Journal of Theoretical and Applied Information Technology
31st October 2022. Vol.100. No 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GENERATION OF MATHEMATICAL MODELS OF LINEAR DYNAMIC SYSTEMS DESCRIBED BY BLOCK DIAGRAMS

MIKHAIL SIMONOV[1], KONSTANTIN ZAYTSEV[2], NATALYA POPOVA[3]

[1]National Research Aviation University MAI (Moscow Aviation Institute), Moscow, Russia

[2]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russia

[3]Limited Liability Company "Center of Special Engineering", Moscow, Russia

E-mail: kszajtsev@gmail.com

ABSTRACT

The purpose of the study is to develop a method for calculating the mathematical models of block diagrams in the form of equations in a state-space representation. The model is calculated through an iterative process. The initial condition of the mathematical model is taken as a model consisting of unconnected elements of the block diagram. The hierarchy of elements, in general, is not set and is arbitrary, but remains unchanged throughout the computation. The iterative process is composed of steps, each of which processes one connection. The connection bridges the output and input ports of the element(s) in the block diagram. It is helpful (for example, to obtain the matrix frequency response of the system) to implement an algorithm for calculating the current connection without excluding and without zeroing the rows and columns of the equation-of-state matrices corresponding to this connection. This corresponds to a derivation of an equivalent model of a block diagram with input ports accessible for signal input and output ports of all elements composing the block diagram being open for observation. In this case, the definition of indexes at all steps of the algorithm remains the same as at step zero. The paper demonstrates that if a linear dynamic system contains a solvable system of algebraic loops, the mathematical model of a said dynamic system can be obtained as equations of state. In this case, the condition is derived, under which the system of algebraic loops can be solvable, and then the mathematical model of the block diagram can be obtained. If this condition is not met, the block diagram is to be rebuilt. The paper provides an example of step-by-step construction of a control system model defined by a block diagram and a class diagram of a system defined by a block diagram.

Keywords: *Mathematical Model, Block Diagram, Algebraic Loop, Object-Oriented Development*.

1. INTRODUCTION

Applied control engineering has mostly taken its shape and has become a subject of academic disciplines at universities. Computer-Aided Control System Design (CACSD) systems are fundamental tools used for analysis and synthesis in various subject areas [1,2]. The traditional way of graphical representation of mathematical models is a block diagram, which defines the main functional elements of the system and their mutual relationships [3].

Among the first developments in this area were instruments based on modeling languages and designed for solving systems of ordinary differential equations (ODEs). Since 1974, scholars have created over forty languages for modeling continuous systems, which evolved from attempts to present the behavior of analog computers, widely used until the 1960s, in a digital form. A famous development from this period is Continuous System Simulation Language (CSSL), a language that defines the system as a compound of blocks, emulates basic analog simulation blocks, and finds a numerical ODE solution. Advanced Continuous Language (ACSL for FORTRAN) [4] is one of the successful implementations of CSSL. Later there appeared two important developments: declarative or equation-based language modeling and object-oriented modeling. Declarative languages do not impose a fixed causality [5] on the model. In these languages, the model is defined by a series of equations that establish the relationship between states, their derivatives, and time. A certain procedure in the modeling process performs the conversion of these equations into software, which is executed by the computer. The advantage of declarative languages is that users do not have to define the mathematical causality of the equations, so the model can be used for any causality imposed by another system

component [6]. Further research is based on the development of component-based modeling [7].

An important step was the development of graphical user interfaces that provide modeling based on block diagrams. Landmark developments were SystemBuild [8] and SIMULAB (later renamed Simulink) [9]. Both products use a customizable set of block libraries that have been integrated into two powerful interactive computing systems: $MATRIX_X$ and MATLAB.

Currently, the widely recognized MATLAB engineering and scientific calculation system and Simulink simulation system [10,11] are integrating many new methodological developments in the sphere of control associated with toolbar control. MATLAB is also developing as a powerful object-oriented programming language. Existing CACSD tools are constantly being expanded to include new classes of models and new computational algorithms.

Block diagrams are used for representing and visualizing dependencies in models of complex dynamic systems. Such schemes can represent different levels of complexity: the blocks that exist in a diagram can be mathematical operators, a description in the form of a system of equations, and subsystems. The syntax of structural diagrams used by engineers is not formally regulated. However, a number of technologically oriented forms of block diagrams are standardized. The IEC 61131-3:2003 (2013) [12] standard describes programming languages for programmable logic controllers. In this case, a graphical language is a software tool for describing a diagram of a set of connected functional blocks. The ISO 10628:2014 [13] standard regulates diagrams for the chemical and petrochemical industry, defines the representation of block diagrams, and serves to indicate the general flow of plant processes (process flow diagram, PFG).

The computational approach to modeling, in some cases, has many advantages over physical modeling [14], because developing a computational model is both easier and less costly [15]. In addition, computational models are more flexible and modifiable. Control system models are developed by engineers in different fields and for different purposes. For example, when designing a control law, the model of an object is often given in the form of continuous differential equations, because they can provide automatic methods of synthesis [10].

In developing control systems with a typical computer-aided design system (CACSD), there are several steps to be followed. The mathematical model of the system is developed in mathematical terms, reflects the qualities of the real system, and has a quantitative description. The construction of the model is the first step, which consists in the development of a certain equivalent that mathematically represents the laws it abides and the connections between its components. Then, the factors that are non-essential to the behavior of this equivalent are discarded and additional factors, also written in mathematical terms, are considered. The mathematical model is studied using theoretical methods, which produces preliminary data. Dynamic models are typically represented by systems of ODEs [16,17], differential-algebraic equations (DAEs) [18], or partial differential equations (PDEs). Here it is essential that the model research goal is present and formulated. In mathematical modeling, which relies on numerical methods, the second step has to be the representation of the model through a computational algorithm [19,20]. In the case of block diagrams being used for the representation of the model, the second step is the application of the chosen method of forming the model of a set of interconnected parts that make up the scheme.

However, for the purposes of control system synthesis, simplification of models is required, e.g., linearization, discretization, or model reduction, or directly defined in some form data from the measured inputs — outputs that use system identification methods. The commonly used continuous or discrete-time synthesis models, linear time-invariant (LTI) systems describe the nominal behavior of object models at a certain operating point [21]. More precise linear parameter-varying (LPV) models can serve to account for uncertainties due to various approximations and nonlinearities performed or changes in model parameters.

The subject of this article is the development of a method (algorithm) for constructing a mathematical model of a block diagram for linear continuous systems.

2. MATERIALS AND METHODS

2.1 Basic Provisions. Problem Statement

The set of states of a mathematical model of a system or process is related to the metric space of states of this system or process. When the system interacts with other systems, a set of controls with elements $u \subseteq U$ and a set of influences $w \subseteq W$ are introduced. Each system can be decomposed into interrelated subsystems. In this case, the space of states X is represented as a sum of subspaces of states that are a section of the space X. The equations

of state for a deterministic linear continuous controlled system in time under deterministic influences can be written as

$$x^{\cdot}(t) = Ax(t) + Bu(t); x(t_0) = x_0; u(t_0) = u_0. y(t) = Cx(t) + Du(t), \quad (1)$$

where x – dimension state vector n, u – dimension input vector k, y – dimension output vector m, and the matrices of state – A, input – B, output – C, and direct transmittance – D have consistent dimensions.

A structural diagram is defined by a set of diagram elements and connection lines linking these elements. Each element has a certain number of inputs and outputs, which are uniquely represented by the input and output ports in the diagram. The connection line uniquely defines the pair to be connected: the output port and the input port.

Each element in the block diagram is defined by its mathematical description in the form of state equations in the state space of the form (1).

The task set is to obtain a mathematical model in the form of state equations of the form (1) for a given block diagram.

2.2 Computational Method for Constructing a Mathematical Model of a Control System Defined by a Linear Block Diagram

Assume that the block diagram contains N linear elements mathematically described in state variables (1).

The computation of the mathematical model is constructed in the form of an iterative process. The initial condition of the mathematical model (the zero step of the iteration, as well as all subsequent steps, is indicated by the upper index in brackets) is taken as a model consisting of unconnected elements of the block diagram.

$$x^{\cdot}(t) = A^{(0)}x(t) + B^{(0)}u^{(0)}(t);\ y^{(0)}(t) = C^{(0)}x(t) + D^{(0)}u^{(0)}(t), \quad (2)$$

where

$$\begin{aligned} A^{(0)} &= diag\ (A_1, A_2, \dots, A_N) \\ B^{(0)} &= diag\ (B_1, B_2, \dots, B_N) \\ C^{(0)} &= diag\ (C_1, C_2, \dots, C_N) \\ D^{(0)} &= diag\ (D_1, D_2, \dots, D_N) \end{aligned} \quad (3)$$

In these equations, N is the number of structural elements with dimensions $n_1, n_2, \dots n_N$, $k_1, k_2, \dots, k_N$, $m_1, m_2, \dots, m_N$; the state vector x with the dimension $n=n_1+n_2+\dots+n_N$, input vector $u^{(0)}$ with the dimension $k=k_1+k_2+\dots+k_N$, and the output vector $y^{(0)}$ with the dimension $m=m_1+m_2+\dots+m_N$.

The order of the elements is arbitrary but constant throughout the calculation process. This entails that all input and output ports on the block diagram, having their sequence number within its structural element, unambiguously receive a unique sequence number within the description of the entire block diagram (2)-(3).

The iterative process of calculating the mathematical model consists of steps, each of which processes one connection represented on the block diagram by one communication line, which uniquely defines one output port and one input port of the diagram.

Taking relations (2)-(3) as the initial state of the mathematical model of the diagram, at the first step of the iterative process, we choose a particular connection (connection line), thereby explicitly defining the output port, let it be $i^{(0)}$, and the input port, let it be $j^{(0)}$, in the (2)-(3) model. Then the condition for this connection in the model of scheme (2) will be the equation

$$u_j^{(0)} = y_i^{(0)} \quad (4)$$

(in this expression, the top index denoting the iteration number for indexes i and j is omitted since the iteration number is specified in the expression itself). The use of (4) together with (2)-(3) delivers the model of the block diagram at the first step of the iterative process

$$x^{\cdot} = A^{(1)}x + B^{(1)}u^{(1)};\ y^{(1)} = C^{(1)}x + D^{(1)}u^{(1)}, \quad (5)$$

Here the vector of open outputs in the first iteration $y^{(1)}$ is obtained by excluding from the output vector of the zero iteration the component $y_i^{(0)}$. The vector of open inputs in the first iteration

$u^{(1)}$ is acquired by excluding the component $u_j^{(0)}$ from the input vector of the zero iteration.

If $1 - d_{i,j}^{(1)} \neq 0$, then the matrices of the diagram model at the first step are calculated through the matrices of the zero-step model by formulas (6, 7)

$$A^{(1)} = A^{(0)} + \frac{1}{1 - d_{ij}^{(0)}} b_j^{(0)} c_i^{(0)};\ B^{(1)} = B_j^{(0)} + \frac{1}{1 - d_{ij}^{(0)}} b_j^{(0)} d_{\overline{j}}^{(0)}, \quad (6)$$

$$C^{(1)} = C_i^{(0)} + \frac{1}{1 - d_{i,j}^{(0)}} d_{j/i}^{(0)} c_i^{(0)};\ D^{(1)} = D_{i,j}^{(0)} + \frac{1}{1 - d_{i,j}^{(0)}} d_{j/i}^{(0)} d_{i/j}^{(0)} \quad (7)$$

where the $B_j^{(0)}$ matrix is formed from the $B^{(0)}$ matrix by extracting from it the $j^{(0)}$-th column, $b_j^{(0)}$;
the $C_i^{(0)}$ matrix is formed from the $C^{(0)}$ matrix by extracting from it the $i^{(0)}$-th row, $c_i^{(0)}$;
the $D_{i,j}^{(0)}$ matrix is formed from the $D^{(0)}$ matrix by extracting from it the $i^{(0)}$-th row, $d_i^{(0)}$, and the $j^{(0)}$-th column, $d_j^{(0)}$; $d_{\overline{i}}^{(0)}$ is the $j^{(0)}$-th column of the $D^{(0)}$ matrix without the $i^{(0)}$-th component; $d_{\overline{j}}^{(0)}$ is the $i^{(0)}$-th row of the $D^{(0)}$ matrix without the $j^{(0)}$-th component.

The further iterative process of calculating the mathematical model of the control system is described by the following recurrence relations.

Let the current mathematical model of the control scheme at the l-th step of the iterative process (the 1st, 2nd, ..., l-th connections already entered into the model) be calculated as

$$x = A^{(l)}x + B^{(l)}u^{(l)};$$
$$y^{(l)} = C^{(l)}x + D^{(l)}u^{(l)} \quad (8)$$

At the $(l+1)$-th step, the iterative process addresses the $(l+1)$-th connection line with unambiguously defined input and output ports of the block diagrams. This explicitly (but perhaps nontrivially) defines the output, let it be $i^{(l)}$, and the input, let it be $j^{(l)}$, in the model (8) constructed at step l. Then the condition for this connection in the model (8) will be the equation

$$u_j^{(l)} = y_i^{(l)} \quad (9)$$

the use of which delivers the model at the $(l+1)$-th step of the iterative process

$$x = A^{(l+l)}x + B^{(l+l)}u^{(l+l)}; \quad (10)$$

$$y^{(l+l)} = C^{(l+l)}x + D^{(l+l)}u^{(l+l)}$$

where the matrices are recurrently defined as

$$A^{(l+1)} = A^{(l)} + \frac{1}{1 - d_{i,j}^{(l)}} b_j^{(l)} c_i^{(l)};$$
$$B^{(l+1)} = B_j^{(l)} + \frac{1}{1 - d_{i,j}^{(l)}} b_j^{(l)} d_{i/j}^{(l)};$$
$$C^{(l+1)} = C_i^{(l)} + \frac{1}{1 - d_{i,j}^{(l)}} d_{j/i}^{(l)} c_i^{(l)}; \quad (11)$$
$$D^{(l+1)} = D_{i,j}^{(l)} + \frac{1}{1 - d_{i,j}^{(l)}} d_{j/i}^{(l)} d_{i/j}^{(l)}$$

The relations (11) assume that $1 - d_{i,j}^{(l)} \neq 0$ and uses the following designations:

$b_j^{(l)}$ – the $j^{(l)}$-th column of matrix $B^{(l)}$; $c_i^{(l)}$ –the $i^{(l)}$-th column of matrix $C^{(l)}$; $d_{\overline{i}}^{(l)}$ – the $j^{(l)}$-th column of matrix $D^{(l)}$ without the $i^{(l)}$-th element; $d_{\overline{j}}^{(l)}$ – the $i^{(l)}$-th column of matrix $D^{(l)}$ without the $j^{(l)}$-th elementa; $B_j^{(l)}$ – the matrix obtained by excluding the $j^{(l)}$-th column from $B^{(l)}$; $C_i^{(l)}$ – the matrix obtained by excluding the $i^{(l)}$-th row from $C^{(l)}$; $D_{i,j}^{(l)}$ – the matrix obtained by deleting the $i^{(l)}$-th row and the $j^{(l)}$-th column from $D^{(l)}$.

2.3 Determination of the Numbers of Diagram Input and Output Ports

The $i^{(l)}$ and $j^{(l)}$ indices need to be recalculated at each step minding the rows and columns of input, output, and bypass matrices of the diagram deleted at the preceding step. Calculations can be performed by zeroing out the corresponding rows and columns

instead of deleting them. This provides an equivalent but redundant closed-loop scheme with fully-dimensional input and output vectors corresponding to the input and output ports of all the elements composing the block diagram. In this case, the calculation of input/output port indices at all steps of the algorithm remains the same as at step zero. To strictly obtain the resulting model of the diagram, it is necessary to delete the rows and columns of the input, output, and bypass matrices of such a redundant model canceled as a result of the implementation of the structural connection.

The case of detecting an algebraic loop (a loop without dynamics) in the block diagram

The presence of algebraic loops causes complications with numerical methods [22-24]. In this section, we will demonstrate that it is possible to obtain a mathematical model of a dynamic system in the form of equations of state for a linear dynamical system if it contains a solvable system of algebraic loops. A condition is derived, under which the system of algebraic loops can be solvable, and then the mathematical model of the block diagram can be obtained. If this condition is not met, the block diagram is to be rebuilt.

Returning to the proposed method, we note that the condition for the resolution of the algebraic contour at the first calculation step is the relation $(1 - d_{ij}^{(0)}) \neq 0$. Similarly, at the l-th step of the algorithm, the condition for the resolution of the algebraic loop is $(1 - d_{ij}^{(l)}) \neq 0$. If at the intermediate step of the algorithm it is found that $\left(1 - d_{ij}^{(l)}\right) = 0$, consideration of the respective connection has to be delayed until the last step of the algorithm, moving to the next connection of the set in the current step.

Assume that at the l-th step of the algorithm, all remaining unprocessed connections are found to form a system of algebraic loops, each of which individually cannot be solved with respect to the scalar output by the algorithm described above. Let there be m of such connections. For each k-th of them, the numbers of the diagram input port, j_k, and the diagram output port, i_k are known. From them, we form vectors $J = [j_1 \cdots j_k \cdots j_m]$ and $I = [i_1 \cdots i_k \cdots i_m]$. Then the condition for the connection will be the vector equality

$$u_J^{(l)} = y_I^{(l)} \qquad (12)$$

where the $u_J^{(l)}$vector is obtained by selecting elements with numbers determined by vector J from vector $u^{(l)}$, the $y_I^{(l)}$ vector is formed by picking out elements with numbers determined by vector I from vector $y^{(l)}$, and equations written down as

$$x(t) = A^{(l)}x(t) + B^{(l)}u^{(l)}(t);$$
$$y^{(l)}(t) = C^{(l)}x(t) + D^{(l)}u^{(l)}(t); \qquad (13)$$
$$u_J^{(l)} = y_I^{(l)}$$

define initial data for constructing the model of the diagram if the system of algebraic loops turns out to be solvable.

Let $C_I^{(l)}$ denote the matrix obtained by selecting from matrix $C^{(l)}$the rows with numbers given by vector I and $D_I^{(l)}$ signify the matrix formed by selecting from matrix $D^{(l)}$the rows with numbers set by vector I. As a result, we have

$$y_I^{(l)} = C_I^{(l)}x + D_I^{(l)}u^{(l)} \qquad (14)$$

Let $D_{IJ}^{(l)}$ denote the matrix acquired by picking out columns with numbers given by vector J from matrix $D_I^{(l)}$ and $D_{I\bar{J}}^{(l)}$ signify the matrix comprised of the remaining columns. Then the following equation is true

$$y_I^{(l)} = C_I^{(l)}x + D_{IJ}^{(l)}u_J^{(l)} + D_{I\bar{J}}^{(l)}u \qquad (15)$$

Using the condition for connection (12), we obtain a system of equations describing the system of algebraic loops

$$(I_m - D_{IJ}^{(l)})u_J^{(l)} = C_I^{(l)}x + D_{I\bar{J}}^{(l)}u \qquad (16)$$

the solvability condition of which is

$$det(I_m - D_{IJ}^{(l)}) \neq 0 \qquad (17)$$

where I_m is an identity matrix of dimension $m \times m$.

If the condition (17) is met, the system of algebraic loops (16) has a solution

$$u_J^{(l)} = (I - D_{IJ}^{(l)})^{-1}C_I^{(l)}x + (I - D_{IJ}^{(l)})^{-1}D_{I\bar{J}}^{(l)}u \qquad (18)$$

Let $B_J^{(l)}$ be the matrix obtained by selecting from matrix $B^{(l)}$ the columns with numbers given by vector J, and let $B_{\bar{J}}^{(l)}$ be the matrix comprised by the rest of the columns. Further, let $D_J^{(l)}$ denote the matrix acquired by selecting from matrix $D^{(l)}$ the columns with numbers set by vector J and $D_{\bar{J}}^{(l)}$ be the matrix formed from the remaining columns. From the output vector $y^{(l)}(t)$ in the system of equations (13), we form the vector of open ports in the system $y(t)$. For this, we form the $C_{\bar{I}}^{(l)}$ matrix by crossing out the rows set by vector I from matrix $C^{(l)}$, the $D_{\bar{I}J}^{(l)}$ matrix by crossing out the rows set by vector I from matrix $D_J^{(l)}$, and matrix $D_{\bar{I}\bar{J}}^{(l)}$ by crossing out the rows set by vector I from matrix $D_{\bar{J}}^{(l)}$. Then from the system of equations (13) we can find that

$$x(t) = A^{(l)}x(t) + B_J^{(l)}u_J^{(l)}(t) + B_{\bar{J}}^{(l)}u(t)$$
$$y(t) = C_{\bar{I}}^{(l)}x(t) + D_{\bar{I}J}^{(l)}u_J^{(l)}(t) + D_{\bar{I}\bar{J}}^{(l)}u(t) \quad (19)$$

whence, using expression (18) of the system of algebraic loops, we have

$$x(t) = \left[A^{(l)} + B_J^{(l)}\left(I - D_{IJ}^{(l)}\right)^{-1}C_I^{(l)}\right]x(t) + [\mathrm{B}_{\bar{J}}^{(l)} + B_J^{(l)}\left(I - D_{IJ}^{(l)}\right)^{-1}D_{I\bar{J}}^{(l)}]u(t)$$
$$y(t) = \left[\mathrm{C}_{\bar{I}}^{(l)} + D_{\bar{I}J}^{(l)}\left(I - D_{IJ}^{(l)}\right)^{-1}C_I^{(l)}\right]x(t) + [\mathrm{D}_{\bar{I}\bar{J}}^{(l)} + D_{\bar{I}J}^{(l)}\left(I - D_{IJ}^{(l)}\right)^{-1}D_{I\bar{J}}^{(l)}]u(t) \quad (20)$$

The resulting system of equations is a model of the diagram in state variables for the considered case of the presence of a solvable system of algebraic contours (16) in the scheme.

If condition (17) is not satisfied, it means that there is an unsolvable system of algebraic contours in the studied diagram. Such a block diagram should be re-formed.

3. RESULTS

3.1 An Example of Operation of the Computational Method

Let us consider the block diagram presented in Figure 1.

As an example, we take the following numerical data:

$k_0 = 1.0$; $k_1 = 0.9$; $k_2 = 0.05$; $k_3 = 1.1$; $T = 0.006$; $k_4 = 3.0$; $k_5 = 2.0$; $l = 4.0$.

Each element of the diagram is assigned its own sequence number, denoted in Roman numerals on the diagram.

The element, called Object, is numbered as the first one. The transfer function of the Object is known to be $W_o(s) = \frac{4.0}{s^3+3.0s^2+2.0s}$. According to the conditions of use of the proposed method, the mathematical model of each element of the diagram is set by the equations of state. To this Object correspond the following matrices of the mathematical model in the state space:

$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2.0 & -3.0 \end{bmatrix}, B_1 = \begin{bmatrix} 0 \\ 0 \\ 4.0 \end{bmatrix}, C_1 = [1 \quad 0 \quad 0], D_1 = [0]$.

The element of the diagram denoted as PID controller is numbered as the second. Its transfer function is known to be $W_{\mathrm{p}}(s) = 0.9 + 0.05/s + 1.1s/(0.006s + 1)$. To this element corresponds the mathematical model in the state space defined by matrices:

$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -166.67 \end{bmatrix}, B_2 = \begin{bmatrix} -30555.51 \\ 5092592.59 \end{bmatrix}, C_2 = [1\ 0], D_2 = [184.23]$.

The summer is numbered as the third element in the diagram, branching is numbered fourth, and the amplification coefficient k_0 in the feedback loop is numbered fifth. These elements in the state space are given by bypass matrices: $D_3 = [1 \quad -1]$, $D_4 = [1 \quad 1]$, and $D_5 = [1.0]$, respectively.

The ports of each diagram element are numbered according to its mathematical model.

All data about the elements of the diagram necessary for the construction of its mathematical model is presented in Table 1.

Journal of Theoretical and Applied Information Technology
31st October 2022. Vol.100. No 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: The set of Elements in the Diagram

Number in the figure	Dimensions	State matrix	Input matrix	Output matrix	Direct transmittance matrix
I	$n=3,k=1,m=1$	$\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -2.0 & -3.0\end{bmatrix}$	$\begin{bmatrix}0\\0\\4.0\end{bmatrix}$	$[1 \quad 0 \quad 0]$	[0]
II	$n=2,k=1,m=1$	$\begin{bmatrix}0 & 1\\ 0 & -166.67\end{bmatrix}$	$\begin{bmatrix}-30555.51\\5092592.59\end{bmatrix}$	$[1 \quad 0]$	[184.23]
III	$n=0,k=2,m=1$	–	–	–	$[1 \quad -1]$
IV	$n=0,k=1,m=2$	–	–	–	$\begin{bmatrix}1\\1\end{bmatrix}$
V	$n=0,k=1,m=1$	–	–	–	[1.0]

All declared connections in the diagram are shown in Table 2. Each connection is defined by the output and input ports to be connected. Each port in the diagram is uniquely specified by the element number from Table 1 and the sequential number of its port. Let us introduce a continuous numbering (throughout the diagram) of both input and output ports according to the order of the elements in the diagram and the ports in each of them. The number set in this way will be called the diagram port number. The port involved in the connection is represented in Table 2 by both the element port number and the diagram port number.

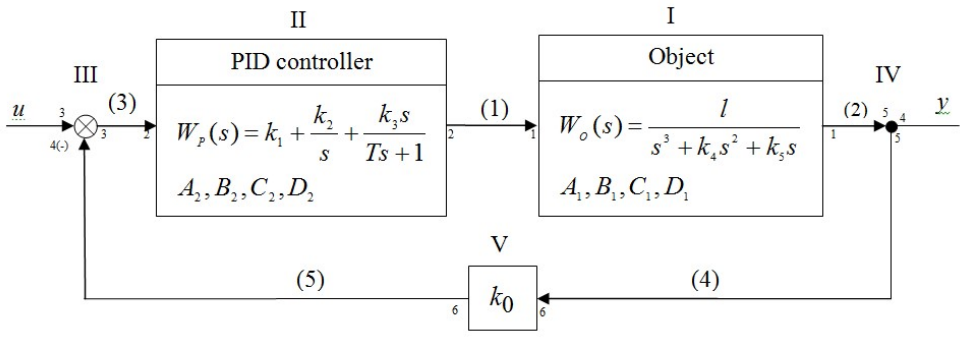

Figure 1: Example of a Linear System Block Diagram

In Figure 1, all connections are numbered according to their order in Table 2 with Arabic numerals in brackets. Ports are numbered as scheme ports in small Arabic numerals.

Table 2: The Set of Connections in the Diagram

Number in the figure	Output port			Input port		
	Element (block)	Element (block) port number	Diagram port number (*i*)	Element (block)	Element (block) port number	Diagram port number (*j*)
(1)	II	1	2	I	1	1
(2)	I	1	1	IV	1	5
(3)	III	1	3	II	1	2
(4)	IV	2	5	V	1	6
(5)	V	1	6	III	2	4

Now we will construct a mathematical model of the block diagram in the form of state-space equations. To determine the corresponding matrices, we use the proposed method.

Using Table 1, we find the initial data:

Number of structural elements of the diagram $N = 5$, the diagram state vector x of dimension $n=5$,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the input vector $u^{(0)}$ of dimension k=6, output vector $y^{(0)}$ of dimension m=6.

$A^{(0)} =$
```
0 1 0 0 0
0 0 1 0 0
0 -2 -3 0 0
0 0 0 0 0
0 0 0 0 -166.67
```
$B^{(0)} =$
```
0 0 0 0 0 0
0 0 0 0 0 0
4 0 0 0 0 0
0 -30556 0 0 0 0
0 5.0926e+06 0 0 0 0
```
$C^{(0)} =$
```
1 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```
$D^{(0)} =$
```
0 0 0 0 0 0
0 184.23 0 0 0 0
0 0 1 -1 0 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 0 1
```

The indices of the bypass matrix columns and input matrix columns correspond to the numbers of the diagram input ports (Table 2): 1, 2, 3, 4, 5, 6.

The indices of the bypass matrix rows and output matrix rows correspond to the numbers of the output ports in the diagram (Table 2): 1, 2, 3, 4, 5, 6.

Step 1.

Connection (1) is processed according to Table 2: $i-2; j-1$. These are the row and column indices from the previous step. Since $1-d_{2,1}^{(0)}=1\neq 0$, using expression (11), the model matrices at this stage of the iterative process are found to be:

$A^{(1)} =$
```
0 1 0 0 0
0 0 1 0 0
0 -2 -3 4 0
0 0 0 0 1
0 0 0 0 -166.67
```
$B^{(1)} =$
```
0 0 0 0 0
0 0 0 0 0
736.93 0 0 0 0
-30556 0 0 0 0
5.0926e+06 0 0 0 0
```
$C^{(1)} =$
```
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```
$D^{(1)} =$
```
0 0 0 0 0
0 1 -1 0 0
0 0 0 1 0
0 0 0 1 0
0 0 0 0 1
```

Indices of the columns of the bypass and input matrices correspond to the numbers of the diagram input ports: 2, 3, 4, 5, 6.

Indices of the rows of the bypass and output matrices match the numbers of the diagram output ports: 1, 3, 4, 5, 6.

Step 2.

Connection (2) is processed according to Table 2: $i=1; j=5$. Since $1-d_{1,5}^{(1)}\neq 0$, using expression (11), the model matrices at this stage of the iterative process are found to be:

$A^{(2)} =$
```
0 1 0 0 0
0 0 1 0 0
0 -2 -3 4 0
0 0 0 0 1
0 0 0 0 -166.67
```
$B^{(2)} =$
```
0 0 0 0
0 0 0 0
736.93 0 0 0
-30556 0 0 0
5.0926e+06 0 0 0
```
$C^{(2)} =$
```
0 0 0 0 0
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
```
$D^{(2)} =$
```
0 1 -1 0
0 0 0 0
0 0 0 0
0 0 0 1
```

Indices of the bypass and input matrix columns match the numbers of the diagram input ports: 2, 3, 4, 6.

Indices of the bypass and output matrix rows match the numbers of the diagram output ports: 3, 4, 5, 6.

Step 3.

Connection (3) is processed according to Table 2: $=3; j=2$. Given that $1-d_{3,2}^{(2)}\neq 0$, using expression (11), the model matrices at the third step of the iterative process are found to be:

$A^{(3)} =$
```
0 1 0 0 0
0 0 1 0 0
0 -2 -3 4 0
0 0 0 0 1
0 0 0 0 -166.67
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$B^{(3)} =$
```
0 0 0
0 0 0
736.93 -736.93 0
-30556 30556 0
5.0926e+06 -5.0926e+06 0
```
$C^{(3)} =$
```
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
```
$D^{(3)} =$
```
0 0 0
0 0 0
0 0 1
```

Indices of the bypass and input matrix columns match the numbers of the diagram input ports: 3, 4, 6.

Indices of the bypass and output matrix rows match the numbers of the diagram output ports: 4, 5, 6.

Step 4.

Connection (4) is processed according to Table 2: $i = 5; j = 6$. As $1 - d_{5,6}^{(3)} \neq 0$, using expression (11), the model matrices at this step of the iterative process are found to be:

$A^{(4)} =$
```
0 1 0 0 0
0 0 1 0 0
0 -2 -3 4 0
0 0 0 0 1
0 0 0 0 -166.67
```
$B^{(4)} =$
```
0 0
0 0
736.93 -736.93
-30556 30556
5.0926e+06 -5.0926e+06
```
$C^{(4)} =$
```
1 0 0 0 0
1 0 0 0 0
```
$D^{(4)} =$
```
0 0
0 0
```

Indices of the bypass and input matrix columns match the numbers of the diagram input ports: 3, 4.

Indices of the bypass and output matrix rows match the numbers of the diagram output ports: 4, 6.

Step 5.

Connection (4) is processed according to Table 2: $i = 6; j = 4$. Given that $1 - d_{6,4}^{(4)} \neq 0$, through expression (11), the matrices of the model at this step of the iterative process are found to be:

$A^{(5)} =$
```
0 1 0 0 0
0 0 1 0 0
-736.93 -2 -3 4 0
30556 0 0 0 1
-5.0926e+06 0 0 0 -166.67
```
$B^{(5)} =$
```
0
0
736.93
-30556
5.0926e+06
```
$C^{(5)} =$
```
1 0 0 0 0
```
$D^{(5)} =$
```
0
```

Indices of the bypass and input matrix columns correspond to the numbers of the diagram input ports: 3.

Indices of the bypass and output matrix rows correspond to the numbers of the diagram output ports: 4.

This exhausts the set of connections in Table 2 and completes the construction of the mathematical model in the form of equations of state for the block diagram shown in Figure 1. The mathematical model of the diagram in full accordance with its depiction in Figure 1 has one free input, numbered as the diagram input port 3, and one free output, numbered as the diagram output port 4, and is represented by the obtained matrices of equations of state.

The calculations were performed in a standard fixed-point and floating-point data format. The results are presented in a five-decimal form.

3.2 Subsystems in Block Diagrams

A subsystem of a block diagram is a group of elements in the scheme with all their connections. A subsystem of a block diagram is used as a single element. The concept of subsystems is one of the concepts of the metamodel of MATLAB systems, Simulink. The concept of the hierarchy of subsystems is also supported. The use of subsystems creates a hierarchical model. The subsystem is located at one level, and the elements that make up the subsystem are located at another level.

The creation and use of subsystems assume the organization of a library of subsystems with the management of user access to subsystems by setting authority parameters.

A subsystem combines functionally related elements that tend to retain the structure of their relationships and parameters during the design process. The number of displayed elements of the modeled scheme is reduced. All this makes the scheme more understandable and easy to handle.

We propose to consider a subsystem to be an element with a mathematical model in the form of equations-of-state matrices calculated by the proposed computational method.

Let us assume that the design of the system provides and defines its subsystem from the start. Then, using the proposed method, it is possible to

calculate the mathematical model of this subsystem in the state space, which can then be used as a model of a new element (subsystem-element) ready for use in this parent structural scheme and not only this one.

If the allocation and creation of a subsystem become a necessity in an already compiled complex block diagram, a part of the parental scheme is selected by means of the graphic interface and transferred to the window of a new block diagram along with all information on the elements comprising this subsystem and their connections. In principle, information on the elements and connections of the subsystem contained in the description of the parental scheme is enough to automatically calculate the mathematical model of the subsystem by the proposed method.

Further, such a subsystem can be formalized as a subsystem-element and used as such element in the original parent scheme, as well as in other schemes.

Subsystems give an opportunity to create a hierarchical model including different levels. Moving through the hierarchy of subsystems provides such features as opening a subsystem for viewing, opening a subsystem for editing, returning from an open subsystem to the parent system, and deploying a subsystem, which consists in moving the subsystem content to the parent schema and flattening the hierarchy to a single level.

The task of realizing a multilevel hierarchy of subsystems in the calculation of mathematical models for the parental scheme and all of its subsystems does not change anything in principle about the essence of the proposed computational method. The resolution of this task is defined by the conditions and features of the application of the method and lies in the area of using graphical interface design tools.

3.3 Object-Oriented Approach to Modeling of Block Diagrams

A dynamic block diagram is considered an object that has properties (elements, connection lines, and possibly others) as well as methods that determine its development (input, removal of elements and connection lines, saving the scheme, loading the scheme, etc.). Utilizing the object-oriented approach [25], we will introduce a data type for describing the scheme – the Schema class. In this class, a description of elements and links should be available. Let us introduce two more data types: Element – a class of schema elements, and Link – a class of links. A Schema object must include as its properties a set of Element objects (let us call this set Elements) and a set of Link objects (Links). The relationship between the Schema class and the Element and Link classes corresponds to the composition relation. The composition relation between the Schema class and the Element and Link classes stipulates that the Schema class object alone is responsible for creating and destroying all (and each separately) objects of the Element and Link classes, and destroying a structural scheme means destroying all its constituent elements and links.

Naturally, a link as an object of class Link does not exist by itself but connects two already existing elements – two objects of class Element. The interaction occurs between elements (objects of the Element class), and their interaction, defined by links, at the logical level is described by a "class-self-association" connection. In this case, deleting an element removes all of its connection lines with other elements of the diagram. Thus, the connection line is a property of the element, or rather a property of a port of the element, at the same time being a property of another port of the diagram element. Removing any link is provided as one of the methods of the diagram modifying the properties of two ports of its elements. Element ports can be described with a reference to an object of the Link class (for this purpose, we introduce the type Link*, connected to the Link class on the class diagram by an implementation relation), in which case an open port corresponds to a NULL value. Note that Заметим, что a property of an element must be a reference to a Link object (a variable of Link* type) and not the object itself, because two equal possessors -- the two elements connected by it -- have a claim to the same link, and, therefore, neither of them can have the prerogative to remove it as an object of Link class. The description of all input and output ports in the Element class can be organized as two arrays, InPort and OutPort, consisting of Link* type elements. In the Link class, we need to define at least four variables: pOutElm, OutNumber, pInElm, InNumber. The pOutElm variable is a reference to the object of the element associated with the beginning of the connection line, while the pInElm variable is a reference to the object of the element associated with the end of the connection line. To describe these variables, the Element* type is introduced (in the diagram of classes, Element and Element* are connected by an implementation relation), the variable of which is a reference to the Element class object. The OutNumber variable of integer type is the number of the output port of the element pointed to by the pOutElm variable. The InNumber variable of integer type is the number of the input port of the element pointed to by pInElm.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Note that the Link object properties described by the pOutElm and pInElm variables cannot be replaced directly with Element objects, because one element can have several Link objects associated with it, and none of them can have the right to remove it.

Thus, the block diagram is described by class Schema, which contains a composition (e.g., array, list) of uniform objects – elements – described by the Element class. Interactions of elements in the scheme, i.e. interaction of objects of the Element class, can be described at the logical level by self-association defined by the Link class. This interaction will correspond to the line of connection, that is, connection lines will be considered objects of class Link. Next, a composition of objects of class Link is introduced into the Schema class to define all lines of connection in the diagram. The introduced data types and their interaction at the logical level are included in the class diagram shown in Figure 2.

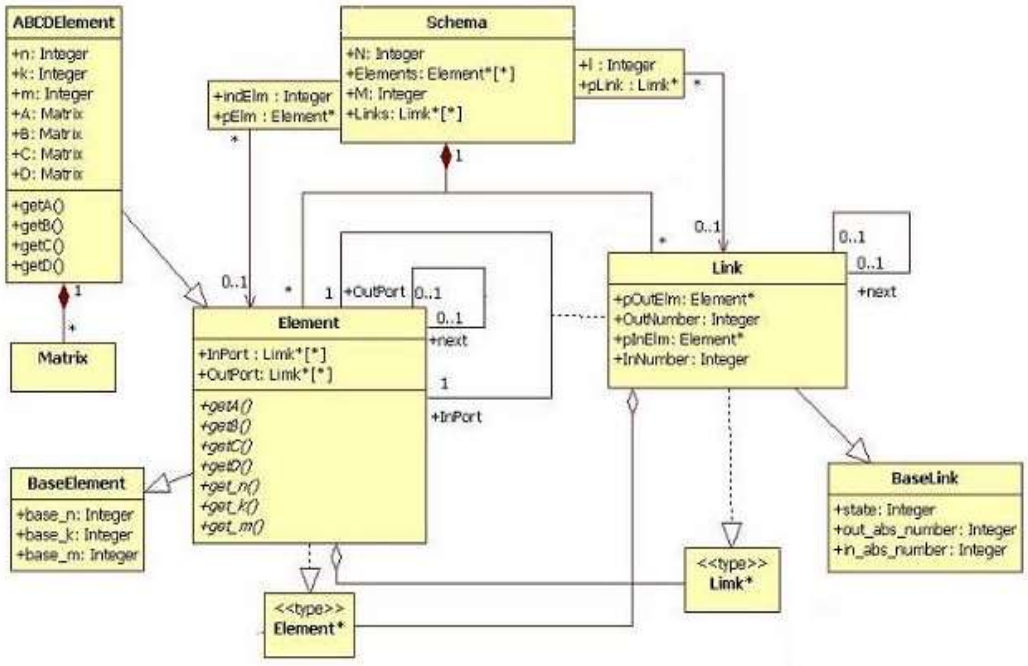

Figure 2: Class Diagram

Here we should note that the proposed realization of the association of elements in a block diagram by means of Link and Element classes is redundant with respect to the data entered. As a consequence, there is a clear and easily realized opportunity to make appropriate changes in the properties of the elements connected by a link when deleting said link and, when deleting an element, to remove all the corresponding connections to it with changes in the properties of the elements that were in conjunction with it.

At the stage of discussion of the proposed method, it is fundamental for us that access to an object of the Element class provides a complete acquisition of data of the element's mathematical model in the form of equations in state variables. However, when displaying the connectivity property of elements in the scheme, the mathematical model of the element can be taken out of the picture. Therefore, it seems reasonable to introduce a system of inherited classes to describe the essence of elements and connection lines.

The Element class describes an element as a structural unit of the diagram, which has all the necessary and only necessary properties to be attached to a scheme consisting of other similar structural units. This class is parent to the ABCDElement class, which describes the mathematical model of the element in the form of equations in state variables. To get the matrices of the element model, abstract (polymorphic) operations getA(), getB(), getC(), getD() are introduced into the Element class. These operations will be defined in the derived classes (in particular, in the ABCDElement class). Aside from the above, the Element class provides abstract operations get_n(), get_k(), get_m(), which will give the

number of states, inputs, and outputs in the Element object.

This system of inheritance will enable further consideration of schemes with elements described by different mathematical models serving as structural units. To describe such elements, we can design classes that inherit the Element class with all its properties and methods, reflecting its essence as a structural unit of the scheme.

In addition, two more classes are introduced, BaseElement and BaseLink, which are designed to give each port a unique number within the diagram. These classes have the feature that the properties of objects of these parent classes cannot be defined without their descendants. Thus, it is only after a Schema class object is created and the Elements composition (with access to objects of specific descendants) and Links composition are defined in it that the properties of the parent BaseElement and BaseLink class objects can be calculated. The values of BaseElement and BaseLink object properties are determined at the initial stage of calculating the mathematical model of the scheme. Three attributes additionally introduced in the BaseElement class are three integer variables (base_n, base_k, and base_m) intended to store the numbering offset within the scheme to recalculate the numbers of the state variables and input and output ports of a given element when it is included in the scheme. In the BaseLink class, there are two attributes introduced: two integer variables (out_abs_number and in_abs_number) to store the sequential numbers of the connected output and input ports when numbering them consecutively in the scheme (which is called an absolute number in contrast to the relative number of the same port within an element). In addition, the connection state variable, *state*, is introduced into the BaseLink class, taking values such as NOT_DONE (not done), DONE (done), PASS (skipped). The inheritance chains of classes BaseElement-Element-ABCDElement and BaseLink-Link are included in the class diagram shown in Figure 2.

4. DISCUSSION

In the theory of systems, connections can be represented by several types of graphs: linear graph, bond graph, signal flow graph, temporal causal graph (TCG), or block diagrams [26]. In a study by J. Willard et al. [27], TCG serves to identify algebraic loops and equivalent transformations to remove them.

The algebraic loop is considered in the Simulink modeling system [10,24,28]. It is noted that in the general case its presence in the dynamic system transfers the system under study from the category of ODEs to the system of DAEs. Possibilities for detecting, diagnosing, and working out algebraic loops are offered. Various response strategies are provided to the user. In the general case, when an algebraic loop is identified, Simulink uses methods for solving nonlinear algebraic equations at each simulation step to resolve it. However, the tools for solving algebraic loops are noted to have limitations.

The present article presents the solution in the form of a mathematical description used to perform a stage of construction of the model of a linear dynamic system given by a structural scheme. At each stage of connection sequence processing, the corresponding mathematical models are formed, and the presence of algebraic loops in the model is revealed. Unprocessed connections form m algebraic loops. The system of equations describing the system of algebraic loops and the condition of existence and uniqueness of its solution are obtained. Based on the obtained solution to the system of equations, a mathematical model of the block diagram is created, which, at the stage of construction, includes solvable algebraic loops. If this is not the case, the block diagram must be reconfigured. The process is constructed as a procedure for checking a mathematical condition and is related to the processing of a particular connection. The user can set the sequence of connection processing and have information about the input and output ports, the connection lines between which give rise to algebraic loops. This feature is desirable when forming high-order multiple input and output (MIMO) systems.

In the scientific literature, the problem of the presence of an algebraic (static) contour in a dynamic system does not go unnoticed. In recent years, several works have been devoted to this subject [13, 29]. Their authors use a single research scheme addressing multidimensional algebraic contours both in terms of their solvability (well-posed) and in terms of practical implementation in linear systems with compensation for the effect of drive saturation in control systems (anti-windup schemes).

The difference between the approach proposed in this article is that the issue of the solvability of a well-posed algebraic loop is solved at the stage of constructing a mathematical model of a control system, given by its block diagram. The difference is also that the article does not set the task of designing

anti-windup control and considers only linear control systems. However, assuming that the nonlinearity of the saturation type is piecewise linear, it is possible to use the proposed method to solve the problem of the solvability of a well-posed algebraic loop that occurs in a linear system with compensation for the effect of saturation of the control drives.

The difference in the approach considered in the article is that the issue of calculating a well-posed algebraic loop is not raised at the modeling stage since the resolvable loop is included in the mathematical model of the simulated linear control system.

5. CONCLUSION

Calculation of the mathematical model is an iterative process consisting of steps, each of which processes one connection represented on the block diagram by one connection line.

The proposed method is convenient for developing a software-oriented algorithm and using it to develop software for the study of complex high-order dynamic systems by classical and modern methods.

The method makes it possible to detect an algebraic loop in the block diagram, and in the general case – a system of algebraic loops. The condition of solvability of the system of algebraic loops is derived, making it possible to obtain a mathematical model of the block diagram. If the said condition is not satisfied, the block diagram is considered ill-posed due to the presence of an algebraic contour in it, which is unsolvable or does not have a unique solution.

The paper explores an illustrative numerical example of the operation of the computational method and the possibility of using this method to implement a multilevel hierarchy of subsystems in a block diagram. Mathematical models of both the parental system and all of its subsystems can be calculated by means of the proposed computational method. The practical resolution of this task is determined by the conditions and features of the application of the method and lies in the area of using graphical interface design tools.

The data types and class system needed for constructing a mathematical model of a dynamic system given by a block diagram are developed and described.

The obtained results give grounds to conclude that this study contributes to the practice of creating software systems for designing control systems.

The limitation of this study is the application of the presented method in the framework of linear dynamic systems.

REFRENCES:

[1] D. Baumann, F. Solowjow, K.H. Johansson, and S. Trimpe, *Identifying causal structure in dynamical systems*, June 6, 2020. [Online]. Available: https://arxiv.org/abs/2006.03906

[2] M. Kramer, P. Hubwieser, and T. Brinda, "A competency structure model of object-oriented programming", in *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, Mumbai, India, March 31 - April 3, 2016, pp. 1-8.

[3] The MathWorks, *Simulink user's guide*, 2019. [Online]. Available: https://www.mathworks.com/help/releases/R2019b/pdf_doc/simulink/sl_using.pdf (access date: October 12, 2020).

[4] G. Steindl, and W. Kastner, "Transforming OPC UA information models into domain-specific ontologies", in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, Victoria, BC, Canada, May 10-12, 2021, pp. 191-196.

[5] A. Varga, "Computer-aided control systems design: Introduction and historical overview", in J. Baillieul, and T. Samad (Eds.), *Encyclopedia of systems and control*. London, UK: Springer, 2014, pp. 1-7.

[6] P.M. DeRusso, A.A. Desrochers, R.J. Roy, and C.M. Close, *State variables for engineers*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[7] M. Jaskolka, "Making simulink models robust with respect to change", Doctoral dissertation, McMaster University, Hamilton, ON, Canada, 2020.

[8] A.A. Samarskii, and A. V. Gulin, *Numerical methods: Textbook, manual for universities*. Moscow, USSR: Nauka, 1989, 432 p. [in Russian].

[9] R. Sinha, C.J. Paredis, V.C. Liang, and P.K. Khosla, "Modeling and simulation methods for design of engineering systems", *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 1, 2001, pp. 84-91.

[10] M. Jaskolka, V. Pantelic, A. Wassyng, and M. Lawford, *Supporting modularity in simulink models*, July 20, 2020. [Online]. Available: https://arxiv.org/abs/2007.10120

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[11] P.J. Mosterman, and J.E. Ciolfi, "Using interleaved execution to resolve cyclic dependencies in time-based block diagrams", in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601)*, Nassau, Bahamas, December 14-17, 2004, pp. 4057-4062. https://doi.org/10.1109/CDC.2004.1429387

[12] A. Zhu, P. Pauwels, and B. De Vries, "Smart component-oriented method of construction robot coordination for prefabricated housing", *Automation in Construction*, Vol. 129, 2021, Art. No. 103778.

[13] A.A. Adegbege, and R.M. Levenson, "Dynamic multivariable algebraic loop solver for input-constrained control", *IEEE Transactions on Automatic Control*, Vol. 67, No. 6, 2021, pp. 3051-3058.

[14] U. Knauer, and K. Knauer, *Algebraic graph theory: Morphisms, monoids and matrices*. Berlin, Germany; Boston, MA, USA: de Gruyter, 2019.

[15] V.W. Noonburg, *Differential equations: From calculus to dynamical systems*, Vol. 43. Providence, Rhode Island, USA: American Mathematical Soc., 2020.

[16] W.Y. Yang, W. Cao, J. Kim, K.W. Park, H.-H. Park, J. Juong, J.-S. Ro, H.L. Lee, C.H. Hong, and T. Im, *Applied numerical methods using MATLAB*. Hoboken, New Jersey, USA: John Wiley & Sons, 2020. http://dx.doi.org/10.5860/choice.43-1615

[17] L.F. Shampine, *Numerical solution of ordinary differential equations*. New York, NY, USA: Routledge, 2018.

[18] N.M. Karayanakis, *Advanced system modelling and simulation with block diagram languages*. New York, NY, USA: CRC Press, 1995.

[19] S. Campbell, A. Ilchmann, V. Mehrmann, and T. Reis (Eds.), *Applications of differential-algebraic equations: Examples and benchmarks*. Cham, Switzerland: Springer International Publishing, 2019.

[20] P.J. Mosterman, E.J. Manders, and G. Biswas, "Qualitative dynamic behavior of physical system models with algebraic loops", in *Eleventh international workshop on principles of diagnosis*, Morelia, Mexico, June 8-10, 2000, pp. 155-162.

[21] E.E. Mitchell, and J.S. Gauthier, "Advanced continuous simulation language (ACSL)", *Simulation*, Vol. 26, No. 3, 1976, pp. 72-78.

[22] F. Golnaraghi, and B.C. Kuo, *Automatic control systems*. New York, NY, USA: McGraw-Hill Education, 2017.

[23] A.C. Grace, "SIMULAB, An integrated environment for simulation and control", in *1991 American Control Conference*, Boston, MA, USA, June 26-28, 1991, pp. 1015-1020.

[24] A.A. Samarskii, and A.P. Mikhailov, *Matematicheskoe modelirovanie* [Mathematical Models]. Moscow, Russia: Fizmatlit Publ., 2006.

[25] P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A novel approach for integrating IEC 61131-3 engineering and execution into IEC 61499", *IEEE Transactions on Industrial Informatics*, Vol. 17, No. 8, 2020, pp. 5411-5418.

[26] The MathWorks, *MATLAB*, 2020. [Online]. Available: https://www:mathworks:com/products/matlab: html (access date: October 12, 2020).

[27] J. Willard, X., Jia, S. Xu, M. Steinbach, and V. Kumar, *Integrating physics-based modeling with machine learning: A survey*, March 10, 2020. [Online]. Available: https://arxiv.org/abs/2003.04919v1

[28] S.C. Shah, M.A. Floyd, and L.L. Lehman, "MATRIXX: Control design and model building CAE capability", in M. Jamshidi, and C.J. Herget (Eds.), *Computer-aided control systems engineering*. Amsterdam, Netherlands: Elsevier Science Publishers, 1985, pp. 181-207.

[29] A.A. Adegbege, and W.P. Heath, "A framework for multivariable algebraic loops in linear anti-windup implementations", *Automatica*, Vol. 83, 2017, pp. 81-90.