International Journal of Medical, Medicine and Health Sciences
ISSN: 2517-9969
Vol:10, No:3, 2016

Acausal and Causal Model Construction with FEM Approach Using Modelica

Oke Oktavianty, Tadayuki Kyoutani, Shigeyuki Haruyama, Junji Kaneko, Ken Kaminishi

Abstract**—**Modelica has many advantages and it is very useful in modeling and simulation especially for the multi-domain with a complex technical system. However, the big obstacle for a beginner is to understand the basic concept and to build a new system model for a real system. In order to understand how to solve the simple circuit model by hand translation and to get a better understanding of how modelica works, we provide a detailed explanation about solver ordering system in horizontal and vertical sorting and make some proposals for improvement. In this study, some difficulties in using modelica software with the original concept and the comparison with Finite Element Method (FEM) approach is discussed. We also present our textual modeling approach using FEM concept for acausal and causal model construction. Furthermore, simulation results are provided that demonstrate the comparison between using textual modeling with original coding in modelica and FEM concept.

Keywords**—**FEM, acausal model, modelica, horizontal and vertical sorting.

I. Introduction

MODELICA is an object-oriented equation-based programming language with the ability for defining model libraries. Its re-usable components and complex application involving parts from several application domains make it powerful [1], [2]. Many advantages of modelica are mentioned by some researchers such as:

– Interface is user-friendly, simple and efficient [3].
– It is well suited for multi domain modeling of large, complex and heterogeneous technical system [4].
– A declarative mathematical description in modelica has simplified further analysis and the code has become concise and easier to change without introducing error [1].
– No particular variable needs to be solved for manually [5], etc.

Modelica is open source software which allows additional tools and algorithm development. It is especially appropriate for academic users [6]. We can easily and freely download the software and guidance book in an electronic self-teaching material called Dr. Modelica [2]. This book teaches about basic concepts of modeling and simulation. It is very good for people who are familiar with basic programming concept to introduce the modelica language. The modelica's book has goals to be a useful textbook for introducing modeling and simulation. This modelica's book also wants to demonstrate modeling example from some application areas [2]. There are some difficulties in understand the book especially in translation process. There is no clear explanation about the solver ordering system. We propose the solver ordering system in modelica. We provide the explanation for solving the algebraic transformation and sorting procedure.

The reasons of most people not using modelica are discussed. The reasons are most people that mainly work on control system design think that using another software is a right and sufficient choice, so no need to switch to modelica. The other reason is, it is much easier to detect and solve numerical problems (example: with Simulink). In addition, even though modelica should be a very appropriate tool, they found that it is difficult to understand. This article also discusses the particular danger when using the sophisticated library that occasionally makes users forget some basic modeling principle [6]. Hopefully, with this paper, we can refresh our knowledge in basic modeling principle.

Some difficulties that can be found in understanding modelica concept is discussed and compared with FEM approach. We present an approach with FEM concept in textual modeling using modelica. Many engineers are familiar with this method in the modeling and simulation of advanced engineering systems [7]. FEM has developed into a key, indispensable technology in various fields. With this study, we expect the engineer who get used with causal model construction has the ability to make acausal model construction by using modelica. We want to show that using FEM approach, it is easier for beginner to make textual modeling and it can also be used with modelica software. Comparison between original concept and FEM approach using acausal and causal model is discussed with simulation results.

II. Hand Translation of Simple Circuit Model with Solver Ordering System

On this section, we would like to propose some additional explanation about give detail steps for solving the algebraic transformation and sorting procedure. We also propose some revision for making it easier to understand, how to extract the equation from simple circuit model and compare the modelica concept with FEM. In modelica, a process of translation and execution of model sketched in Fig 1. Modelica model is changed to flat model by translator then using horizontal and vertical sorting as the analyzer. The result of sorted equation is

Oke Oktavianty is with Department of Mechanical Engineering, Yamaguchi University, Japan and lecturer at Industrial Engineering Department, University of Brawijaya, Indonesia (e-mail: u504wc@yamaguchi-u.ac.jp).

Shigeyuki Haruyama, Junji Kaneko and Ken Kaminishi are with graduate School of Innovation & Technology Management, Yamaguchi University, Ube, Japan (e-mail: haruyama@yamaguchi-u.ac.jp, jkaneko@yamaguchi-u.ac.jp).

Tadayuki Kyoutani is with Graduate School of Science and Technology, Department of Mechanical Engineering, Yamaguchi University, Ube, Japan.

optimized by state-space representation. Next stage is Code generator and C compiler and the final is the simulation. For modelica concept, simple circuit model with explicitly labeled connection nodes N1, N2, N3, and N4 and wires 1 to 7 shown in Fig. 2.

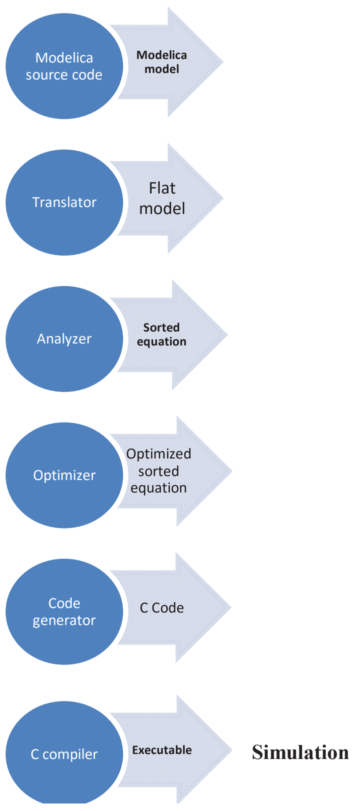


Fig. 1 Stages of translating and executing a modelica model

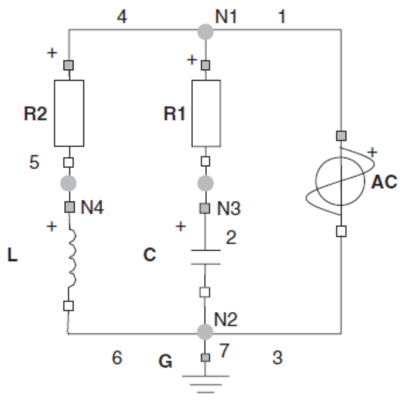


Fig. 2 Simple circuit model with explicitly labeled connection for Modelica Concept

Depending on whether the variables in connected connector using flow prefix or nonflow (default), there are two types of coupling in modelica:

- Equality coupling for nonflow variables according to Kirchoff's first Law.
- Sum-to-zero coupling for flow variables according to Kirchoff's current Law.

It is difficult for a student to understand the way to extract the equation without any explanation about ordering system and get the extracted result. For making it easier to understand, we make the ordering system to provide variable extracting steps. For modeling a circuit, we must transform the implicit differential algebraic system of equations (DAE system) become explicitly formulated algebraic and differential equations. There are 3 following steps or processes:

1. Horizontal sorting
2. Vertical sorting
3. State space representation

Rules for horizontal sorting:

- The state variables (variables that appear in differentiated form) may be assumed as known variable.
- Equations that contain only one unknown must solve for first. The solved variables then become known variable.
- Solve unknown variable one by one until all variable become known variable.
- Variables that show up in only one equation must be solved for using that equation.
- All rules may be used recursively.

Rules for Vertical Sorting:

We can do the vertical sorting for the equations that have become assignment such that no variable is being used before defined.

The first step, we assume that C.v and L.i as a known variable (temporarily), and algebraically extracting the variables in the vector as (1) and (2):

$$C.i = C.C * der(C.v) \tag{1}$$

$$L.v = L.L * der(L.i) \tag{2}$$

A. Horizontal Sorting

An example for solver ordering system shown in Table I.

Assume that C.v and L.i is available at t = 0 when we start the simulation (t as input and L.i, C.v as state variable). Our horizontal sorting steps are as follow:

- Horizontal sorting is conducted by input (time) and an initial value of state variable (L.i and C.v).
- Next target is unknown variable (C.i, L.v), are the variables in the State-space representation.
- We may divide the Horizontal Sort using a middle variable (R2.v, R1.v, R1.p.v) as needed.
- Finally reverse sort and arrange it as an expression chain from the target unknown variable (C.i, L.v).

1) C.i = C.p.i
2) 0 = R1.n.i + C.p.i $\rightarrow$ C.p.i = - R1.n.i
3) 0 = R1.p.i + R1.n.i $\rightarrow$ - R1.n.i = R1.p.i
4) R1.i = R1.p.i $\rightarrow$ R1.p.i = R1.i
5) R1.v = R1.R*R1.$\rightarrow$ R1.i = R1.v / R1.R

Then C.i = R1.v / R1.R

TABLE I
AN EXAMPLE FOR HORIZONTAL SORTING ORDER FOR C.I

1)C.i= C.p.i = -R1.n.i = R1.p.i =R1.i =R1.v/R1.R (Reverse Horizontal sort)						
AC	0	= AC.p.i + AC.n.i	**L**	0	= L.p.i + L.n.i	
	AC.v	= AC.p.v – AC.n.v		L.v	= L.p.v – L.n.v	
	AC.i	= AC.p.i		L.i	= L.p.i	
	AC.v	= AC. VA*sin (2*AC.PI*AC.f * time);		L.v	= L.L*der(L.i)	
R1	0	= R1.p.i + R1.n.i③	**G**	G.p.v = 0		
	R1.v	= R1.p.v – R1.n.v				
	R1.i	= R1.p.i ④				
	R1.v	= R1.R*R1.i ⑤				
R2	0	= R2.p.i + R2.n.i	**wires**	R1.p.v	= AC.p.v	// wire 1
	R2.v	= R2.p.v – R2.n.v		C.p.v	= R1.n.v	// wire 2
	R2.i	= R2.p.i		AC.n.v	= C.n.v	// wire 3
	R2.v	= R2.R*R2.i		R2.p.v	= R1.p.v	// wire 4
				L.p.v	= R2.n.v	// wire 5
				L.n.v	= C.n.v	// wire 6
				G.p.v	= AC.n.v	// wire 7
C	0	= C.p.i + C.n.i	**Flow**	0	= AC.p.i + R1.p.i + R2.p.i	// N1
	C.v	= C.p.v + C.n.v	**at**	0	= C.n.i + G.p.i + AC.n.i + L.n.i	// N2
	C.i	= C.p.i ①	**node**	0	= R1.n.i + C.p.i ②	// N3
	C.i	= C.C*der(C.v)		0	= R2.n.i + L.p.i	// N4

Total: 32 equations

Result of horizontal sort using a middle variable:

$$C.i= R1.v/R1.R \tag{3}$$

$$R1.v= R1.p.v - R1.n.v = R1.p.v - C.v \tag{4}$$

$$R1.p.v= AC.VA*sin(2*AC.f*AC.PI*t) \tag{5}$$

$$L.v = L.p.v - L.n.v = R1.p.v - R2.v \tag{6}$$

$$R2.v=R2.R*L.i \tag{7}$$

B. Vertical Sorting

Next step is vertical sorting. The sorting order is shown in Fig. 3. The result of extracted variable is shown in Table II. We make a little change to the table to make it easier to understand. The revised version is shown in Table III.

TABLE II
VARIABLE EXTRACTED FROM SIMPLE CIRCUIT MODEL FOR MODELICA

R1.p.i	**R1.n.i**	**R1.p.v**	**R1.n.v**	**R1.v**
R1.i	R2.p.i	R2.n.i	R2.p.v	R2.n.v
R2.v	R2.i	C.p.i	C.n.i	C.p.v
C.n.v	C.v	C.i	L.p.i	L.n.i
L.p.v	L.n.v	L.v	L.i	AC.p.i
AC.n.i	AC.p.v	AC.n.v	AC.v	AC.i
G.p.i	G.p.v			

Total 32 variables

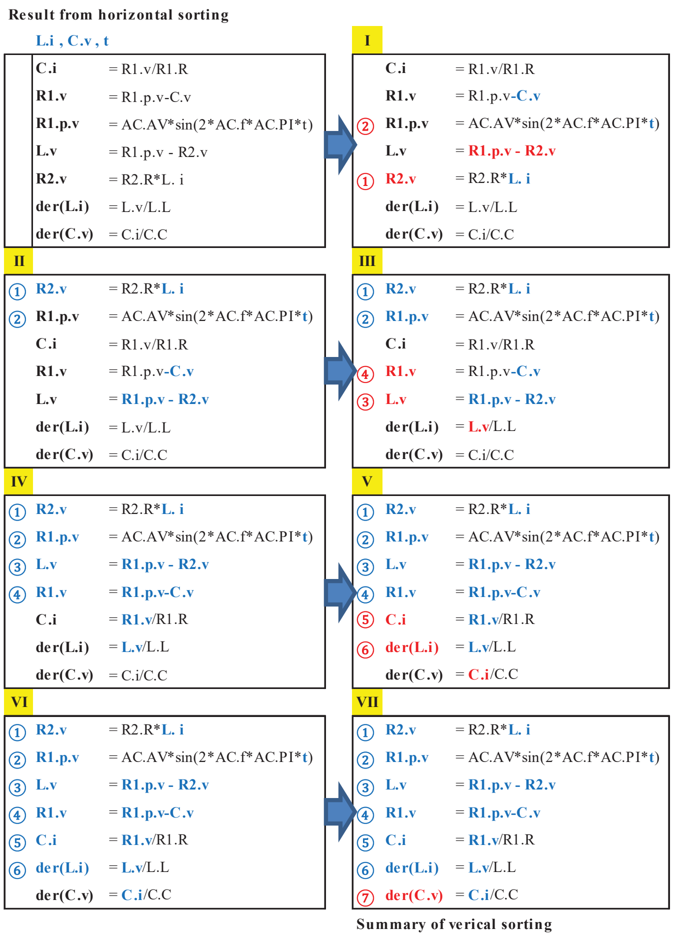


Fig. 3 Vertical sorting order

The equation then converted to Block Lower Triangular (BLT) as shown on Fig. 4.

$$\begin{cases} R2.v = R2.R * L.i \\ R1.p.v = AC.VA * \sin(2 * AC.f * AC.PI * time) \\ L.v = R1.p.v - R2.v \\ R1.v = R1.p.v - C.v \\ C.i = R1.v / R1.R \\ der(L.i) = L.v / L.L \\ der(C.v) = C.i / C.C \end{cases} = \begin{array}{ccccccc} R2.v & R1.p.v & L.v & R1.v & C.i & L.i & C.v \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}$$

Fig. 4 Block Lower Triangular (BLT) form of the simple circuit model example

TABLE III
REVISED VERSION (PROPOSAL) OF VARIABLE EXTRACTED TABLE FROM SIMPLE CIRCUIT MODEL

Total 32 variables		connector				In component	
State variable: L.i ,C.v		Pin.p		Pin.n		flow	nonflow
component	R1	R1.p.i	R1.p.v	R1.n.i	R1.n.v	R1.i	R1.v
	R2	R2.p.i	R2.p.v	R2.n.i	R2.n.v	R2.i	R2.v
	C	C.p.i	C.p.v	C.n.i	C.n.v	C.i	C.v
	L	L.p.i	L.p.v	L.n.i	L.n.v	L.i	L.v
	AC	AC.p.i	AC.p.v	AC.n.i	AC.n.v	AC.i	AC.v
	G	G.p.i	G.p.v				

There is small typing mistake in Ci row in Fig. 4. Therefore, we propose small revision on BLT. The revision is shown in Table IV.

Matrix of vertical sorting result is:

$$\begin{Bmatrix} R2.v \\ R1.p.v \\ L.v \\ R1.v \\ C.i \\ der(L.i) \\ der(C.v) \end{Bmatrix} = \begin{bmatrix} 1 & & & 1 & & & & & & \\ & & 1 & & 1 & & & & & \\ & & & 1 & 1 & 1 & & & & \\ & 1 & & & 1 & & 1 & & & \\ & & & & & & 1 & 1 & & \\ 1 & & & & & 1 & & & 1 & \\ & 1 & & & & & & 1 & & 1 \end{bmatrix} \begin{Bmatrix} L.i_0 \\ C.v_0 \\ AC.VA \cdot \sin(\ ) \\ R2.v \\ R1.p.v. \\ L.v \\ R1.v \\ C.i \\ L.i \\ C.v \end{Bmatrix} \quad (8)$$

C. State-Space Representation

Next step is to make state-space representation. From BLT form, we make state-space representation with 2 cases:

- Theoretical Case

$$C.i = C.C*der\,(C.v)$$

$$L.v = L.L*der\,(Li)$$

$$\begin{Bmatrix} \frac{dC.v}{dt} \\ \frac{dL.i}{dt} \end{Bmatrix} = \begin{bmatrix} \frac{1}{C.C} & 0 \\ 0 & \frac{1}{L.L} \end{bmatrix} \begin{Bmatrix} C.i \\ L.v \end{Bmatrix} = \begin{bmatrix} \frac{1}{C.C} & 0 \\ 0 & \frac{1}{L.L} \end{bmatrix} \left(\begin{bmatrix} \frac{1}{R1R} & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} R1.v \\ R1.p.v - R2.v \end{Bmatrix} \right)$$
$$= \begin{bmatrix} \frac{1}{C.C}\frac{1}{R1R} & 0 \\ 0 & \frac{1}{L.L} \end{bmatrix} \left(\begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{Bmatrix} R1.p.v \\ C.v \\ R2.v \end{Bmatrix} \right) = \begin{bmatrix} \frac{1}{C.C}\frac{1}{R1R} & -\frac{1}{C.C}\frac{1}{R1.R} & 0 \\ \frac{1}{L.L} & 0 & -\frac{1}{L.L} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R2.R \end{bmatrix} \begin{pmatrix} R1.p.v \\ C.v \\ L.i \end{pmatrix}$$
$$= \begin{bmatrix} \frac{1}{C.C}\frac{1}{R1R} & -\frac{1}{C.C}\frac{1}{R1.R} & 0 \\ \frac{1}{L.L} & 0 & -\frac{R2.R}{L.L} \end{bmatrix} \begin{pmatrix} R1.p.v \\ C.v \\ L.i \end{pmatrix}$$
$$= \begin{bmatrix} -\frac{1}{C.C}\frac{1}{R1.R} & 0 \\ 0 & -\frac{R2.R}{L.L} \end{bmatrix} \begin{Bmatrix} C.v \\ L.i \end{Bmatrix} + \begin{Bmatrix} \frac{1}{C.C}\frac{1}{R1R} \\ \frac{1}{L.L} \end{Bmatrix} \{AC.AV * \sin(2 * AC.f * AC.PI * t)\} \quad (9)$$

- Practical Case

From horizontal result in (3) until (7), we can get the practical case of state-space representation as (10)

- R1.v = R1.p.v – C.v
- R1.p.v= AC.VA*sin(2*AC.f*AC.PI*t)
- R2.v= R2.R*L.i
- L.v = L.p.v - L.n.v = R1.p.v - R2.v
- C.i= R1.v / R1.R

$$\begin{Bmatrix} \frac{dL.i}{dt} \\ \frac{dC.v}{dt} \end{Bmatrix} = \begin{bmatrix} \frac{1}{L.L} & 0 \\ 0 & \frac{1}{C.C} \end{bmatrix} \begin{Bmatrix} L.v \\ C.i \end{Bmatrix} \quad (10)$$

TABLE IV
REVISED VERSION (PROPOSAL) FOR BLOCK LOWER TRIANGULAR FORM OF THE SIMPLE CIRCUIT MODEL

	R2.v	R1.p.v	L.V	R1.v	C.i	L.i	C.v
R2.v	1						
R1.p.v		1					
L.v	1	1	1				
R1.v		1		1			
C.i				1	1		
del(L.i)			1			1	
del(C.v)					1		1

III. ACAUSAL AND CAUSAL MODEL CONSTRUCTION WITH FEM APPROACH USING MODELICA

A. Comparison between Original Concept in Modelica and FEM Approach

In accordance with Fig. 1, first we want to discuss the differences between modelica original concept and FEM approach in stages of translating and executing a modelica model. As explained before, in modelica concept, we must conduct the horizontal and vertical sorting as an analyzer and get the sorted equation. By using FEM approach, we can make the simpler process. By using the simple circuit problem as an example, it is obtained that in modelica, the first model includes all variables (both known and unknown) and difficult to separate. With FEM approach, it is easier to separate known and unknown variables after building the system model.

The problem for a beginner in modelica concept is about the connector. Particularly in multi-domain, it is difficult to understand connector concept because in a different domain we must use different connector e.g. in electrical we use pin while in a mechanical domain use flange. To connect different domain, we must utilize the special connector. In FEM approach there is no connector concept, make it easier to understand. Furthermore, the other difference is we could not use graphical modeling in FEM approach and this matter can be one disadvantage. Admittedly, it can sometimes be rather convenient to use graphical modeling compared to textual modeling especially for beginner. Nevertheless, for practical use, it is difficult to define connector in multi domain as mentioned before. In FEM approach, we only use textual modeling to solve the problem. However, this approach has superiority in language using for different software. We can use the same language or same textual modeling for the other software with acausal model such as acausal model in scilab.

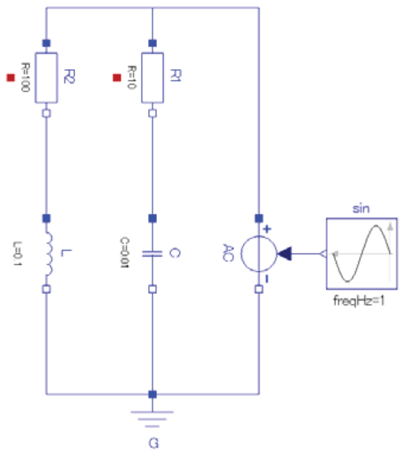


Fig. 5 Graphical modeling in modelica (orginal)

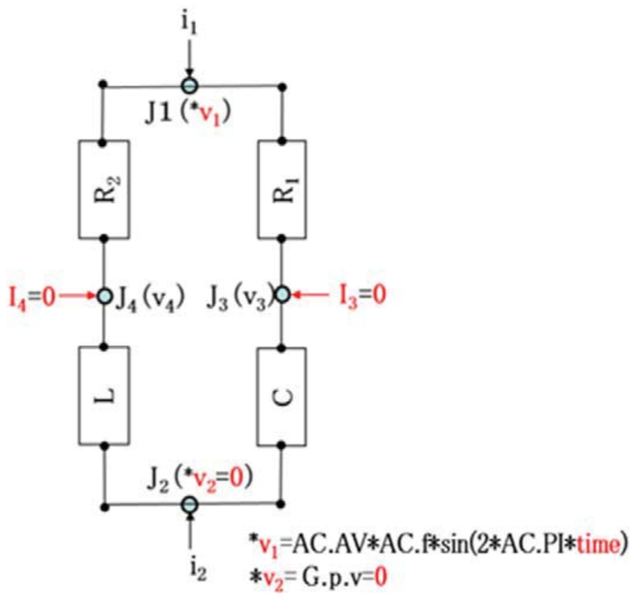


Fig. 6 Graphical modeling in modelica (FEM approach)

B. FEM Approach Using Modelica

In this section, we propose FEM approach as causal model construction for learning acausal model construction technology using modelica. We also show the causal model construction with FEM approach. Graphical modeling with the original concept in modelica is shown in Fig. 5.

In modelica, we must draw all known variables (such as a ground or a voltage source) on the graph model from beginning, so it may be said that in a sense it has already become the causal model. Whereas characteristic of our modeling technique suggesting that we made a truly acausal model, because we make all node-equilibrium equations without the distinction of known and unknown variable (v_j, i_j). Later then we set a value in the known variable. As a result, the necessarily unknown variable (node voltage and addition electric current) will be found.

For making it easier to understand, we make graphical modeling in FEM approach that uses 4 joint namely J1, J2, J3 and J4. FEM approach graphical modeling shown in Fig. 7.

Compare with original table in modelica for equation extracted from simple circuit model shown in Table I, we provide FEM approach. A component in modelica is named as an element in FEM approach i.e. AC, R1, R2, C and L. In modelica, each component defined at the first step and then connector. In FEM, each element is already connected and variable at joint are named as v1, v2, v3 and v4. So that the other difference between original modelica and FEM approach is in naming rule. In FEM, it is not only scalar variable is available but also vector and we can combine different domain. Internal variable definitions for FEM are R1.i_1 and R1.i_3 for R1, R2.i_1 and R2.i_4 for R2, C.i_4 and C.i_2 for C, L.i_3 and L.i_4 for L. Node variable definitions at joint are v_1, v_2, v_3, v_4 for voltage and i_1, i_2, i_3, i_4 for current. We can make system model for simple circuit model (internal variable representation) from Fig. 7 as (11):

$$\begin{cases} R1.i_1 + R2.i_1 = i_1 \\ L.i_2 + C.i_2 = i_2 \\ R1.i_3 + C.i_3 = i_3 \\ R2.i_4 + L.i_4 = i_4 \end{cases} \tag{11}$$

TABLE V
FEM APPROACH FOR EQUATIONS EXTRACTED FROM SIMPLE CIRCUIT MODEL

AC	$0=\{1 \quad -1\}\begin{Bmatrix} AC.p.i \\ AC.n.i \end{Bmatrix}$ $AC.AV*\sin(\ )=\{1 \quad -1\}\begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix}=\{1 \quad -1\}\begin{Bmatrix} v_1 \\ 0 \end{Bmatrix}$ $\therefore \quad v_1 = AC.AV*\sin(\ )$	L	$\begin{Bmatrix} v_2 \\ v_4 \end{Bmatrix}=L.L*del\left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} L.p.i \\ L.n.i \end{Bmatrix}\right)$ $\because \quad \{v_2 - v_4\} = L.L*del(L.p.i)$
R1	$\begin{Bmatrix} R1.p.i \\ R1.n.i \end{Bmatrix}=\frac{1}{R1.R}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} v_1 \\ v_3 \end{Bmatrix}$	G	$v_2 = 0$ =constant G.p.i = Unknown variable (reaction electric current)
R2	$\begin{Bmatrix} R2.p.i \\ R2.n.i \end{Bmatrix}=\frac{1}{R2.R}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} v_1 \\ v_4 \end{Bmatrix}$	wires	v_1,v_2,v_3,v_4

C	$\begin{Bmatrix} C.p.i \\ C.n.i \end{Bmatrix}=C.C*del\left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} v_3 \\ v_2 \end{Bmatrix}\right)$
	 $=C.C*\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} \dot{v}_3 \\ \dot{v}_2 \end{Bmatrix}$
		Flow at node	0 = AC.i + R1.p.i + R2.p.i //J1
			 0 = C.n.i + G.p.i + AC.n.i + L.n.i //J2
			 0 = R1.n.i + C.p.i //J3
			 0 = R2.n.i + L.p.i //J4

By inputing the component, element and internal variable definition in Tables IV and V, we can get flow variable equilibrium equation with node variable representation as (12):

$$\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & CC & -CC & 0\\ 0 & -CC & CC & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\\ \dot{v}_3\\ \dot{v}_4\end{Bmatrix}+\begin{bmatrix}\frac{1}{R1.R}+\frac{1}{R2.R} & 0 & -\frac{1}{R1.R} & -\frac{1}{R2.R}\\ 0 & 0 & 0 & 0\\ -\frac{1}{R1.R} & 0 & \frac{1}{R1.R} & 0\\ -\frac{1}{R2.R} & 0 & 0 & \frac{1}{R2.R}\end{bmatrix}\begin{Bmatrix}v_1\\ v_2\\ v_3\\ v_4\end{Bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & \frac{1}{LL} & 0 & -\frac{1}{LL}\\ 0 & 0 & 0 & 0\\ 0 & -\frac{1}{LL} & 0 & \frac{1}{LL}\end{bmatrix}\begin{Bmatrix}\int v_1 dt\\ \int v_2 dt\\ \int v_3 dt\\ \int v_4 dt\end{Bmatrix}=\begin{Bmatrix}i_1\\ i_2\\ i_3\\ i_4\end{Bmatrix} \quad (12)$$

In FEM approach, we can separate the known variable on left side and unknown variable on right side easily as follow:

- known variable:

  v_1 = sin (2*pi*time)"AC.VA"

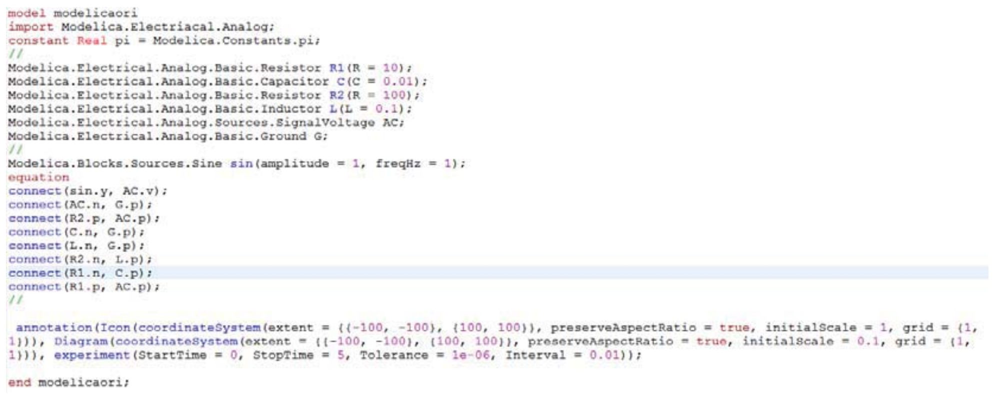
  v_2 = 0 " G.p.v"

  i_3 = 0;

  i_4 = 0;

- Then unknown variable: i_1, i_2, v_3, v_4 can be solved.

Afterwards, we can build textual modeling with modelica software and make a comparison between textual modeling in original concept with FEM approach. It is not difficult for simple circuit model to make textual modeling. For the more complex system, it becomes more complicated to make so many connectors between all components exist in modelica concept. Perhaps the FEM approach can be another alternative for a user to utilize modelica. We make two models with FEM approach, acausal and causal model construction. Causal model construction with FEM approach only needs 2 equations while acausal model needs 4 equations. Textual modeling and simulation results show in Figs. 8-13. The simulation results for all approaches are same.



```
model modelicaori
import Modelica.Electriacal.Analog;
constant Real pi = Modelica.Constants.pi;
//
Modelica.Electrical.Analog.Basic.Resistor R1(R = 10);
Modelica.Electrical.Analog.Basic.Capacitor C(C = 0.01);
Modelica.Electrical.Analog.Basic.Resistor R2(R = 100);
Modelica.Electrical.Analog.Basic.Inductor L(L = 0.1);
Modelica.Electrical.Analog.Sources.SignalVoltage AC;
Modelica.Electrical.Analog.Basic.Ground G;
//
Modelica.Blocks.Sources.Sine sin(amplitude = 1, freqHz = 1);
equation
connect(sin.y, AC.v);
connect(AC.n, G.p);
connect(R2.p, AC.p);
connect(C.n, G.p);
connect(L.n, G.p);
connect(R2.n, L.p);
connect(R1.n, C.p);
connect(R1.p, AC.p);
//

 annotation(Icon(coordinateSystem(extent = {{-100, -100}, {100, 100}}, preserveAspectRatio = true, initialScale = 1, grid = {1,
1})), Diagram(coordinateSystem(extent = {{-100, -100}, {100, 100}}, preserveAspectRatio = true, initialScale = 0.1, grid = {1,
1})), experiment(StartTime = 0, StopTime = 5, Tolerance = 1e-06, Interval = 0.01));

end modelicaori;
```

Fig. 7 Textual modeling with modelica original concept

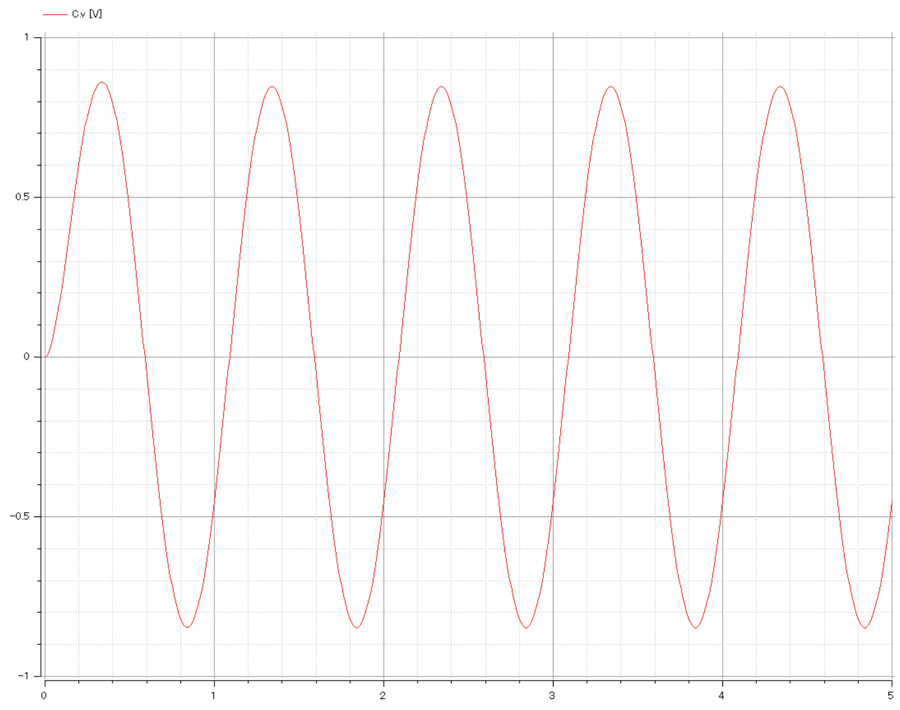
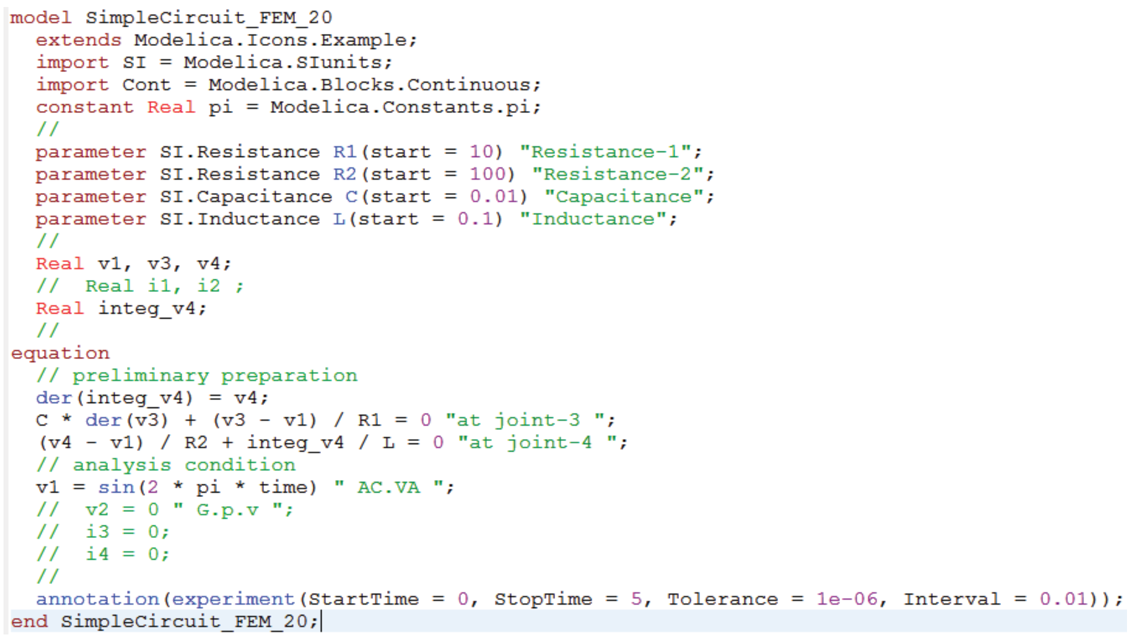


Fig. 8 Simulation result with modelica original concept



```
model SimpleCircuit_FEM_20
  extends Modelica.Icons.Example;
  import SI = Modelica.SIunits;
  import Cont = Modelica.Blocks.Continuous;
  constant Real pi = Modelica.Constants.pi;
  //
  parameter SI.Resistance R1(start = 10) "Resistance-1";
  parameter SI.Resistance R2(start = 100) "Resistance-2";
  parameter SI.Capacitance C(start = 0.01) "Capacitance";
  parameter SI.Inductance L(start = 0.1) "Inductance";
  //
  Real v1, v3, v4;
  //  Real i1, i2 ;
  Real integ_v4;
  //
equation
  // preliminary preparation
  der(integ_v4) = v4;
  C * der(v3) + (v3 - v1) / R1 = 0 "at joint-3 ";
  (v4 - v1) / R2 + integ_v4 / L = 0 "at joint-4 ";
  // analysis condition
  v1 = sin(2 * pi * time) " AC.VA ";
  //  v2 = 0 " G.p.v ";
  //  i3 = 0;
  //  i4 = 0;
  //
  annotation(experiment(StartTime = 0, StopTime = 5, Tolerance = 1e-06, Interval = 0.01));
end SimpleCircuit_FEM_20;
```

Fig. 9 Textual Modeling for FEM Approach (Causal model construction)

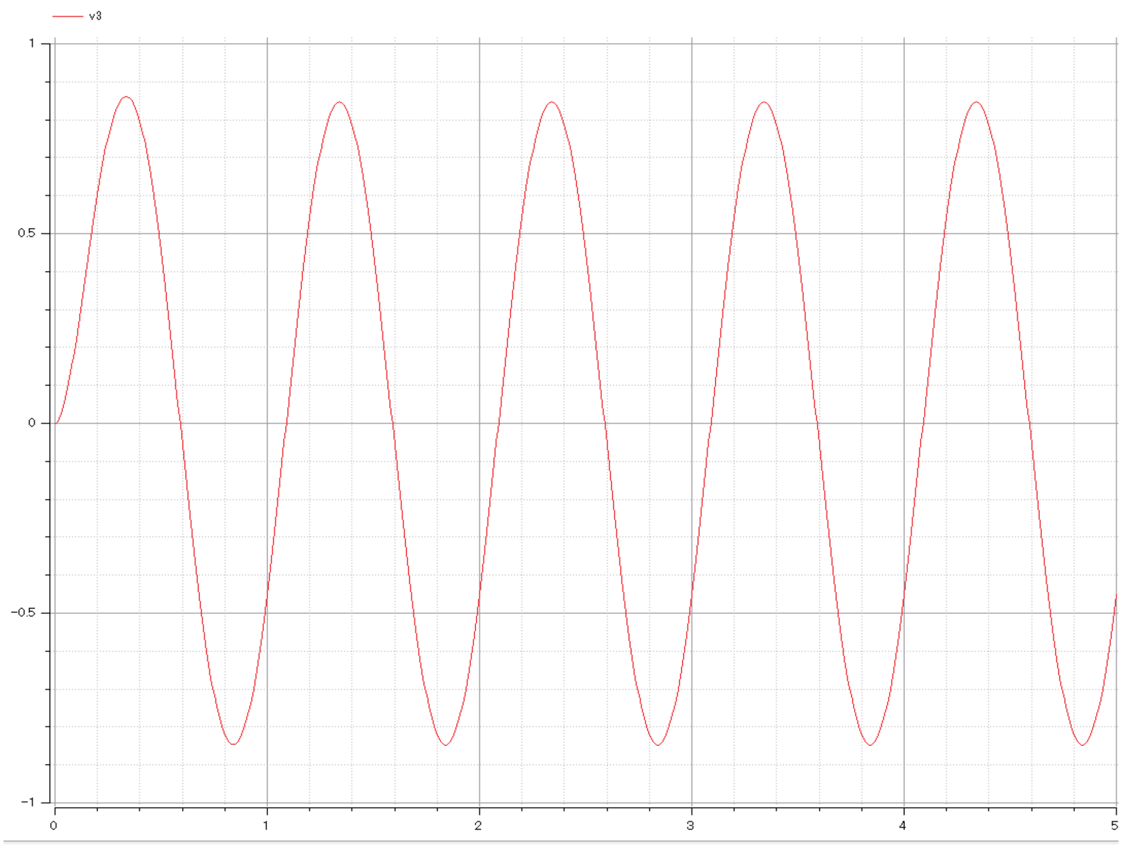
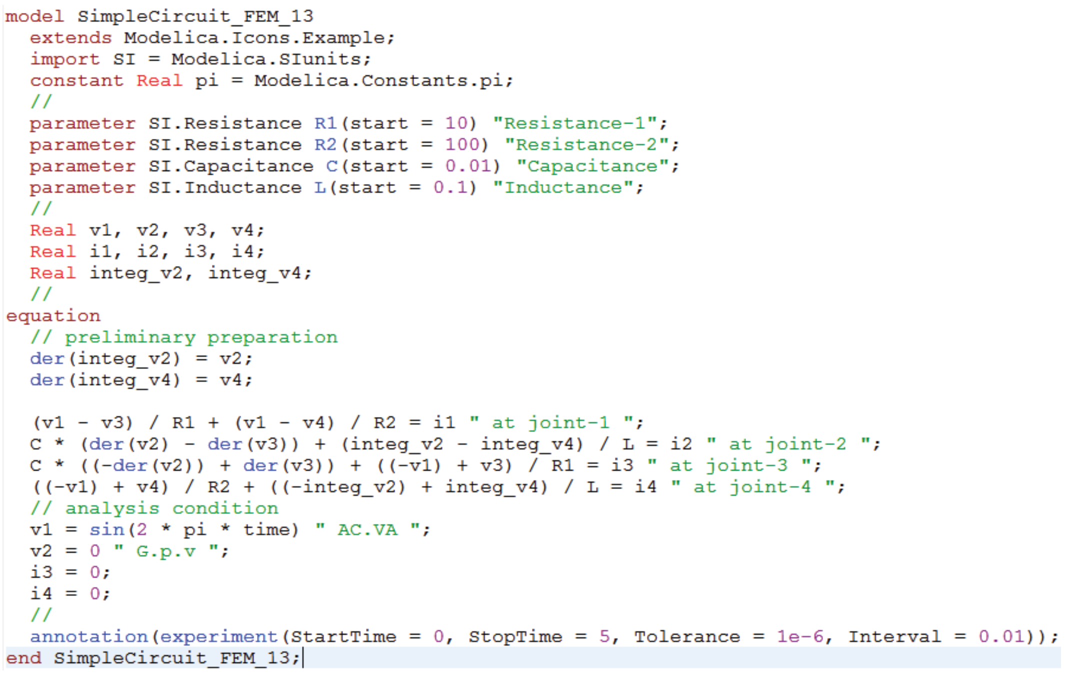


Fig. 10 Simulation result for FEM Approach (Causal model construction)



```
model SimpleCircuit_FEM_13
  extends Modelica.Icons.Example;
  import SI = Modelica.SIunits;
  constant Real pi = Modelica.Constants.pi;
  //
  parameter SI.Resistance R1(start = 10) "Resistance-1";
  parameter SI.Resistance R2(start = 100) "Resistance-2";
  parameter SI.Capacitance C(start = 0.01) "Capacitance";
  parameter SI.Inductance L(start = 0.1) "Inductance";
  //
  Real v1, v2, v3, v4;
  Real i1, i2, i3, i4;
  Real integ_v2, integ_v4;
  //
equation
  // preliminary preparation
  der(integ_v2) = v2;
  der(integ_v4) = v4;

  (v1 - v3) / R1 + (v1 - v4) / R2 = i1 " at joint-1 ";
  C * (der(v2) - der(v3)) + (integ_v2 - integ_v4) / L = i2 " at joint-2 ";
  C * ((-der(v2)) + der(v3)) + ((-v1) + v3) / R1 = i3 " at joint-3 ";
  ((-v1) + v4) / R2 + ((-integ_v2) + integ_v4) / L = i4 " at joint-4 ";
  // analysis condition
  v1 = sin(2 * pi * time) " AC.VA ";
  v2 = 0 " G.p.v ";
  i3 = 0;
  i4 = 0;
  //
  annotation(experiment(StartTime = 0, StopTime = 5, Tolerance = 1e-6, Interval = 0.01));
end SimpleCircuit_FEM_13;
```

Fig. 11 Textual Modeling for FEM Approach (Acausal model construction)

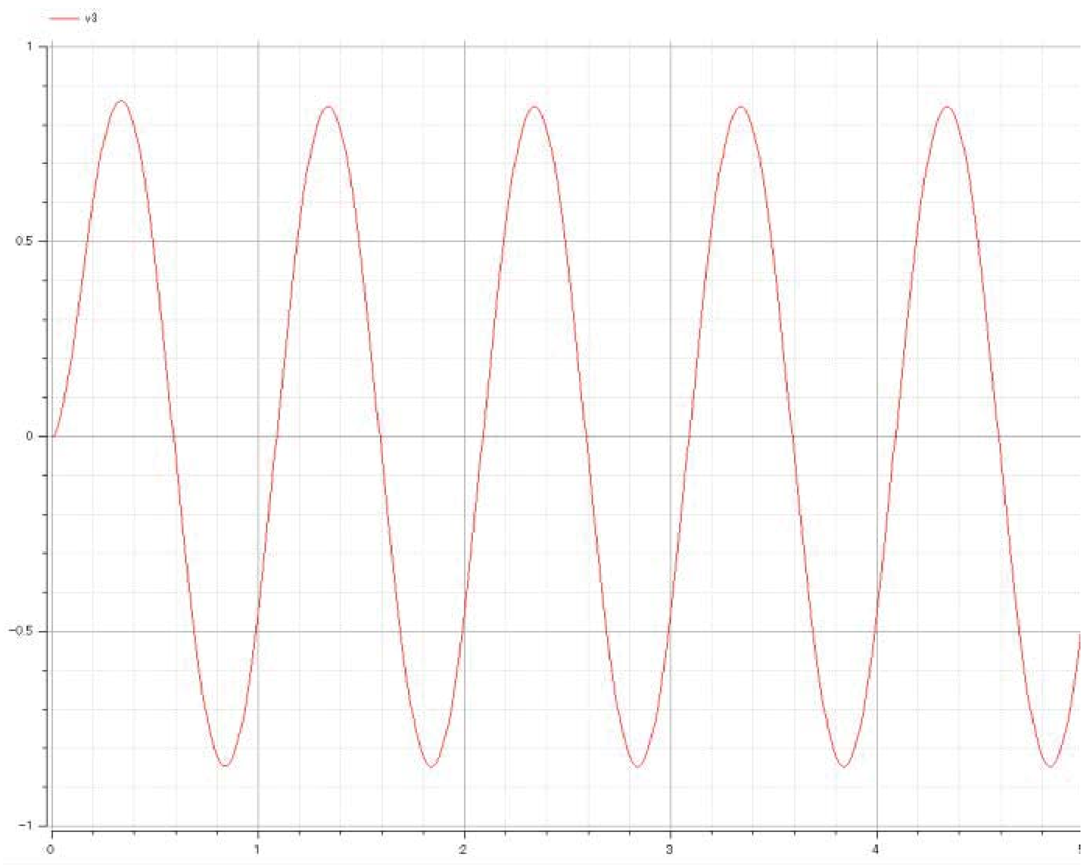


Fig. 12 Simulation result for FEM Approach (Acausal Model Construction)

IV. Conclusion

In this article, explanation for solver ordering system is provided with some revisions to improve modelica's book ability as an electronic self-teaching material. The difficulties of sorting step and connector concept in modelica can be solved by using FEM approach. Textual modeling with FEM approach for acausal and causal model construction is made and compared with modelica concept. Simulation result that is obtained from all different textual modeling is the same. Therefore, FEM approach can be an alternative for beginner who familiar with FEM for using modelica.

Acknowledgment

This project supported by JSPS KAKENHI Grant Number 26420079 and Strength of Material laboratory Yamaguchi University, Japan. The first author wishes thank for scholarship support from the Directorate of Higher Education Indonesia cooperated with University of Brawijaya.

References

[1] Fritzson P., Bunus P., “Modelica – A General Object-Oriented Language for Continuous and Discrete-Event System Modeling and Simulation”, In Proc. Annual Simulation Symposium (2002): 365-380.
[2] Fritzson P., “Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica”, A John Wiley & Sons, Inc., Publication, 2011.
[3] Kousa J. A., “Virtual Computational Test bed for Building Integrated Renewable Energy Solutions”, Thesis report of Eindhoven University of Technology, 2014.
[4] Otter M., “Modeling, Simulation & Control with Modelica 3.0 & Dymola 7”, 2009.
[5] Elmqvist H., et al, “Object Oriented & Hybrid Modeling in Modelica”, Journal Europeen des Systemes Automaties, AP II-JESA, volume 35, 2001.
[6] Zupančič B., Sodja A.,” Computer-aided Physical Multi-Domain Modelling: Seome Experiences from Education and Industrial Applications”, Simulation Modelling Practice and Theory 33, (2013): 45-67.
[7] Liu G. R., et al, “The Finite Element Method, A Practical Course”, Elsevier Science Ltd. All rights reserve, 2003.