Prototype Software for Automated Structural Analysis of Systems

Jørgensen, A.; Izadi-Zamanabadi, Roozbeh; Kristensen, M.

Publication date:
2004

Document Version
Også kaldet Forlagets PDF

PROTOTYPE SOFTWARE FOR AUTOMATED STRUCTURAL ANALYSIS OF SYSTEMS

Anders Jørgensen *
Roozbeh Izadi-Zamanabadi**, **Michael Kristensen** **

* *Aalborg University, Center for Embedded Systems,*
Fr. Bajers Vej 7B, DK-9220 Aalborg Ø, Denmark
e-mail: gauss@cs.auc.dk
** *Aalborg University, Department of Control Engineering,*
Fr. Bajers Vej 7C, DK-9220 Aalborg Ø, Denmark,
e-mail: {riz,mek77}@control.auc.dk

Abstract: In this paper we present a prototype software tool that is developed to analyse the structural model of automated systems in order to identify redundant information that is hence utilized for Fault detection and Isolation (FDI) purposes. The dedicated algorithms in this software tool use a tri-partite graph that represents the structural model of the system. A component-based approach has been used to address issues such as system complexity and reconfigurability possibilities.

1. INTRODUCTION

Industrial systems usually large number of include a large number of actuators, sensors, and parts. Faults in such systems, if allowed to persist and propagate, can evolve into system failure with grave consequences (economical/ environmental/ safety) as a result.

The field of Fault diagnosis/monitoring has been subject to intense research in last two decades. However, the main focus has been on the dynamic systems (linear/nonlinear) with limited number of involved components. With increasing number of components, the task of modelling and analysis of such systems, with the purpose of employing fault diagnosis, will become an overwhelming task and will require an extensive and very time consuming effort. In this regard, being able to automatically analyse the system will be of great advantage and cost-effective.

The quest for such ability has inspired us to develop a prototype software tool for automatic analysis of the available information within a system. This design tool is based on a component approach, where the information is placed in the components (controllers, actuators, sensors and plants) and the components are connected together. The component approach makes it easy to build models of large and complex systems. It is also easy to reuse and replace components from one model to others. The algorithms for finding FDI possibilities is based on structural analysis (Izadi-Zamanabadi and Blanke, 2002).

The paper presents a software prototype tool for analysis of dynamic systems in order to identify systems' inherent redundant information. The information then can be used to develop dedicated algorithms for fault diagnosis/monitoring purposes. Section 2 provides a short description of the theory of structural analysis and the definitions. The design of the prototype software tool is found in section 3. An example of how to use the design tool is given in section 4.

2. SYSTEM'S STRUCTURAL MODEL

For a given system S, there exist a set of relations, which each can be static, dynamic, linear or nonlinear relations. In a structural model these relations are represented by constraints. Constraints

contain information about which variables are involved in a relation and whether a variable can be calculated from the relation or not. The set of constraints are represented by $\mathscr{F} = \{f_1, f_2, ..., f_m\}$ and all the system's variables are represented by $\mathscr{Z} = \{z_1, z_2, ..., z_n\}$. $\mathscr{Z} = \mathscr{K} \cup \mathscr{X}$, where $\mathscr{X}$ is the set of unknown variables in the system, while $\mathscr{K}$ is the set of known variables in the system. $\mathscr{K}$ can be divided into the subcategories $\mathscr{U}$ and $\mathscr{Y}$ ($\mathscr{K} = \mathscr{U} \cup \mathscr{P} \cup \mathscr{Y}$). Where $\mathscr{U}$ is the known controller outputs and $\mathscr{Y}$ is the measured signals.

The structural model can be represented by a dedicated tri-partite directed graph. The nodes (vertices) in this graph can be divided in the three categories '**K**', '**F**' and '**X**', which have the the following properties:

(1) No edge (arc) can exist between two '**K**', two '**F**' or two '**X**'nodes.
(2) No edge can exist between any '**K**' and '**X**' nodes.

Definition 1. (Izadi-Zamanabadi and Blanke, 2002) The structure graph of the system is a tri-partite directed graph $\mathtt{G}(\mathscr{F}, \mathscr{Z}, \mathscr{A})$ are defined by the following mappings:

$$\begin{cases} \mathbf{A} & : \mathscr{F} \times \mathscr{X} \longrightarrow \{0,1\}, \\ \mathbf{A}^* & : \mathscr{X} \times \mathscr{F} \longrightarrow \{0,1\}, \\ \mathbf{KF} & : \mathscr{F} \times \mathscr{K} \longrightarrow \{0,1\}. \end{cases}$$

Where the elements $a_{ij} \in \mathbf{A}, a^*_{ij} \in \mathbf{A}^*$ and $kf_{ij} \in \mathbf{KF}$ are defined as

$$a_{ij} = (f_i, x_j) = \begin{cases} 1 \text{ iff } f_i \text{ applies to } x_j, \\ 0 \text{ Otherwise} \end{cases}$$

$$a^*_{ij} = (x_i, f_j) = \begin{cases} 1 \text{ iff } x_i \text{ is calculable through } f_j, \\ 0 \text{ Otherwise} \end{cases}$$

$$kf_{ij} = (f_i, k_j) = \begin{cases} 1 \text{ iff } f_i \text{ applies to the} \\ \quad \text{known variable } k_j, \\ 0 \text{ Otherwise} \end{cases}$$

An element $a_{ij} = 1$ means that there is a directed edge, that connects the i^{th} relation with the j^{th} unknown variable; $a_{ij} = 0$ means that there is no edge.

The tri-partite graph can be represented by an incidence matrix. The incidence matrix is defined as in equation 1.

$$I = \begin{array}{c} \\ \mathscr{K} \\ \mathscr{F} \\ \mathscr{X} \end{array} \begin{array}{c} \begin{array}{ccc} \mathscr{K} & \mathscr{F} & \mathscr{X} \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{KF} & \mathbf{0} \\ \mathbf{KF}^T & \mathbf{0} & \mathbf{A} \\ \mathbf{0} & \mathbf{A}^* & \mathbf{0} \end{bmatrix} \end{array} \tag{1}$$

The calculability property, which is used in the $\mathbf{A}^*$ is defined as in definition 2.

Definition 2. (Calculability).
(Izadi-Zamanabadi, 1999) Let $z_j, j = 1, ..., p, ..., n$ be variables that are related through a relation f_i e.g. $f_i(z_1, ..., z_p, ..., z_n) = 0$. The variable z_p is calculable if its value can be determined through the constraint f_i under the condition that the values of the other variables $z_j = 1, ..., n, j \neq p$ are known.

It is therefore not always given, that a variable can be calculated through a constraint. This is illustrated in example 1.

Example 1. A system has the following relation with saturation:

$$f_1 = \begin{cases} x_2 = \alpha x_1 \text{ for } -k \leq x_1 \leq k \\ x_2 = a \text{ for } x_1 > k \\ x_2 = -a \text{ for } x_1 < -k \end{cases}$$

For a given value of x_1 it is always possible to uniquely determine the value of x_2 but it is not always possible to uniquely determine the value for x_1 for a given x_2, for instance, choose $x_2 = a$, then multiple choices exist for x_1.

□

2.1 *Matching and redundancy relations*

Matching is a tool for finding the redundancy relations in a system. The matching concept is in general terms a way for finding which and how unknown variables can be computed from known variables.

Definition 3. (Declerck and Staroswiecki, 1991) The (sub)graph $\mathtt{G}(F_M, X_M, A_M)$ is a matching on $\mathtt{G}(F, X, A)$ [1], $F_M \subseteq F$ and $X_M \subseteq X$ iff:

(1) $A_M \subseteq A$,
(2) $\forall\, a1, a2 \in A_M$ [2] $\mid a1 \neq a2 \Leftrightarrow F_M(a1) \neq F_M(a2) \wedge X_M(a1) \neq X_M(a2)$.

A complete matching w.r.t. F is obtained when $F_M = F$.
A complete matching w.r.t. X is obtained when $X_M = X$.

When a matching is made upon a system, a system can be decomposed into three kinds of subsystems:

- over-determined subsystem, denoted by $\mathtt{G}^+(F^+, X^+, A^+)$,
- just-determined subsystem, denoted by $\mathtt{G}^=(F^=, X^=, A^=)$, and
- under-determined subsystem, denoted by $\mathtt{G}^-(F^-, X^-, A^-)$.

[1] $F \subseteq \mathscr{F}$, $X \subseteq \mathscr{X}$, $A \subset \mathscr{A}$
[2] a_i is a edge between a constraint $f_p = F(a_i)$ and a variable $x_q = X(a_i)$ ($F(a_i) \overset{a_i}{\leftrightarrow} X(a_i)$)

The over-determined subsystem is of interest for fault diagnosis purposes because it contains the redundant relations. The three subsystems are graphically illustrated in figure 1. The grey area in

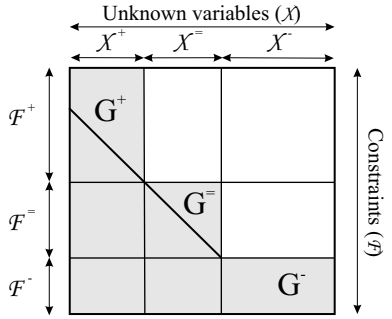

Figure 1. Decomposition of a system into over-($\mathtt{G}^+$), just-($\mathtt{G}^=$) and under-determined subsystems ($\mathtt{G}^-$)

figure 1 contains both one and zeroes, while the white area containing only zeroes meaning that there is no link the variables and constraints. In $\mathtt{G}^+$ there is more constraints that variables. In $\mathtt{G}^=$ there is the same numbers of variables as numbers of constraints. While $\mathtt{G}^-$ has more variables than constraints. The thick line illustrate the matching, where the matched pairs of constraints and variables are placed on the line.

The principles for the matching procedure can be explained though figure 2. The figure shows that

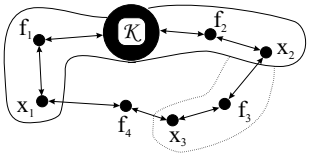

Figure 2. Matching of constraints and variables

the matching starts with the set of known variables ($\mathscr{K}$). It then finds all the constraints that contain only one unknown variable and match the unknown variables to the known variables. In the case of figure 2 f_1 only contains the unknown variable x_1, and f_2 contains only the unknown variable x_2. x_1 and x_2 are now added to the set of known variables ($\mathscr{K} = \mathscr{K} \cup \{x_1\} \cup \{x_2\}$). The procedure repeat the matching until there are no more unknown variables that can be matched. As it is illustrated in the figure, the first matchings has made it possible to match x_3 and the matching procedure has then met its stopping criteria. The matching algorithm is described in more details in next section.

2.1.1. *Loop problems* For some systems there exist loops in the structural model, where two or more variables have to be determined simultaneously. These loops can be divided into two kinds of groups: algebraic loops and differential-algebraic loops ((Blanke *et al.*, 2003), Ch. 5). An example of a loop can be seen graphically in figure 3. It can be seen in figure 3 that both

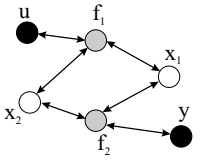

Figure 3. An example of an algebraic loop.

x_1 and x_2 have to be determined simultaneously. Matching can be performed on a loop by assuming that one unknown variable as known, and then match the other unknown variables. When the other variables are matched, the variable which was assumed to be known is set to unknown, and it will then be matched. The main condition, which has to be fulfilled, is that number of constraints may not be smaller than the number of variables.

2.1.2. *Quality of matching* From graph theoretical point of view, redundant relations exist when there are at least two pathes from the known variables to an unknown variable. Another interesting issue that is relevant for sensor fusion is to determine (i.e. compute/estimate) the value of an unknown variable by using the constraints on the path on the graph that connects the known variables to the unknown variable. The constraints represent information with different quality level, i.e. they could represent linear or nonlinear equations (involving some disturbances or uncertainties), or table of data. Hence the quality of the computed value of an unknown variable may vary depending on the quality level of the involved constraints on each path. So, for sensor fusion (as well as residual generation) purposes, it is important to find the best solutions, i.e. pathes with constraints of highest qualities.

To find the solution a quality factor is been introduced: 4.

Definition 4. The quality of an unknown variable can be calculated though the following product:
$\eta_{x_p} = \eta_{f_q} \prod \eta_{x_i}$, where f_q is used to match x_p, and x_i is all unknown variables $x_i \neq x_p$ which are involved in relation f_q. $x_i \in \mathscr{X}$.
$\eta_{k_i} = 1$, $k_i \in \mathscr{K}$
$0 < \eta_{f_j} \leq 1$, $\eta_{f_j} \in \mathbb{R}$.
$\eta_{f_j} = 1$ means that the relation is pure deterministic and without any kind of disturbance.

The quality of a residual can be calculated as following:
$\eta_{r_p} = \eta_{f_r} \prod \eta_{x_j}$, where f_p is an unmatched relation, $f_r \in F^+$ which is used as residual, and x_j is all variables involved in f_r.

The calculating of the qualities can be illustrated in the following example.

Example 2. A matching of a given system be done as in figure 4.

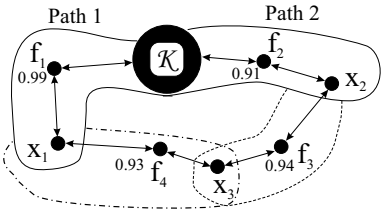

Figure 4. Matching procedure with quality factor introduced

The matching starts by matching x_1 and x_2 though f_1 and f_2. The quality for x_1 is $\eta_{x_1} = \eta_{f_1} = 0.99$ and $\eta_{x_2} = \eta_{f_2} = 0.91$. Now x_3 can be matched by two path (path 1 and path 2). Path 1 is then chosen. $\eta_{x_3} = \eta_{f_4} \cdot \eta_{x_1} = 0.92$. f_3 can now be used as a residual, which give the quality for the residual: $r_1 = \eta_{f_3} \cdot x_3 \cdot x_4 = 0.84$ □

2.1.3. *Minimal over-determined subsystems* The minimal over-determined subsystems represent the smallest subsystems that have redundant information.

Definition 5. (Izadi-Zamanabadi and Blanke, 2002) A minimal over-determined subsystem, $G^+_{min-f} = (F^+_{min-f}, X^+_{min-f}, A^+_{min-f})$ is the smallest over-determined (hence observable) subsystem which is obtained by back-tracking the unknown variables in an unmatched relation f in F^+. For a mainimal over-determined subsystem, the following statement is valid:

$$|F^+_{min-f}| = |X^+_{min-f}| + 1$$

where $|X|$ denoted the number of elements in the set X.

For each minimal over-determined subsystem there exist an analytical redundancy relation (ARR). The analytical redundancy relation is of the form:

$$f(z_i, \ldots, z_j) = 0,\ z_i, \ldots, z_j \in \mathscr{K} \qquad (2)$$

The residual for the same subsystem can then be defined as:

$$r = f(z_i, \ldots, z_j),\ z_i, \ldots, z_j \in \mathscr{K} \qquad (3)$$

3. PROGRAM DESIGN

The main focus for the program is on the development of a software tool, which can perform the structural analysis on a complex system, and it is easy to use. In order to make it easy to define and analyse systems the prototype tool has a graphical user interface (GUI). It is chosen that the tool shall encapsulate the relations in a system in components, such that the overall system view represents the involved physical components and their connections. One of the advantages of component based approach is, that the component only have to be specified once and it can then be reused over and over again. Another advantage the component based approach provides, is the possibility to examine different strategies of placing sensors, e.g. to ensure the cheapest/fewest set of sensors while obtaining highest level of fault-tolerance .

The components are categorised in to four groups of components; *Controller, Actuator, Plant, Sensor.*
The controller and the sensor categories are the only categories, which have known output. These are the only two categories of components, where it is possible to specify known variables. The components are numbered with respect to their categories. This give the following numeration:

- → Controller $co1, co2, \ldots, con$
- → Actuator $ac1, ac2, \ldots, acn$
- → Plant $pl1, pl2, \ldots, pln$
- → Sensor $se1, se2, \ldots, sen$

The components will be automatically numbered by these numbers and the numbers from deleted components will automatically be reused to number new components. By using automatic numeration the program ensures that identical numeration is not taking place.
Each component in the system is symbolized by a box with a number of pins. The relations in a component can be entered by opening a window for the components properties. It had been chosen to make it possible to enter linear differential equations as well entering constraints directly. The program will also automatically add an extra constraint, when a differential equation is entered. This constraint is the relation between $\dot{x}_i$ and x_i. $\dot{x}_i$ is entered in the program as dx_i, where '_' is used to separate the variable (dx, x, y, u) and the index numbers. A none-calculable variable in a constraint is symbolized by entering a '#' like $x\_1\#$. The constraints number are automatically added, when the constraint is entered. The id number of the component is added to the constraint and variable names, when these are shown outside the component. The in- and output for the component is specified by the drag and drop the variables to either input or output. All known variables (y and u) are automatically specified as output.
The input and output from the components can be connected by drawing a line between them, so the output variable from one component becomes

a synonym with the input variable of the other component it is connected to.

3.1 *Algorithm for structural analysis*

When the whole system design has been made (all components are added, specified and connected), the program can make an analysis. The analysis starts with building the system structural model. Thereafter the program performs all possible matchings. A decomposition of the results from the matchings are thereafter made. The program will finally present all the minimal over-determined subsystems which can be used for constructing the ARRs.
The structural model is built by taking all the components in the system. All the input variables in a component are then replaced by the output variables, which they are connected to. All the constraints and variables in a component is then added the component's id as and extension, so all the variables and constraints are unique. The extended incidence matrix (and a compact version of it) are built, and a tri-partite graph is then constructed.
The matching procedure, which has been developed for this program is based on the general matching procedure (Izadi-Zamanabadi and Blanke, 2002). The procedure is modified in order to both calculate qualities and make a total matching and furthermore a level of matching has been introduced in order to better overview the order of performed matchings. The procedure for the program can in general terms be described by the following algorithm:

Algorithm 1. **Initialize:**

- $MatchingNumber = 1$.
- $MatchingLevel = 0$.
- $MatchPair = \{\}$.
- $Matchings = \{\}$.
- $PossibleMatchings = \{\}$.

while(other possible matchings exist)

- **while**(variables are possible to match)
  - Project $\mathscr{K}$ via $\mathscr{F}$ onto $\mathscr{X}$ using the incidence matrix I_{ehn}.
  - Identify unmatched relations in $\mathscr{F}$ that contain exactly one unknown variable and fulfill the calculability condition.
  - calculate the quality product.
  - Register the matched pair(s): $MatchPair = MatchPair \cup (f_j, x_i)$
  - Include the matched unknown variable(s) x_i to the set of known variables: $\mathscr{K} = \mathscr{K} \cup x_i$.
  - Exclude the matched constraints: $\mathscr{F} = \mathscr{F} \setminus f_j$.
  - **if** (match pair, (f_p, x_i) also exist)
    $newPossibleMatching$ = a image of the matching until now.
    Replace (f_j, x_i) with (f_p, x_i).
    Exclude the matched constraints: $\mathscr{F} = \mathscr{F} \setminus f_p$.
    $PossibleMatchings = PossibleMatchings \cup newPossibleMatching$
  - Exclude the matched constraints: $\mathscr{X} = \mathscr{X} \setminus x_i$.
  - $MatchingLevel = MatchingLevel + 1$.
- **end**
- **If** (matching loop exist)
  - Run matching loop algorithm (algorithm **??**).
- **If** ((This matching)$\notin Matchings$)
  - $Matchings = Matchings \cup$(This matching).
  - $MatchingNumber = MatchingNumber + 1$
- This matching = $PossibleMatchings[0]$.
- $PossibleMatchings = PossibleMatchings \setminus$(This matching).

end

The condition, **If** ((This matching)$\notin Matchings$), is introduced in order to avoid identical matchings. The problem occur, when there exist several possible matchings on the same matching levels. The identical matchings are identified and sorted out automatically.
The matching algorithm is extended in order to handle the loop problem , which will not be addressed in detail in this paper. Additionally, the program provides a graphical view of the decomposed subsystems, e.g. over-, just-, and under-determined.
The program provides a list of all different (but non-identical) minimal over-determined subsystems by backtracking the variables and constraints in each over-determined subsystem that has been obtained through matching.

4. A SYSTEM EXAMPLE

Consider the following simple system consisting of a position controller, a DC-motor, and 3 sensors, modelled by

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & \frac{K}{R} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} V \quad (4)$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} \quad (5)$$

where J is the moment of inertia of the rotor, b is the mechanical damping ratio, K is the electromo-

tive force constant, R is the (electric) resistance, L is the inductance and V is the supply voltage. Parameters are obtainable from The motor can be specified in the program by dragging a new actuator (the DC motor) component from the list of components and drop it on the design area. The motor can then be specified by right clicking on the component and entering the relations as seen in figure 5. Where $\theta = x\_1$, $\dot{\theta} = x\_2$,

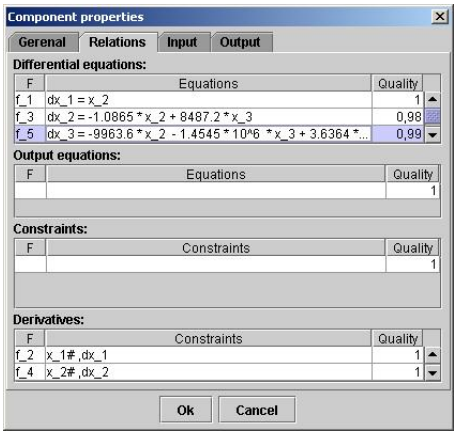

Figure 5. Entering of relations for the motor model.

$i = x\_3$ and $V = x\_4$. x_4 is then set as input and x_1, x_2 and x_3 are set to outputs. Then the sensors can be added to the system, here a tacho-meter, encoder and AM-meter are added. They are modelled to measure the states in the motor directly. A controller is added to specify that V is control signal and is known. The variables are connected by drawing a lines between components. The total system view is now as in figure 6. When the model is set up, then

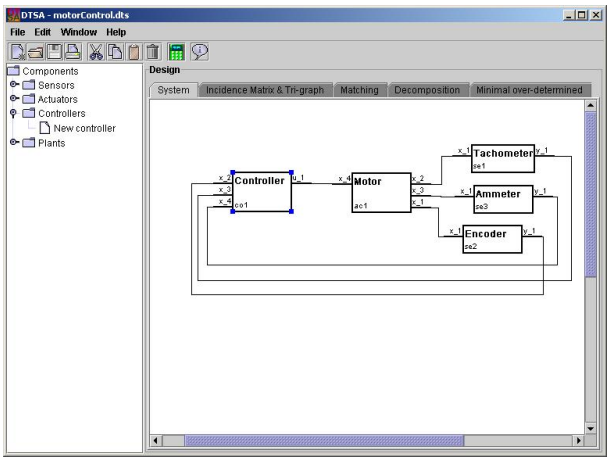

Figure 6. Total model of the system design.

it is analysed automatically and the results will be delivered graphically in different windows. The program finds 8 possible different matchings for the system. Furthermore the program shows that all system parts are over-determined. Finally, the following minimal over-determined subsystems is provided: All duplicated subsystems are deleted

	Unknown variables	Constraints
1-f_3,co1	x_1,co1, dx_1,co1	f_1,co1, f_2,co1, f_3,co1
1-f_2,ac1	dx_1,ac1, x_1,ac1, x_2,ac1	f_1,se2, f_1,ac1, f_2,ac1, f_1,se1
1-f_3,ac1	x_2,ac1, dx_2,ac1, x_3,ac1	f_1,se3, f_3,ac1, f_4,ac1, f_1,se1
1-f_5,ac1	x_2,ac1, x_3,ac1, dx_3,ac1	f_1,se3, f_5,ac1, f_6,ac1, f_1,se1

Table 1. Minimal overdetermined subsystems for the motor position control example.

and only the disjoint minimal over-determined subsystems are listed in table 1. The first row in the table only indicates that all variables in the controller are known, which is obvious. The three remaining rows can lead directly to 3 analytical redundant relations (ARRs) that can be used for fault diagnosis purposes.

5. CONCLUSIONS

The paper presents a software prototype tool for modelling and automated analysis of systems. The main algorithms and functionalities were listed and an example was provided in order to illustrate the programs's functionality. The program enables analysis of large and complex systems that are extremely difficult to handle otherwise. The program is freely available and it is our hope that other research groups employ it on their systems and provide us with feedback in order to deal with possible insufficiencies in the program. The program is obtainable via following URL address:

http://www.control.auc.dk/ftc

Relevant documentation concerning the theoretical background and quick user manuals as well as some examples can be found on the same address.

6. REFERENCES

Blanke, M., Kinneart, M., Lunze, J. and Staroswiecki, M., Eds.) (2003). *Diagnosis and Fault-Tolerant Control*. Springer Verlag. New York.

Declerck, P. and M. Staroswiecki (1991). Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants.

Izadi-Zamanabadi, R. and M. Blanke (2002). Structural analysis for diagnosis with application to ship propulsion problem.

Izadi-Zamanabadi, Roozbeh (1999). *Active Fault-tolerant Supervisory Control - System Analysis and Logic Design.*