Improving the Finite Element Method using Cell-DEVS

Hesham Saadawi **Gabriel Wainer**

Dept. of Systems and Computer Engineering
Carleton University
4456 Mackenzie Building. 1125 Colonel By Drive
Ottawa, ON. K1S 5B6. Canada.
gwainer@sce.carleton.ca

Keywords: Cell-DEVS models, Finite Elements analysis, DEVS formalism.

Abstract

Finite element analysis has been used successfully since the advent of computers to analyze complex engineering and physical systems. FEM modeling lacks a well-defined formal definition for composing new models from pre-defined simpler models. Building hierarchical models of engineering and physical systems out of simpler components provides advantages in terms of testing, maintenance and reuse. In order to achieve these goals, we explored the use of Cell-DEVS as an alternative to model and solve problems that are usually tackled using the Finite Elements Method (FEM). We focus on an example about how to approach the problem definition using Cell-DEVS, which provides a base to extend this to more complex problems. We discuss the advantages of this approach to solving complex physical and engineering problems against the of finite elements or finite differences methods.

INTRODUCTION

The advance in science and technology has usually relied in the definition of models representing properties of systems under study. Complex problems in the domain of physics, chemistry and biology were usually modeled with differential equations, and by traditional analysis, these equations were solved analytically for the desired system. This approach only works for very simple systems with simple geometries and property distributions. For any real life problem, obtaining an analytical solution is almost impossible unless many approximations are done. The advent of digital computers permitted the definition of advanced numerical methods, which enabled the analysis of very complex differential equations. One of such methods, the **finite element analysis** [1] has been used successfully to analyze complex engineering and physical systems. Typical areas of applications include structural analysis, thermal analysis for heat transfer, electromagnetic analysis, fluid analysis, etc. The Finite Elements Method (FEM) consists of defining a solution that satisfies the partial differential equation on average over a finite element. Every element is connected to a neighboring elements, and the field under study is analyzed by propagating the current values from one element to another through connection points.

In recent years, there has been a tendency to analyze complex natural and artificial systems like the ones traditionally modeled with FEM as cell spaces [2, 3] and cellular Automata (CA) [4]. CA are defined as infinite n-dimensional lattices of cells whose values are updated according to a local rule. This is done simultaneous and synchronously using the current state of the cell and the state of a finite set of nearby cells (known as the neighborhood). CA have several problems when used in modeling complex systems: they usually require large amounts of compute time due to its synchronous nature, which constrain the precision of the model. The Cell-DEVS formalism solved [5] these problems by using the DEVS (Discrete EVents Systems specifications) formalism [6] to define a discrete event cell space with explicit timing delays.

The discretization of model into a grid poses constraints on the precision that can be achieved by the model. Finite element analysis, instead, is able to provide higher precision. We explored the use of Cell-DEVS to model and solve problems usually tackled by FEM. We intend to use FEM as a very precise technique for defining the problem, while having the simplicity of a cellular approach to facilitate model definition. Likewise, we base the development in the use of Cell-DEVS, which enable immediate integration with other existing DEVS and Cell-DEVS models. A simple example shows how to approach the problem definition using Cell-DEVS, and try to predict how to extend this to more complex problems. It also compares results obtained by Cell-DEVS approach to those by FEM to highlight the advantages and disadvantages of Cell-DEVS.

The advantage of modeling FEM with Cell-DEVS is that in real life, continuous and discrete systems often interact together. This dictates that we need a way to integrate both models and simulate their global behavior. DEVS provides means for modeling discrete event systems, and enables the construction of hierarchical models. This would enable modelers to build hybrid hierarchical systems from continuous and discrete components using the same base formalism, improving interaction and facilitate implementation in existing tools.

We describe a method of mapping a physical problem that is usually modeled by a partial differential equation and solved numerically either by finite differences or finite elements methods, into a formal Cell-DEVS specification. This model is then defined using the CD++ tool [7], an environment that enables the creation of Cell-DEVS models. We also discuss the advantages of our

approach to solving complex physical and engineering problems against the use of finite elements or finite differences methods.

BACKGROUND

FEM was originally created to solve problems of structural mechanics, and it was later applied to many other problems in Physics. FEM provides piecewise approximation of a partial differential equation over a continuum. A partial differential equation is presumed over that continuum. A finite element is a discrete piece of that continuum. We usually find two major components that can be identified within each element: **field**, and **potential**. The field is a quantity that varies with position within structure analyzed. The fields are related to the potentials as their derivatives with respect to position. The potential can be thought as the driving force for the spread of the field in the material. For example, temperature difference in a material would cause a heat flux to be transferred from one point to another in that material. The heat flux direction and quantity is related with the difference in temperature in that material (temperature gradient).

The finite element method is a mathematical procedure for satisfying partial differential equation over the element. By assuming the field variable is following a simple function over the finite element, we can approximate the solution of the partial differential equation over that element. Elements in the structure are considered to be connected together through, the vertices on boundaries of each element, which are called **nodes**. In order to solve the problem, we must:

1. Divide the structure under study into a large number of elements, each of them with a simple geometry
2. A simple interpolation function is assumed over the element, representing the shape of the spatial solution in that element.
3. The differential equations can be solved for that particular element by assuming the shape of the change of potential function in that element. This gives an approximate solution for a single element: simple algebraic equations are obtained for the element, represented in a matrix.
4. As all the elements in the structure are connected together through nodes located at their edges, we obtain a system of equations represented in N x N matrices. We only know the values at certain points in the structure (usually at its boundaries). These values are used to get the unknown potential inside the structure.
5. The global equations are solved, and the final solution would give the distribution of the potential over the structure, represented by the values obtained at the nodes of each element. The precision can be enhanced by dividing the structure into more elements, or by assuming a more precise distribution of the potential inside the element itself.

Most systems modeled by FEM can be represented by a matrix of the form: $[S]\,[T] = [P]$

The matrix form results from combining the equations for all elements defined in a structure. Thus, [S] is an N x N "stiffness" matrix and represents elements properties (material constants, dimensions, etc.), [T] is an N column vector representing the unknown potential values at each element node. [P] is an N column vector representing the forcing function, for example in heat transfer problems this would be the power input per unit volume.

Cell-DEVS defines cell spaces in which each cell is defined as a DEVS model. The DEVS formalism was originally proposed to model discrete events systems. In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS model. Cell-DEVS atomic models are specified as:

$$TDC = < X, Y, S, N, delay, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >.$$

Each cell will use the **N** inputs to compute the future state **S** using the function $\boldsymbol{\tau}$. The new value of the cell is transmitted to the neighbors after the consumption of the delay function. A **transport** delay allows us to model a variable commuting time for each cell with anticipatory semantics. Using **inertial** delays, the semantics is preemptive: some scheduled events are not executed due to a small interval between two input events. **Delay** defines the kind of delay for the cell, and **d** its duration. This behavior is defined by the $\boldsymbol{\delta_{int}}$, $\boldsymbol{\delta_{ext}}$, $\boldsymbol{\lambda}$ and **D** functions, as in other DEVS models.

Once the atomic cell model is defined, a number of cells they can be put together to form a coupled model, built as an array of atomic cells. A Cell-DEVS coupled model is defined by:

$$GCC = < X_{list}, Y_{list}, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z >.$$

The cell space **C** is a coupled model composed by an array of atomic cells with size $\{\mathbf{t_1\ x...x\ t_n}\}$. Each cell in the space is connected to the cells defined by the neighborhood **N**. The cell space can be "wrapped", meaning that cells in a border are connected with those in the opposite one. Otherwise, the borders **B** should have a different behavior than the remaining cells. The **Z** function allows one to define the internal and external coupling of cells in the model. The **CD++** tool [7] was developed following the formal definitions of the Cell-DEVS formalism. CD++ is a tool to simulate both DEVS and Cell-DEVS models. DEVS models can be written in C++, and Cell-DEVS models are described using a built-in specification language. The language provides a set of primitives to define the size of the cell-space, the type of borders, a cell's interface with other DEVS models and a cell's behavior.

FEM models resembles to a large extent Cell-DEVS models, in which changes of a cell value would trigger its neighboring cells to change themselves, as though a field is propagating through all of them. The following sections will be devoted to show how Cell-DEVS can be used to describe FEM models. The idea is to describe the model behavior in terms of cell behavior, discrete event interaction and timing delays.

MAPPING FEM INTO CELL-DEVS

In this section we will show how FEM models can be mapped into Cell-DEVS using a traditional example found in [1]. This model epresents steady-state heat transfer with convection from a fluid into a composite wall of different materials. This resembles the heat flow through a wall of a heated furnace to ambient air. This example involves heat transferred by convection from hot air to the inside wall and by conduction through wall material (in this particular example, heat transferred by radiation is neglected). Steady-state heat flow is defined as a heat flux fixed with regard to time. In non steady-state heat transfer, the heat flux value and thus temperature distributions in a material change over time.

Heat transfer occurs when there is a temperature difference within a body or between a body and its surrounding medium. This temperature difference constitutes the potential driving the heat flux through the material. The temperature difference over a infinitely small piece of material would give us the *temperature gradient* over this element. Heat flows from hot spots towards cooler ones. Heat conduction in a two-dimensional steady state isotropic medium is given by Fourier's law as:

$$q_x = -k\frac{\partial T}{\partial x} \quad , \quad q_y = -k\frac{\partial T}{\partial y} \qquad (1)$$

Where q is the heat flux (W/m^2), q_x is the heat flux component in x direction, q_y is the heat flux component in y direction, k is the thermal conductivity of the material (W/m. ^{o}C), T = T(x,y) is the Temperature field in the medium and is a function in x and y, $\partial T / \partial x$ and $\partial T / \partial y$ are the temperature gradients over x and y respectively. The minus sign is to indicate that the direction of heat flux is opposite to direction of increasing temperature.

In convection heat transfer, the heat flux is given by:

$$q = Ah(T_\infty - T_s) \qquad (2)$$

where h (W/m^2.^{o}C) is the film (a property of the fluid around the surface), T_∞ and T_s are fluid and surface temperature respectively, A is the surface area exposed to the flow. For a small element assuming a linear temperature distribution along its unit length, and a unit area perpendicular to heat flow direction, the heat flux conduction would be:

$$q = k\frac{dT}{dx} = k\frac{(T_h - T_l)}{1} = k(T_h - T_l) \qquad (3)$$

Where, T_h, T_l are the high and low temperatures of its ends respectively.

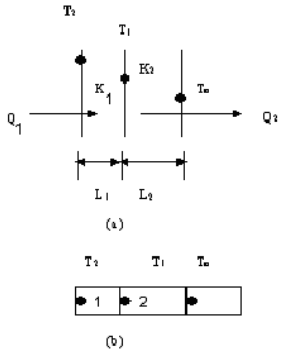

Figure 1. (a) Two elements physically connected. (b) Elements represented as finite elements/nodes.

In order to get the updating rules for any cell in our Cell-DEVS model, we first study a subset of the complete problem to solve, in which we consider only two elements connected together through their nodes. By solving this sub-problem, we would then generalize our findings to the complete problem model that we would describe later. In figure 2 (a), we show two layers of the wall. These two layers are connected together through the surface in the middle. Each Layer *i* has different physical properties: K_1 is the thermal conductivity, L_i the length, and Q_i the heat flux through that wall. Temperature distribution on each surface on the walls is denote as T_2 , T_1, and T_0 as shown in Figure 1 (a).

Each layer can be represented by one finite element, as showed in Figure 1 (b). Elements 1 and 2 in that figure contain two nodes, one at each end. These two elements are connected together through their nodes, making the middle node shared between both of them as in Figure 2 (b). Every node would then represent a surface of a wall and the corresponding node value represents this surface temperature. In a Cell-DEVS model, every node would be represented by a cell.

From (3) by assuming a linear temperature distribution along the elements, we get:

$$Q_1 = K_1 / L_1 . (T_2 - T_1)$$
$$Q_2 = K_2 / L_2 . (T_1 - T_o)$$

Having the conservation of energy equation over a control volume containing both elements 1 and 2 (input heat flux equals output heat flux), we have:

$$Q_1 = Q_2 \qquad (3a)$$

and we get:

$$T_1 = \frac{\frac{K_1}{L_1}T_o + \frac{K_2}{L_2}T_2}{\frac{K_1}{L_1} + \frac{K_2}{L_2}} \qquad (4) \text{ for heat conduction.}$$

Similarly, when we study two elements in which one is a convective and the other is conductive, we get:

$$T_1 = \frac{hT_\infty + \frac{K_1}{L_1}T_o}{h + \frac{K_1}{L_1}} \qquad (5) \text{ for heat convection.}$$

Equations (4) and (5) above can be used as the updating rules for our cell-DEVS model. Equation (4) describes the heat conduction rule inside the material. It specifies the middle node T_1 temperature as a function of its two adjacent nodes and constant material properties. Equation (5) describes the middle node temperature as a function of adjacent nodes of fluid temperature T_∞ and inner node temperature T_o inside the material. This represents the case as at a convective boundary, and T_1 is the surface temperature.

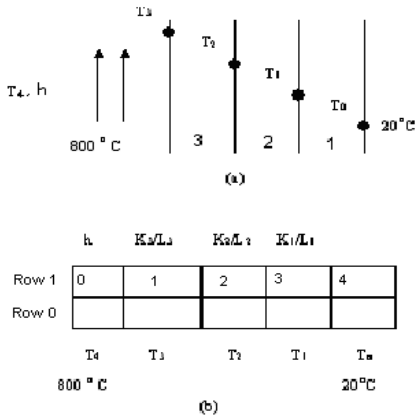

Figure 2. (a) Steady-state heat transfer through a composite wall. (b) The problem as a Cell-DEVS space.

Every cell value would thus be a function of its right cell value, its stored physical properties, left cell value, and its left cell physical properties. Note that in case of having identical elements (same K and L), the updating rule for a cell's temperature would be a simple arithmetic mean of its two neighboring cell temperatures. Figure 2 shows an extended version of the example presented in

[1]. Figure 2(a) represents a composite wall of three materials with layers numbered 1, 2 and 3. The outer temperature is T_o = 20 ^oC. Convection heat transfer takes place on the inner surface of the wall with fluid temperature T_4 = 800 ^oC and film coefficient h = 25 $W/m^2.^oC$. We need to determine the temperature distribution in the wall (i.e. on surfaces of each layer). Composite layers lengths are L_1 = 0.3 m, L_2 = L_3 = 0.15 m. Conductivities are K_1 = 50 W/m. ^oC, k_2 = 25 W/m. ^oC, K_3 = 20W/m.^oC, for layers numbered 1, 2, and 3 respectively.

In figure 2(b), we show an equivalent Cell-DEVS model for the problem defined in (a). In this Cell-DEVS model, we represent each point with temperature measure in Figure 2(a), with a cell in the bottom row. Thus, cell (0,0), which is cell in Row 0 and Column 0, represents fluid temperature T_4. Cell (0,1) represents T_3. Cell (0,2) represents T_2. Cell (0,3) represents T_1. Cell (0,4) represents T_0. Cells in Row 1 store the physical properties corresponding to wall layers as shown in Figure 2(b). Cell (1,0) contains value of h, cell (1,1) contains the value of (K_3/L_3), cell (1,2) contains the value K_2/L_2, cell (1,3) contains the value k_1/l_1, and cell (1,4) contains unity value, as it is not used in calculations.

```
[conduction-rule]
rule : { ((-1,0)*(0,1)+(-1,-1)*(0,-1)) / ( (-
1,0)+(-1,-1) ) } 1 { t }
; Implements equations (3) and (4). Physical
properties K/L are stored in (-1,0) and (-1,-1).

[Constants]
rule : {(0,0)} 1 {t}
; Constant physical properties stored in row 0 as
defined in a "zone" in the model.

[Boundary]
rule : {(0,0)} 1 {t}
; boundary conditions.
```

Figure 3. CD++ rules for the heat transfer model.

Cells in Row 1 are constant as the physical properties are independent of temperature distribution. Cells (0,0) and (0,4) contain constant temperature which constitute the boundary conditions, and fixed. Each element properties are stored in the cell above it (for example, physical properties of element (0,0) would be in (-1,0), as in figure 4). Only cells in Row 0, which corresponds to T3, T2, and T1, are updated. The model is initially loaded with values representing material properties on row 0, boundary values on cells (1,0) and (1,4), and arbitrary values in other cells. The following is the specification for cell updating rules in CD++. *[conduction-rule]* implements the equations (4) and (5). It also works for convection as it uses film coefficient value instead of thermal conductivity at cells adjacent to fluid cells. *[Constants]* defines the updating rule for cells in row zero. *[Boundary]* defines the updating rule for boundary cells, which also keep a fixed value.

The complete specification of the Cell-DEVS coupled model is shown in Figure 4. The *heatcond* model is a Cell-DEVS component, built as a bidimensional grid (2 rows and 5 columns). Its delay type is a *transport* delay. Its *border* is non-wrapped (we run specific rules for the boundary conditions). The *neighbors* define the cell space neighborhood, and *Localtransition* the rules used for local transition function (defined previously in Figure 3), and *Zone* defines group of cells that constitute a zone with its own updating rule that differs from the general updating rule.

```
[heatcond]
type : cell   dim : (2,5)  delay : transport
border : nowrapped
neighbors : (-1,0) (0,0) (0,1) (-1,-1) (0,-1)
localtransition : conduction-rule
zone : Constants { (0,0)..(0,4) }
zone : Boundary { (1,0) (1,4) }
```

Figure 4. Cell-DEVS coupled model definition.

MODEL EXECUTION RESULTS.

After defining the CD++ specification, we executed this model with multiple test cases. The models were executed until cell values become stable, converging to a solution for our problem that satisfies equilibrium equations of steady-state heat transfer as defined in equation (3a) in section 3. The resultant values would represent temperature distribution over the structure. In our first test case, the cells between the hot and cold boundaries were initialized with a temperature of 20 ^oC inside the wall. Figure 7 in the Appendix shows the results for this model. The final step shows T_3=304.75^oC, T_2=119.03 ^oC, T_1 = 57.14 ^oC, corresponding to nodes temperatures as in figure 3. This same example were solved in [1] using Finite Elements Analysis and gave the following values: T_3 = 304.6 ^oC, T_2 = 119.0 ^oC, and T_1 = 57.1 ^oC.

This shows that the solution converges to the correct answer after 23 steps. The above results show the solution converges to the right value with each successive iteration. This process is in fact equivalent to solving a system of linear equations using Jacobi iterative method. In this method, we solve one equation for one variable, with all other variables fixed. In the Jacobi method [9], if we have a system of *n* linear equations on the form $Ax = b$, and if the *i*th equation in the form

$$\sum_{j=1}^{n} a_{i,j}x_j = b_i$$

We can solve for value of x_i while assuming the all other variables are fixed as

$$x_i = (bi - \sum_{j \neq i} a_{i,j}x_j) / a_{i,i}$$

And iteratively would be

$$x_i^{(k)} = (b_i - \sum_{j \neq i} a_{i,j}x_j^{(k-1)}) / a_{i,i}$$

where *k* is the iteration number. The method begins by assigning arbitrary values to all variables, and a new value is computed for each variable. During each iteration *k*, variable values from previous iteration *(k-1)* are used to calculate the new value of x_i. Only after iterating with all variables, the variables are updated to their new values. The order in which the equations are examined is irrelevant, since the Jacobi method treats them independently. The CD++ execution of our Cell-DEVS model resembles Jacobi method. We have a system of linear equations defined in our Cell-DEVS model, from applying equations (4) and (5) for every adjacent two elements, thus producing *n-1* equations (*n* = number of elements). From each equation, we update the cell values as a function of the neighboring cells, and we keep all the cell values from one time step fixed to for use in next time step to calculate new cell values. The order of evaluation of cell values is irrelevant to the result, which enables parallel execution of our model. During each time step, cell updates can be done simultaneously as they are independent of each other. This exploits the nature of asynchronous updates of Cell-DEVS.

In a second test case, we considered, the nodes T_3, T_2, and T_1 were initialized with temperatures of 3000 °C, in order to measure speed of convergence to that solution. The results are shown at Figure 6, as we converge into the final correct answer after 30 steps. This is a 36% more steps than the first test case. This shows the impact of choosing cells initial values on the number of steps to converge to the final solution. Solving the same problem with FEM, we would get a system of linear equations that models the problem. Then, using a suitable method for solving this system we get the temperature distribution. Usually the method of Gaussian elimination is used [10]. This does not prevent using an iterative method like Jacobi, which is equivalent to the way CD++ solves this problem. However, CD++ implementation of Cell-DEVS models can enhance the Jacobi method performance: if a cell's neighborhood is unchanged during a simulation time step, the cell's external transition function is not executed at next time step. This means that the cell would not re-calculate.

The last two cases presented are defined to stress on the previous finding that initializing the model cells with proper values can save valuable simulation time. Assume, for instance, that the hot fluid temperature T_4 is now 850 ^oC. We need now to solve the new problem, which resembles the old one in every aspect, but in fluid temperature. This new problem would be modeled the same as the previous model, except that we use a high temperature equals to 850 degrees in the *boundary* element [cell (1,0)] instead of old value of 800. The model cells for T_3, T_2, and T_1 are initialized with values that resulted from solving the previous problem. As we may expect, the iterative solution in this case starts with cell values near the correct solution, thus saving some iteration cycles and converging in 20 time (Figure 7). Finally, we initialized the cells with arbitrary values (–3000 ^oC). The model execution took 35 time steps to converge. This shows that for complex and large models, we can enhance performance of subsequent model executions if we initialize all unknown cell values with results coming from the previous simulation for an almost similar model, as we did in the example presented in Figure 8.

CONCLUSION

We showed how to use the Cell-DEVS formalism to model problems that have been usually tackled with other methods as FEM and Finite differences. Although the example used here is simple compared with real-life problems, this technique can be extended to model more complex problems. Modeling physical and engineering problems with Cell-DEVS can have its own advantages over traditional methods. Cell-DEVS is a well-defined formalism for model specification including its interfaces with other models. This property enables the construction of atomic and coupled models that can be composed together to form a more complex model. As engineering systems are typically constructed from several and possibly thousands of components, modeling and simulating these systems need a well-defined method to build and integrate those models. A method that is capable of building more complex models from simpler ones, and allow for re-use of such models can be very useful when using simulation to design and optimize systems. Cell-DEVS enables building coupled systems that are composed of many atomic or coupled components. This property can be very useful when simulating engineering systems as they often compose of many continuous and discrete components. Simulation of complex models can use the ability of Cell-DEVS models to execute in parallel.

Initializing the model cells with a previous solution of similar model can cut simulation time, because solving the problem in Cell-DEVS resembles the Jacobi method for solving system of linear equations. As in Jacobi method, starting the iteration with variables initialized near to their correct values, would accelerate the convergence to the final solution. Definition of complex equations can be easily done using the rule specification techniques of CD++, as it could be seen with this particular example. This highly reduces the effort spent by the users in developing the applications. Automated verification facilities in the toolkit improve testing and reduce delivery time. Likewise, applying changes to the model only results in slight changes in the model specifications, without any need for code rewriting, which enables easy analysis of more complex system conditions.

In contrary to this, unless some intelligence is in its matrix-solving algorithm is used; FEM would solve the N x N matrix all over again even if a small change were done to a boundary condition. This small change may only affect several cells around that boundary condition, in which case a Cell-DEVS model simulation can be more efficient than FEM. Cell-DEVS asynchronous nature makes possible to have independent components working at their own rate of execution, and each element can be associated with different level of activity. This can be easily modeled thanks to the explicit timing delay constructions available in each cell in a Cell-DEVS model.

REFERENCES

[1] CHANDRUPATLA, T. ; BELEGUNDU; A. "Introduction to finite elements in engineering", *Prentice Hall*, 1997.

[2] TALIA, D. "Cellular processing tools for high-performance simulation". IEEE Computer. September 2000. Pp. 44 –52.

[3] SIPPER, M. "The emergence of cellular computing". IEEE Computer. July 1999. Pp. 18-26.

[4] WOLFRAM, S. "A new kind of science". Wolfram Media, Inc. 2002.

[5] WAINER, G.; GIAMBIASI, N. "N-Dimensional Cell-DEVS". In *Discrete Events Systems: Theory and Applications*, Kluwer. Vol. 12, No. 1. January 2002. pp. 135-157.

[6] ZEIGLER, B.; KIM, T.; PRAEHOFER, H. "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems". Academic Press. 2000.

[7] WAINER, G. "CD++: a toolkit to define discrete-event models". G. Wainer. *Software, Practice and Experience*. Wiley. Vol. 32, No.3. pp. 1261-1306. November 2002.

[8] BRAUER, J. "What Every Engineer Should know About Finite Element Analysis", *Marcel Dekker, Inc.* 1988.

[9] BARRETT, R.; BERRY, M.; CHAN, T.F.; DEMMEL, J.; DONATO, J.; DONGARRA, J.; EIJKHOUT, V.; POZO, R.; ROMINE, C.; VAN DER VORST, H. "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition", *SIAM*. 1994.

[10] BRANDAL, W. “Numerical Linear Algebra”, *BCS Associates*. 1991.

APPENDIX

```
Line : 29 - Time: 00:00:00:000
        0             1             2             3             4
 +-------------------------------------------------------------------------+
0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
1|  800.00000000   20.00000000   20.00000000   20.00000000   20.00000000|
 +-------------------------------------------------------------------------+

Line : 44 - Time: 00:00:00:001
        0             1             2             3             4
 +-------------------------------------------------------------------------+
0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
1|  800.00000000  232.72727966   20.00000000   20.00000000   20.00000000|
 +-------------------------------------------------------------------------+
...
Line : 473 - Time: 00:00:00:023
0            1             2             3             4
+-------------------------------------------------------------------------+
0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
1|  800.00000000  304.74679565  119.02681732   57.13505936   20.00000000|
+-------------------------------------------------------------------------+
```

Figure 5. Simulation results for initial temperature = 20 ^{o}C.

```
Line : 30 - Time: 00:00:00:000
                0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  800.00000000 3000.00000000 3000.00000000 3000.00000000   20.00000000|
    +-------------------------------------------------------------------------+
         . . ...
Line : 638 - Time: 00:00:00:031
                0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  800.00000000  304.78213501  119.07543182   57.15327835   20.00000000|
    +-------------------------------------------------------------------------+
```

Figure 6. Simulation results for initial temperature = 3000 ^{o}C.

```
Line : 31 - Time: 00:00:00:000
             0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  850.00000000  304.76190186  119.04762268   57.14286041   20.00000000|
    +-------------------------------------------------------------------------+
         . . . . . .
Line : 459 - Time: 00:00:00:020
              0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  850.00000000  323.01135254  125.39394379   59.52148056   20.00000000|
    +-------------------------------------------------------------------------+
```

Figure 7. Changing boundary conditions and initial temperature = 20 ^{o}C.

```
Line : 30 - Time: 00:00:00:000
             0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  850.00000000-3000.00000000-3000.00000000-3000.00000000   20.00000000|
    +-------------------------------------------------------------------------+
         . . . . . .
Line : 610 - Time: 00:00:00:035
              0             1             2             3             4
    +-------------------------------------------------------------------------+
   0|   25.00000000   66.66666412  200.00000000  333.33334351    1.00000000|
   1|  850.00000000  323.01116943  125.39035034   59.52138138   20.00000000|
    +-------------------------------------------------------------------------+
```

Figure 8. Changing boundary conditions and arbitrary initial temperature.