Proceedings of the 2006 IEEE
Conference on Computer Aided Control Systems Design
Munich, Germany, October 4-6, 2006

Java Simulation Platform for Control System based on Block Diagram

Masanobu Koga, Yusuke Tsutsui, and Jun Yabuuchi

Abstract— **This paper proposes a new efficient method for modeling and simulation of control systems using an adjacency matrix in graph theory. A Java simulation platform, which has been developed based on the proposed method, is introduced. It provides an interactive graphical environment for modeling and simulation of control systems.**

I. INTRODUCTION

In recent years, the control system continues to be larger and more complicated since the size of the plant becomes larger and more advanced controllers are used. Through the closed-form state equation and the output equation of the whole system may be obtained from the state equations and the output equations of the sub-systems, it takes very long time for simulation based on the closed-form equation since they are terribly complicated.

In this paper, a new efficient method for modeling and simulation of control systems is proposed using an adjacency matrix in graph theory. A new simulation platform, Jamox (Java Agile MOdeling Tool for Control System), for modeling and simulation of the control systems has been developed using the proposed method. By drawing a block diagram in Jamox like using Simulink, we can easily create a model of the system and perform the simulation. Jamox supports continuous-time systems, discrete-time systems with multi-rate, and sampled-data systems.

II. ADJACENCY MATRIX REPRESENTATION

The adjacency matrix is one of the representations of a graph and the (i, j) element contains the adjacency information between node i and node j. When we regard an arrow of the block diagram as a node of graph, and a block and a summing point of the block diagram as an edge of graph, we can consider a block diagram as a weighted directed graph. The corresponding graph of a block diagram is a simple graph since the graph has no multi-edge and no self-loop. Therefore the data of a block digram can be represented as an adjacency matrix.

For example, the adjacency matrix of the block diagram shown in Fig.1 is given by (1), where S_i is an operator which represents input-output characteristic of the system, I is an unit operator which transmits the input without modification, $-I$ is an operator whose output is opposite in sign to the input. When a system is linear, S_i is a transfer function matrix. The adjacency matrix can be used as an unified form to represent general control systems. Furthermore, the structure of the adjacency matrix is useful for the data operation in computer.

M. Koga is with the Department of Systems Innovation and Informatics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka, Fukuoka, Japan koga@ces.kyutech.ac.jp

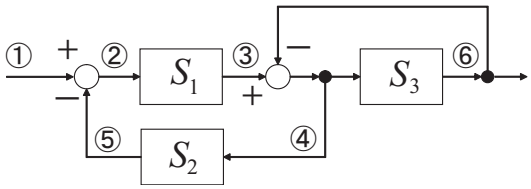

Fig. 1. Block Diagram of Control System

$$J = \begin{array}{c} \begin{array}{cccccc} ① & ② & ③ & ④ & ⑤ & ⑥ \end{array} \\ \begin{bmatrix} & I & & & & \\ & & S_1 & & & \\ & & & I & & \\ & & & & S_2 & S_3 \\ & -I & & & & \\ & & & -I & & \end{bmatrix} \\ \begin{array}{c} ① \\ ② \\ ③ \\ ④ \\ ⑤ \\ ⑥ \end{array} \end{array} \tag{1}$$

A. Adjacency Matrix of Linear System

Figure 2 shows the block diagram of a proper linear system, where $\frac{I}{q}$ is a diagonal matrix with integrators as diagonal elements for continuous-time systems and is a diagonal matrix with delay operators as diagonal elements for discrete-time systems. And I is an unit matrix.

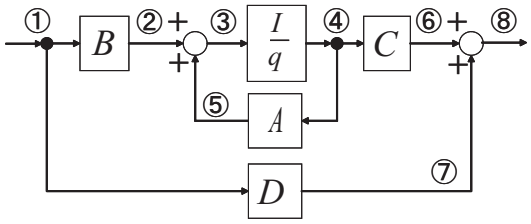

Fig. 2. Block diagram of a Linear System

The adjacency matrix of a linear system is given as follows.

$$\begin{array}{c} \begin{array}{cccccccc} ① & ② & ③ & ④ & ⑤ & ⑥ & ⑦ & ⑧ \end{array} \\ \begin{bmatrix} & B & & & & & D & \\ & & I & & & & & \\ & & & \frac{I}{q} & & & & \\ & & & & A & C & & \\ & & I & & & & & \\ & & & & & & & I \\ & & & & & & & I \\ & & & & & & & \end{bmatrix} \\ \begin{array}{c} ① \\ ② \\ ③ \\ ④ \\ ⑤ \\ ⑥ \\ ⑦ \\ ⑧ \end{array} \end{array}$$

In this paper, we define a series-edge with constant weights as a *constant series-edge*. It is possible to contract a constant series-edge to a edge whose weight is the product of the weights of the edges in the original series-edge. Contracting all the constant series-edges in the graph which corresponds to a linear system gives the following matrix.

①	③	④	⑧	
	B		D	①
		$\frac{I}{q}$		③
	A		C	④
				⑧

Exchanging node 3 and node 4 (exchange rows and columns respectively), we get the next definition.

Definition 1: The adjacency matrix of a proper linear system is given by

$$J = \begin{bmatrix} & & B & D \\ & & A & C \\ & \frac{I}{q} & & \\ & & & \end{bmatrix} \tag{2}$$

III. State Space Representation of Combined Systems

A. Feedback connection case

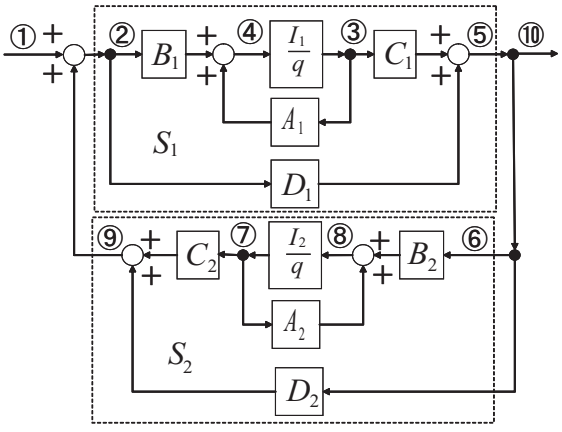

Fig. 3. Block Diagram of a Feedback Connected System

The adjacency matrix of a feedback connected system, which contains two linear systems, shown in Fig. 3 is given by

①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	
	I									①
			B_1	D_1						②
			A_1	C_1						③
		$\frac{I_1}{q}$								④
					I				I	⑤
							B_2	D_2		⑥
							A_2	C_2		⑦
						$\frac{I_2}{q}$				⑧
	I									⑨
										⑩

.

Contract some constant series-edges and exchange rows and columns appropriately, then we have

①	③	⑦	⑨	④	⑧	⑩	
			D_2D_1	B_1	B_2D_1	D_1	①
			D_2C_1	A_1	B_2C_1	C_1	③
			C_2		A_2		⑦
			D_2D_1	B_1	B_2D_1	D_1	⑨
	$\frac{I_1}{q}$						④
		$\frac{I_2}{q}$					⑧
							⑩

(3)

When neither of two linear systems are strictly proper, i.e., $D_1 \neq 0$ and $D_2 \neq 0$, the matrix has a nonzero diagonal element D_2D_1 at (4,4), and there exists a self-loop at node 9.

Let v_i be the value of node i, then we have

$$v_9 = D_2D_1v_1 + D_2C_1v_3 + C_2v_7 + D_2D_1v_9$$

, and

$$(I - D_2D_1)v_9 = D_2D_1v_1 + D_2C_1v_3 + C_2v_7$$

If the well-posedness condition of the combined system is satisfied, then $(I - D_2D_1)$ is nonsingular and we have

$$v_9 = P(D_2D_1v_1 + D_2C_1v_3 + C_2v_7) \tag{4}$$

, where

$$P = (I - D_2D_1)^{-1}$$

From Eq. (3) and Eq. (4), we have

①	③	⑦	⑨	④	⑧	⑩	
			PD_2D_1	B_1	B_2D_1	D_1	①
			PD_2C_1	A_1	B_2C_1	C_1	③
			PC_2		A_2		⑦
				B_1	B_2D_1	D_1	⑨
	$\frac{I_1}{q}$						④
		$\frac{I_2}{q}$					⑧
							⑩

Hence, the self-loop can be removed when the well-posedness condition of the combined system holds. Additional contraction of constant series-edges and exchange of

nodes leads to

①	③	⑦	④	⑧	⑩	
			B_1+ $B_1PD_2D_1$	B_2D_1+ $B_2D_1PD_2D_1$	D_1+ $D_1PD_2D_1$	①
			A_1+ $B_1PD_2C_1$	B_2C_1+ $B_2D_1PD_2C_1$	C_1+ $D_1PD_2C_1$	③
			B_1PC_2	A_2+ $B_2D_1PC_2$	D_1PC_2	⑦
	$\frac{I_1}{q}$					④
		$\frac{I_2}{q}$				⑧
						⑩

(5)

Since the matrix in (5) has the structure of the adjacency matrix of a linear system defined in (2), the coefficient matrices of the state space representation of the combined linear system are given by

$$
\begin{aligned}
A &= \begin{bmatrix} A_1 + B_1PD_2C_1 & B_1PC_2 \\ B_2C_1 + B_2D_1PD_2C_1 & A_2 + B_2D_1PC_2 \end{bmatrix} \\
B &= \begin{bmatrix} B_1 + B_1PD_2D_1 \\ B_2D_1 + B_2D_1PD_2D_1 \end{bmatrix} \\
C &= \begin{bmatrix} C_1 + D_1PD_2C_1 & D_1PC_2 \end{bmatrix} \\
D &= D_1 + D_1PD_2D_1
\end{aligned}
$$

B. General Connection Case

Proposition 1: Suppose that all the systems in a block diagram are linear and proper, and the well-posedness condition of the system are satisfied. If the input node and the output node are specified, then the state space representation of the whole system are obtained by algorithm 1. The algorithm requires only simple calculation with real matrices which are the coefficient matrices of each block. And the order of the system matrix is equal to the sum of the order of each dynamic system block.

Algorithm 1:

1) Get the adjacency matrix of the whole system using the adjacency matrices of each linear system.
2) Repeat the following steps while there exist any constant series-edges.
   a) If there is no self-loop (all diagonal elements are 0), contract constant series-edge.
   b) If there exist any self-loops (let $J_{ii} \neq 0$), multiply the inverse of $P_i = (I - J_{ii})$ to the elements in the i-th column from left and set 0 to the (i, i) element. It is found that the well-posedness condition is not satisfied when P_i is singular.
3) Exchange nodes (exchange rows and columns respectively) such that the integrators (delay operators) get lined up in diagonal in the left lower part of the adjacency matrix.
4) Get the coefficient matrices of the state space representation from the right upper part of the adjacency matrix according to the structure defined in (2).

C. Method used in Simulink/Matlab

In this section, we compare the proposed method with one used in Simulink/Matlab. Simulink/Matlab provides the functions to get the linear model of the simple combined systems which include only serial, parallel, and feedback connections. As for the systems with general connections, they provide the method to get the linearized approximate model of the combined systems. They give the following two algorithms to get the linearized approximate model.

1) numerical-perturbation linearization
2) block-by-block analytic linearization

The former linearizes the whole system by numerically perturbing the system's inputs and states about the operating point and measuring the response to this perturbation. The state-space matrices A, B, C, and D of the linearized model is obtained as the Jacobians of the system. This algorithm is used in functions linmod and linmod2. It was the only linearization method available with old Simulink prior to version 3.0. There are several disadvantages with this method:

(a) The resulting linearized model relies on the size of the perturbations. We have limited control over the perturbation levels for each block.
(b) It is sensitive to scaling issues (models with large and small output signal values)

The linmod2 function tries to balance the trade-off between round-off error (caused by small perturbations due to finite precision mathematics), and truncation error (caused by large perturbation levels, which invalidate the piecewise linear approximation). Hence the first disadvantage is diminished by linmod2.

The latter linearizes the blocks individually and then combines the results to produce the linearization of the whole system. If the blocks contain analytic Jacobians, these information is used instead of numerically perturbing the blocks. The second disadvantage is solved by this method. However, there is no statements how to combine the linearized block to get the linearzation of the whole system.

When all the blocks in a block diagram are linear, the latter is equivalent to the proposed method in terms of using the linear model of each block. However, it is likely that the method uses the transfer functions to combine the linearized blocks and applies the realization technique.

IV. Algebraic Loops

An algebraic loop is formed when two or more blocks with direct feedthrough (the output of the block at time t is a function of the input at time t) form a feedback loop. The basic problem with algebraic loops is that the output y at time t is a function of itself and we have to solve the algebraic equation in order to get the output y at each time step. When two or more algebraic loops are connected, the corresponding algebraic equations should be solved simultaneously. In this paper, we propose a new efficient algorithm, which detects the algebraic loops in a given block diagram and decides the simultaneous algebraic equations with minimum unknowns, to calculate the output of the system which contains algebraic loops.

A. Output of System with Algebraic Loops

The algorithm to calculate the output of the system with algebraic loops at time t is given below.

Algorithm 2:

1) Get the output of the blocks without feedthrough.
2) Get the output of the blocks with feedthrough as many as possible by using the results obtained in the previous step.
3) Remove the blocks (edges) whose output are determined.
4) From remaining blocks pick out groups such that the member of each group are directly connected, and make an algebraic equation for each group and solve it as follows.
   a) Select the systems such that the outputs of the unselected systems are determined from their outputs.
   b) Solve the algebraic equation about the output of the selected system by Newton's method.
   c) Get the outputs of the unselected systems by using the results obtained in the previous step.

B. Local Maximum Cycle

Each group in step 4 of algorithm 2 forms a local maximum cycle in the graph. This section gives an algorithm to seek the local maximum cyles.

Consider a matrix K whose elements are 1 at diagonal and the position where the elements of the adjacency matrix are non-zero, and are 0 at other positions. This matrix is called a connection matrix and the boolean algebra

$$x + y := \max(x, y), \qquad x \times y := \min(x, y)$$

is used for the operation of the elements. Let n be the number of node of the graph. When the (i, j) element of the reachable matrix :

$$R := K + K^2 + \cdots + K^{n-1}$$

is 1, there exists a path from node i to node j. Multiplying R and its transpose R^T element-wise defines a matrix

$$\bar{R} := R \cap R^T$$

The (i, j) element of $\bar{R}$ is 1 if and only if both (i, j) and (j, i) element of R is 1. It implies that there is a cycle between node i and node j since there exist the pathes from node i to node j and node j to node i. Moreover when there are any nonzero elements in the i-th row in $\bar{R}$, there is a cycle including those nodes corresponding the column number and node i which is the local maximum cycle to which node i belongs.

C. Algebraic equation with Miminum Unknowns

At step 4(a) in algorithm 2 we only have to select the systems (edges) such that all the algebraic loops diminish by removing the edges. In other words we should select the systems which correspond to the edges which are removed from the graph to produce a spanning tree. In order to promote the efficiency of the calculation of step 4(b) we propose a method to select the systems such that the algebraic equation has minimum unknowns.

Theorem 1: Consider a graph which has the same connections (nodes and edges) as the given block diagram and whose edges have the weights of reciprocal of the number of output of each corresponding system. Make an algebraic equation about the outputs of the systems which correspond to the edges which are removed from the graph to produde a miminum spanning tree. Then it is the algebraic equation with miminum unknowns among the equations which is solved to get the outputs of the whole system.

Proof: Since a miminum spanning tree is a spanning tree with minimum sum of weights, the sum of the weights of the removed edges is maximum. Since the weight is a reciprocal of the number of the system the sum of the number of the outputs of the removed systems is minimum. Therefore the algebraic equation about the outpus of the removed systems has minimum unknowns. ■

A minimum spanning tree is given by J. B. Kruskal's algorithm [1].

Algorithm 3 (J.B.Kuruskal):

1) Pick up a edge with miminum weight.
2) Pcik up a edge with miminum weight among the remaining edges such that there exist no cycles.
3) Repeat the previous step until the all edges are checked.

D. Eliminating Algebraic Loops

When all the systems in an algebraic loop are linear, the algebraic loop can be eliminated by applying step 1 and step 2 of algorithm 1. By elimination of the algebraic loops, the efficiency of calculation can be promoted greatly.

E. Methods used in Simulink

This section compares the proposed method for the simulation of the system with algebraic loops with one used in Simulink. When a model contains an algebraic loop, Simulink calls a loop solving routine at each time step. The loop solver performs iterations to determine the solution of the equation which contains the signals in the loop as the unknown variables. To solve an algebraic equation, the Simulink loop solver uses Newton's method with numerical Jacobians of partial derivatives. The number of unknown variables of the equation is the sum of degreees of the signals in the loop. That is the reason why it takes longer time for the solver to converge or the residual of the solution is larger when the number of blocks in the loop increases.

On the other hand, the proposed method performs iterations of Newton's method to determine the solution of the equation with mimimum unknowns. Since the equation contains miminum unknowns, the solver requires small number of iterations to converge and the numerical Jacobians which is evaluated at each iteration can be calculated quickly.

V. Java Simulation Platform

We have developed a new simulation platform, Jamox (Java Agile MOdeling Tool for Control System), for modeling and simulation of control systems based on the method described above. Jamox has been developed in Java which is platform independent and provides an interactive graphical environment where we can create a model of the system by using interactive graphical editor and obtain the response of the system with desired simulation setting. It provides the facility to get the time responses for the specified inputs and the frequency responses between the specified inputs and outputs if all the blocks are linear system. Figure 4 shows a screen shot of Jamox. A block diagram is created by draging and drop blocks from the block library to the canvas and connecting each block.

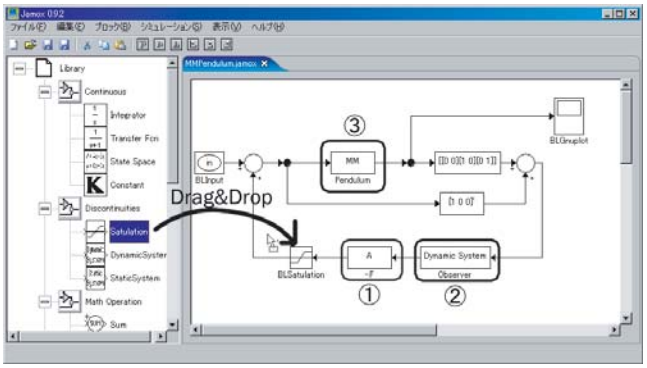

Fig. 4. Screen shot of Jamox

A. Modeling in Jamox

Block ① in Fig. 4 is a constant matrix block which is one of the built-in blocks and block ② is a user-defined block which represents a dynamic system or static system and is bind to a Java class which extends *DynamicSystem* class or *StaticSystem* class. It is possible to create reusable blocks efficiently by exploit the inheritance hierarchy of classes which represents the classification of control systems. NFC (Numerical Foundation Classes), which is a class library for numerical computation in Java, [2], is provided for creating user-defined blocks.

B. Simulation in Jamox

In order to obtain the time reponse of the system, Jamox uses several solvers which implement *OrdinaryDifferential-Solver* interface in NFC, such as RungeKutta class. It is possible to incorporate a new algorithm which implements *OrdinaryDifferentialSolver* interface. If there an algebraic loop in the system, Jamox uses some algebraic solvers which implement *NonLinearEquationSolver* interface in NFC, such as NewtonRaphson class [3].

VI. Summary

In this paper, a new efficient method for modeling and simulation of control systems was proposed using an adjacency matrix in graph theory. A new platform-independent simulation platform for modeling and simulation of the conrol systems has been introduced.

References

[1] Reinhard Diestel. *Graph Theory*. Springer-Verlag, 1997.

[2] Masanobu Koga and Takeshi Matsuki. Os-neutral numerical foundation class library and its application to control system design (in japanese). *The 3rd symposium of SICE control division*, pp. 725–728, 2003.

[3] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Veterling. *Numerical Recipes in C (The Art of Scientific Computing)*. Cambridge University Press, 1988.