About detection substitutions in nonlinear algebraic equations with help of Tarjan's algorithm.

Isakov A.A., Senichenkov Yu.B., Distributed Computing and Networking department, Saint Petersburg state Polytechnical University, senyb@dcn.icc.spbstu.ru, SPBSTU, Polytehnicheskaj 29, St. Petersburg,195251, Russia

Abstract — MvStudium (www.mvstudium.com) is a tool for visual modelling and simulation of complex dynamical systems, including hybrid systems. While model building MvStudium forms and reduces large scale systems of differential-algebraic equations for each state of hybrid automation used for specification of hybrid systems. State equations written by user may contain substitutions. Numerical solution costs for current system are in proportion to a number of equations and their structure. Automated detecting substitutions among equations leads to decreasing size of solving equations. Taking into consideration a structure of built system allows calling most effective Solver. For example if system may be transformed to the system with block-triangular form, Solver should solve systems for diagonal blocks only.

The MvStudium's version of Newton's method for solving systems of nonlinear algebraic equations with detecting substitutions is considered. In this version Tarjan's algorithm of finding the strongly connected components is used for detecting substitutions and transforming of an initial system to block-triangular form.

***Keywords* —** complex dynamical systems, modeling languages, equation-based models, hybrid systems, Tarjan's algorithm, Newton's root-finding method.

I. INTRODUCTION

In this paper «Analyzer» means «module that can detect type and structure system of equations written by User», and «Solver» - «module that can choose most effective numerical method using information about type and structure about solved system getted from Analyzer».

Newton's method is the basic method for solving nonlinear algebraic equations

$$f_i(x_1, x_2, ..., x_n, time) = 0, \mathrm{i} = 1, \mathrm{n} \qquad (1)$$

Isakov A.A. and Senichenkov Yu.B. are with the National Research University «St. Petersburg State Polytechnical University», SPBSTU, Polytehnicheskaj 29, St. Petersburg,195251, Russia, senyb@dcn.icc.spbstu.ru.

used in MvStudium.

The MvStudium's modification of the method takes into account structure of system and can freeze Jacobi matrix if it is efficiently. For detecting matrix structure (band, block-triangular, sparse and so on) Analyzer uses structure matrix $\mathbf{S}$ of system with $s_{ij} \in \{0,1\}$. Non-zero element s_{ij} in matrix $\mathbf{S}$ says that j-th unknown is presented in i-th equation.

Let us consider an example.

Example 1. User wants to solve the system of nonlinear algebraic equation written in the form (1) respect x_i. Constants C_i are known.

$$\begin{cases} 10 \cdot x_3 + x_4 - 10 = 0 \\ x_3 + \sin(x_4) + \cos(x_4) - 10 = 0 \\ x_1 - C_1 = 0 \\ x_2 - \sin(x_1) - C_2 = 0 \\ x_3 + \sin(x_1) - x_1 - x_2 = 0 \\ x_{25} - \sin^2(x_4) - \cos^2(x_4) = 0 \end{cases}$$

It is obvious that among equations there are substitutions.

$$\begin{cases} x_1 = C_1 \\ x_2 = \sin(x_1) - C_2 \\ x_3 = \sin(x_1) - x_1 - x_2 \\ x_{25} = 1 \end{cases}.$$

There is no such conception as «substitutions» in Model Vision Modeling language (MVL), but MvStudium's Analyzer can detect formal substitutions of type

$$x_i = f_i(x_1, x_2, ..., x_n, time)$$

by itself.

Formal substitutions may be considered as equations of course, but it increases size of solved system and may cause difficulties for Newton's method (slow convergence, bad initial conditions).

Detection of formal substitutions among equations and reordering them if necessary is a goal of Analyzer. Find substitutions should form a sequence suitable for calculations.

Using different sequences of substitutions leads to different final systems.

ISBN: 978-1-61804-251-4

Example 2. Initial User's sequence

$$\begin{cases} x_1 = -16 \cdot x_4 + 1 \\ x_2 = \sqrt{|x_3|} \\ x_3 = 25 \cdot x_2 - \dfrac{1}{x_3 + 16} \\ x_4 = x_1 + 25 \end{cases}$$

may be transformed into three final systems with substitutions - **A**), **B**) or **C**).

Final system with substitutions A. Two substitutions and two equations. (Symbol «:= » is used for substitutions).

$$\begin{cases} x_1 := -16 \cdot x_4 + 1 \\ x_2 := \sqrt{|x_3|} \\ x_3 = 25 \cdot x_2 - \dfrac{1}{x_3 + 16} \\ x_4 = x_1 + 25 \end{cases},$$

Final system with substitutions B. One substitution and three equations.

Subroutine for calculation residuals may be written in the form

$$F(x_1, x_3, x_4) = \begin{pmatrix} x_3 - 25 \cdot x_2 + \dfrac{1}{x_3 + 16} \\ x_1 + 16 \cdot x_4 - 1 \\ x_4 - x_1 - 25 \end{pmatrix}, x_2 := \sqrt{|x_3|}$$

Final system with substitutions C. Block-diagonal system. First block: one substitution and one nonlinear equation. Second block: two linear equations.

$$\begin{cases} x_2 := \sqrt{|x_3|} \\ x_3 - 25 \cdot x_2 + \dfrac{1}{x_3 + 16} = 0 \end{cases}$$

$$\begin{cases} x_1 + 16 \cdot x_4 = 1 \\ x_4 - x_1 = -25 \end{cases}$$

Using equations as substitutions may be dangerously even for linear systems.

Let us consider a system of linear equations

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{A}_{12} \\ \mathbf{A}_{21}\mathbf{A}_{22} \end{bmatrix}; \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix},$$

$$\mathbf{A}_{12} = \mathbf{A}_{21} = \mathbf{E}$$

$\mathbf{E}$ - identity matrix, $\mathbf{A}$ - square matrix with square block matrices of the same dimension, and let $\mathbf{A}$ is not singular.

It is possible to build two new systems with substitutions

$$\begin{cases} \mathbf{x}_2 := \mathbf{b}_1 - \mathbf{A}_{11} \cdot \mathbf{x}_1 \\ \mathbf{x}_1 + \mathbf{A}_{22} \cdot \mathbf{x}_2 = \mathbf{b}_2 \end{cases}$$

or

$$\begin{cases} \mathbf{x}_1 := \mathbf{b}_2 - \mathbf{A}_{22} \cdot \mathbf{x}_2 \\ \mathbf{A}_{11} \cdot \mathbf{x}_1 + \mathbf{x}_2 = \mathbf{b}_1 \end{cases},$$

but their numerical properties depend now on conditional numbers of matrices

$$\mathbf{E} - \mathbf{A}_{22}\mathbf{A}_{11}$$
$$\mathbf{E} - \mathbf{A}_{11}\mathbf{A}_{22},$$

and even there is no insurance arrangements that they are not singular.

II. Construction formally calculable sequence of substitutions

Definition. *Sequence of formal substitutions*

$$x_i = f_i(x_1, x_2, ..., x_n, time), i = k, m \quad (2)$$
$$1 \le k \le m \le n$$

is called formally calculable if each its member contains in right-hand function only already calculated unknowns.

It means that square $n \times n$ structure matrix $\mathbf{S}$ for *right-hand* functions $f_i(x_1, x_2, ..., x_n, time), i = 1, n$ with n arguments x_i is square *lower triangular* matrix with zero diagonal.

Example 3. Formally calculable sequence of substitutions and its structure matrix.

$$\begin{cases} x_1 = C_1 \\ x_2 = \sin(x_1) + C_2 \\ x_3 = -\sin(x_1) + x_1 + x_2 \\ x_4 = 1 \end{cases} \qquad \mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

However a sequence of formal substitutions may contains equations:

$$\begin{cases} x_1 = C_1 \\ x_2 = \sin(x_1) + C_2 + x_3 \\ x_3 = -\sin(x_1) + x_1 + x_2 \\ x_4 = 1 \end{cases},$$

In this case Analyzer has to divide formal substitutions on formally calculable substitutions and equations.

Initial information about sequence of formal substitutions may be written in the form (3)

$$\mathbf{x} = \mathbf{S} \cdot \mathbf{x} + \mathbf{C}; \mathbf{x}, \mathbf{C} \in \Re^n; S \in \Re^{n \times n} \quad . \quad (3)$$

Let us concatenate with sequence of formal substitutions (3) a system of linear algebraic equations

$$\mathbf{S} \cdot \mathbf{y} = 0 \quad (4)$$

ISBN: 978-1-61804-251-4

with structure matrix $\mathbf{S}$. Simulating elimination of variables in (4) it is possible to detect is (3) *formally calculable* or not.

Example 3. Consider matrix

$$\mathbf{S} = \begin{bmatrix} 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 1\ 1\ 0 \end{bmatrix}.$$

First zero line says that $y_1 = C_1, (f_1(t))$. Let us eliminate unknown y_1, removing first line and first colon from the matrix

$$\mathbf{S}_1 = \begin{bmatrix} 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 1\ 0\ 0 \\ 1\ 1\ 0\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 1\ 0 \end{bmatrix}.$$

Then it will be possible eliminate y_2 and substitute y_3 into 4,5,6-th equations

$$\mathbf{S}_3 = \begin{bmatrix} 0\ 1\ 0\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0 \end{bmatrix}.$$

Final matrix $\mathbf{S}_3$ is a block-triangular matrix. Its first diagonal block

$$\mathbf{S}_{11} = \begin{bmatrix} 0\ 1 \\ 1\ 1 \end{bmatrix}$$

corresponds to equations, and second

$$\mathbf{S}_{22} = \begin{bmatrix} 0\ 0 \\ 1\ 0 \end{bmatrix}$$

to substations.

This method of detection of substitutions leads to fill-in effect as Gaussian Elimination.

III. Using Tarjan's Algorithm for Detection Substitutions

Oriented graphs usually are used for description of a structure of sparse systems of equations [4].

Fig. 1 illustrates using oriented graphs for description structure of a) - systems of equations, b) - systems of equations with substitutions, c) – system of substitutions.

$$a)\ \begin{cases} x_1 = 10 \cdot x_1 + 16 \cdot x_2 - 1 \\ x_2 = 10 \cdot x_2 - x_1 - 25 \end{cases};\quad b)\ \begin{cases} x_1 = -25 \\ x_2 = x_3 - 15 + x_1 \\ x_3 = 10 \cdot x_2 - 15 \\ x_4 = x_1 + x_3 \end{cases};$$

$$c)\ \begin{cases} x_1 = -1 \\ x_2 = x_1 - 25 \\ x_3 = x_2 - 15 \end{cases}$$

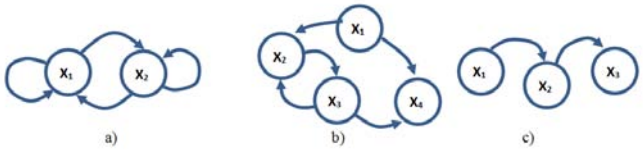


Fig. 1 Graphs for systems and substitutions

Tarjan's algorithm [1] for block triangularization of a matrix was used in MA28 [2]. We suggest using Tarjan's algorithm not only for reducing a matrix to block-triangular form but for detection of substitutions among equations.

Tarjan's algorithm will detect one strongly connected component for system a), three – for system b) and c). There are strongly connected component of dimension 1×1 in case of b), c). Those are substitutions.

Example 4. Matrix $\mathbf{S}$

$$\mathbf{S} = \begin{bmatrix} 1\ 1\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0 \end{bmatrix} = \begin{bmatrix} S_{11} & 0 & 0 \\ S_{21} & S_{22} & 0 \\ S_{31} & S_{32} & S_{33} \end{bmatrix},$$

has block-triangular form. Each diagonal block S_{ii} corresponds to strongly connected component of oriented graph. Block S_{11} has dimensions 2×2, therefore there is one equation as minimum. If to solve the system corresponding to the first block, then sufficiently calculating only substitutions corresponding to blocks 2 and 3 (S_{22}, S_{33}).

ISBN: 978-1-61804-251-4

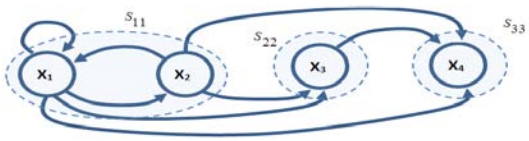


Fig. 2 Graph for the system with substitutions

Algorithm for detecting formally calculable sequence of substitutions.

Stage 1. Detect formal substitutions among equations

$$x_i = f_i(x_1, x_2, ..., x_n, time)$$

Stage 2. Delete from derived sequence explicit «members-equations»

$$x_i = f_i(x_1, .. \ \ x_i, \ \ ....x_n, time)$$

Stage 3. Detect strongly connected component with the help of Tarjan's algorithm.

Stage 4. Divide strongly connected component on substitutions (dimension 1×1) and equations (dimension greater then 1×1)

Stage 5. Reorder sequence and get *formally calculable* sequence.

Example 5. The system of formal substitutions

$$\begin{cases} x1 = -7 * x7 + x8 - 2 \\ x2 = x4 * 3 - 2 * x6 + 1 \\ x3 = \dfrac{x5}{2} + 8 \\ x4 = x6/x1 + 12 \\ x5 = (x7 + x8 + x3)^2 - 23 \\ x6 = x3 + x2 - 3 \\ x7 = 2 * x3 + 2 \\ x8 = (x7 - 2)^3 + 2 \end{cases}$$

contains among substitutions «algebraic loops» (for example, x2→x4→x6→x2), which should be detected [3] and removed («cutting algebraic loops»). Let us apply described algorithm to given system of formal substitutions.

Step 1.

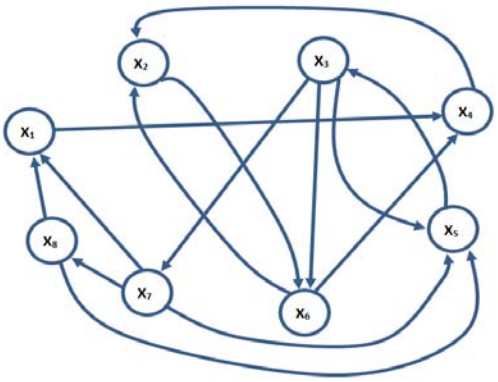


Fig. 3. Initial graph.

Step 2.

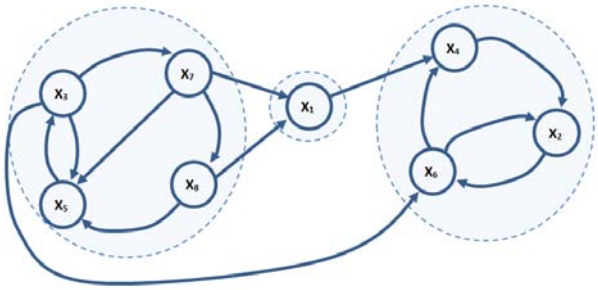


Fig. 4a. Strongly connected components detected by Tarjan's algorithm.

Edge's orientations (Fig 4.b) says about existance block-triangular form.

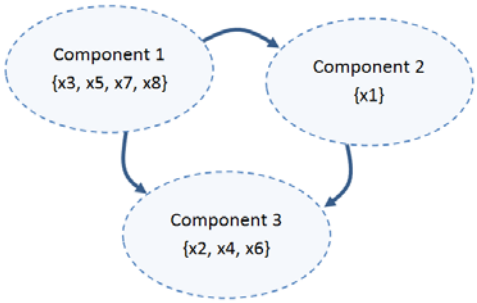


Fig. 4b. Links between strongly connected components.

Component 2 corresponds to substitution, components 1,3 – to equations.

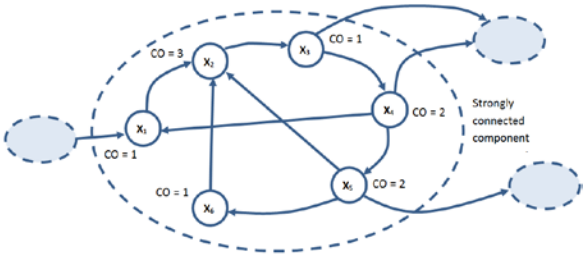
«Algebraic loops»

1. $x_3 \rightarrow x_5 \rightarrow x_7 \rightarrow x_8 \rightarrow x_3$
2. $x_2 \rightarrow x_4 \rightarrow x_6 \rightarrow x_2$

have to be founded into components 1,3.

It is obvious that «algebraic loop» may be cutting in different ways. We will choose equation applying principle of maximum cycling order for graph's nodes. Cycling order (CO) for node) is OC = max(number of outgoing arc, number of incoming arc) .

Outgoing and incoming arc have leading to nodes belong to strongly connected component (Fig. 5).



Fiq. 5. Strongly connected component with calculated cycling orders.

ISBN: 978-1-61804-251-4

Step 3. Calculating cycling orders. Fig. 6.a, 6.b.

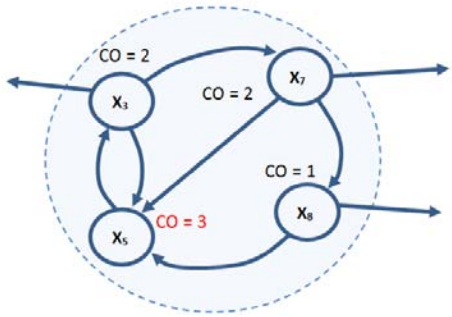


Fig. 6.a. Cycling orders for component 1.

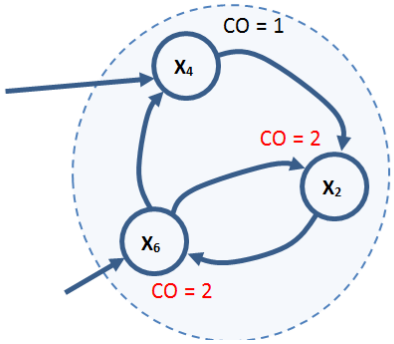


Fig. 6.b. Cycling orders for component 3.

Component 1 has only one node with maximum cycling order (node x_5, CO=3). Corresponding «substitution» becomes «equation».

Component 3 has two nodes with maximum cycling orders. Node x_2 was the first in Tarjan's algorithm list of nodes belong to Component 3. So corresponding «substitution» becomes «equation».

Step 4. Remove nodes marked as equations

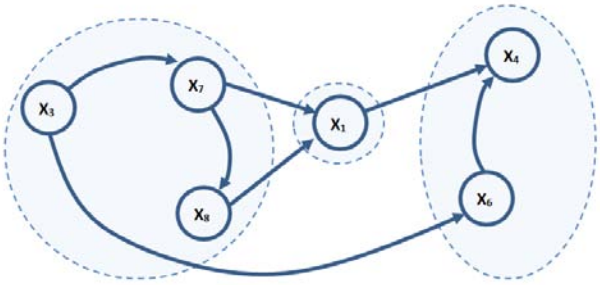


Fig. 7. Graph after removing «node-equation»

After removing nodes Component 1 and component 3 become not strongly connected components.

Step 5. Restart Tarjan's algorithm only for Components 1,3. Final graphs is shown on Fig. 8. Strongly connected components have dimensions 1×1, therefore all «algebraic loops» are detected and corresponding «substitutions» are announced as «equations».

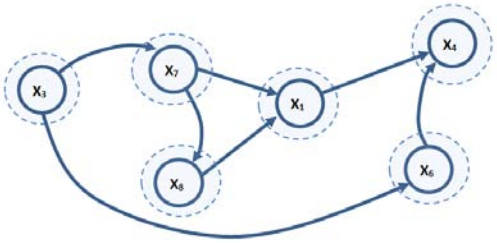


Fig. 8. Final graph.

Step 6. Building system of equations and substitutions

- Formal substitutions announced as equations:

$$\begin{cases} x2 = x4*3 - 2*x6 + 1 \\ x5 = (x7 + x8 + x3)^2 - 23 \end{cases}$$

- *formally calculable* sequance

$$\begin{cases} x3 = \dfrac{x5}{2} + 8 \\ x7 = 2*x3 + 2 \\ x8 = (x7 - 2)^3 + 2 \\ x1 = -7*x7 + x8 - 2 \\ x6 = x3 + x2 - 3 \\ x4 = x6/x1 + 12 \end{cases}$$

IV. Conclusion

Suggested algorithm was tested on TRANSAS' problems (www.transas.com) and demonstrated high performance. Extended version of this paper will contain results of computations experiments with TRANSAS models.

References

[1] Tarjan R. E. Depth-first search and linear graph algorithms. SIAM Journal on Computing. – 1972. – № 2. – p. 146–160

[2] Duff I. S. MA28 – a set of FORTRAN subroutines for sparse unsymmetric linear equations, 1977.

[3] The MathWorks Inc. Simulink. Simulation and Model-Based Design. – Eighth Printing – 2005.

[4] Pissanetzky S. Sparse matrix technology. – 1984.

Isakov Andrey A. -- MPhil (Computer Science, 2012, SPBSTU), postgraduate student of Distributed Computing and Networking Computing department, Saint Petersburg state Polythecnical University.

Senichenkov Yuri B. – DPhil (Numerical software, 2005, SPBSTU), Professor of Distributed Computing and Networking Computing Department, Saint Petersburg state Polythecnical University.