control*, AC-19(3), 201-208.
- Murota, K. (1987). *Systems Analysis by Graphs and Matroids, Structural, Stability and Controllability*, Springer-Verlag, New York, U.S.A.
- Reinschke, K. J. (1988). *Multivariable Control. A Graph-Theoretic Approach.*, Springer-Verlag, New York, U.S.A.

- Shields, R. W. and J. B. Pearson (1976). 'Structural controllability of multi-input linear systems', *IEEE Transactions on Automatic Control*, **AC-21**(2), 203–212.
- Van der Woude, J. W. (1999). 'The generic number of invariant zeros of a structured linear system', *SIAM J. Control Optim.*, **38**(1), 1–21.
- Van der Woude, J. and K. Murota (1995). 'Disturbance decoupling with pole placement for structured systems: A graph-theoretic approach', *SIAM Journal on Matrix Analysis and Applications*, **16**(3):922–942.