Available online at www.sciencedirect.com
ScienceDirect
Simulation Modelling Practice and Theory xxx (2007) xxx–xxx
SIMULATION MODELLING PRACTICE AND THEORY
www.elsevier.com/locate/simpat

Modeling physical systems using finite element Cell-DEVS

Hesham Saadawi, Gabriel Wainer *

Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada K1S 5B6

Received 23 July 2006; received in revised form 22 August 2007; accepted 31 August 2007

Abstract

We propose a method to analyze complex physical systems using two-dimensional Cell-DEVS models. These problems are usually modeled with one or more Partial Differential Equations and solved using numerical methods. Our goal is to improve the definition of such problems by mapping them into the Cell-DEVS formalism, which permits easy integration with models defined with other formalisms, and its definition using advanced modeling and simulation tools. To show this, we used two methods for solving PDEs, and deduced the updating rules for their mapping to Cell-DEVS. As our simulation results show, the accuracy of the Cell-DEVS solution is the same of these previous methods, showing that we can use Cell-DEVS as a tool to obtain numerical solution for systems of PDEs. Simultaneously, this method provides us with a simpler mechanism for model definition, automated parallelism, and faster execution.

Keywords: Finite elements method; Cell-DEVS; DEVS; Discrete-event simulation

1. Introduction

The advance in science and technology has usually relied on the definition of models representing properties of systems under study. Complex problems in the domains 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. Instead, scientists have to solve variety of problems unthinkable in the earlier stages of scientific development by using advanced numerical methods, which enabled the solution of very complex differential equations. The advent of digital computers permitted one of such methods to be widely used, the *finite element analysis* [1], which has been successfully used to analyze complex engineering and physical systems. The basis of this method is to represent the region of calculation by finite subdivisions (or finite elements). An approximate solution is then calculated over each subdivision. This is obtained by defining a solution satisfying the partial differential equation on average over a finite element, which is connected to

* Corresponding author. Tel.: +1 613 520 2600; fax: +1 613 520 5727.
E-mail address: gwainer@sce.carleton.ca (G. Wainer).

1569-190X/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.simpat.2007.08.006

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 cellular Automata (CA) [3], became popular to represent many models of real systems as cell spaces [4,5]. 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 has several problems when used in modeling complex systems. CA usually requires large amounts of compute time, mainly due to its synchronous nature. The use of a discrete time base is also a constraint to the precision of the model. The Cell-DEVS formalism [6] solved these problems by using the DEVS (Discrete EVents Systems specifications) formalism [7] to define a cell space where each cell is defined as a DEVS model. This technique permits to build discrete-event cell spaces, improving their definition by making the timing specification more expressive. Besides this, discretizing the model into a bi-dimensional grid poses constraints on the precision that can be achieved by the model. Finite element analysis, instead, is able to provide higher precision due to the characteristics of the technique.

We have explored the use of the Cell-DEVS formalism 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. The use of Cell-DEVS also enables integration with other existing DEVS [7] and Cell-DEVS models, permitting to define multi-paradigm models. An advantage of this approach is that, in real life, engineering systems are often composed of continuous and discrete components interacting together. Finally, in building these models as Cell-DEVS, we can make use of existing infrastructure, including parallel simulators and distributed environments without changes to the original models. We describe a method for mapping problems modeled by partial differential equations and solved by finite differences or FEM, into a Cell-DEVS specification. We first describe the physical system modeled, and show how to obtain the cell-update rules from an approximate solution.

The work in this paper would fit into a methodology to model and simulate complex systems using DEVS and Cell-DEVS formalisms. Complex systems usually compose of many smaller subsystems, or components. Some of these subsystems may be continuous in nature or discrete. Both types of subsystems and components can be modeled and simulated with Cell-DEVS and DEVS as shown in this paper. The methodology starts by decomposing a complex system into simpler subsystems; each could be modeled in a simple DEVS or Cell-DEVS model. It can be summarized as follows:

1. *Divide and Conquer*: Divide the modeling task of a complex system into smaller manageable tasks of modeling subsystems and components of the larger system at hand.
2. Repeat dividing these subsystems to simpler ones, until reaching components that could be modeled with single DEVS, if it has discrete behavior, or a single Cell-DEVS model if it is modeled mathematically with PDE.
3. For components with physical continuous nature, as the one presented in this paper, extract and define cell-updating rules to build Cell-DEVS models for these components. Methods of Finite elements analysis, or Finite Differences could be used for this purpose as shown in this paper.
4. Complete the definition of component models either in DEVS or Cell-DEVS, and simulate each one to validate its model.
5. Integrate the smaller models into larger subsystems. This step would be straightforward as all models are based on DEVS formalism. Then, Integrate subsystems to form the complete model for the problem under study.
6. Execute the model in a DEVS simulator, and observe system behavior.
7. If a change in design is needed, iterate through steps 2–6 until desired system behavior is obtained, and hence optimum system design is reached.

The method of modeling and simulation presented in this paper has advantages over conventional methods for solving similar problems. The most important advantage is that it enables the methodology described above that integrates models of discrete and continuous behavior, both based on DEVS formalism, into a single complex system. Very complex systems could be modeled incrementally using the above methodology. This

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

property enables vendors of engineering component to publish models for their products based on DEVS and Cell-DEVS formalisms. This would then enable engineers designing complex large systems to model their designs by integrating these smaller models into their design model, and then simulating their designs.

Another advantage of this approach over conventional simulation approaches is the ability to use different techniques to shorten the simulation runtime without changing the code or the model. This is achieved by optimizing the simulator executing the model, which is similar to a runtime environment in which models are executed. One example is the desire to execute of the simulation on a parallel computer system. Usually executing a simulation of large and complex systems needs huge computing power. Single processing systems would put a time-wise limit on simulation practicality for large models. The solution for this is to execute the simulation on a parallel computing system. However, for conventional simulation methods, this requires simulation code to be rewritten for the distributed environment. This rewriting of code is not necessarily with the above methodology. The DEVS/Cell-DEVS simulator itself has a version that is implemented as a distributed application and executes the simulation in parallel. Other possible technique for optimizing the simulator is for the simulator to cache intermediate execution results of model components, thus eliminating the need to execute some of the time consuming executions during a simulation run.

2. Background

FEM was originally created to solve problems of structural mechanics, and it was later applied to many other problems in engineering. FEM provides piecewise approximation of a partial differential equation over a continuum. A finite element is a discrete piece of that continuum. Two major components can be identified within each element: *field*, and *potential*. The field is a quantity that varies with position within structure analyzed. The potential can be thought as the driving force for the spread of the field in the material. For example, in heat transfer, temperature difference would cause a heat flux from one point to another through a material. The heat flux direction and quantity is related to the difference in temperature (temperature gradient). The heat flux would then represent the field, and the temperature represents the potential driving this field.

FEM uses a mathematical procedure for satisfying a partial differential equation over an element. By assuming the potential 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 connected together through the vertices on boundaries of each element, which are called *nodes*. Accuracy of final approximate solution would then depend on the assumed function over the element, and the number of elements in the structure. The better representation of the assumed function to the real potential distribution, the better accuracy we get. Similarly, by dividing the structure to more number of elements, we would get better accuracy for our solution. Solving the problem using FEM includes the following:

1. Divide the structure under study into a large number of simple geometry elements.
2. Represent the spatial solution in the element with a simple interpolation or shape function over the element.
3. Solve the differential equation for the element using the assumed shape function of potential.
4. As all the elements in the structure are connected through their nodes, we obtain a system of equations represented in a form of $N \times N$ matrices for the whole structure (where N represents the number of elements in that structure). These values are used to compute the unknown potential inside the structure.
5. The global equations are solved using a suitable mathematical method. The final solution gives the distribution of the potential over the structure, represented by the values obtained at the nodes of each element.

Cell-DEVS [6] is a novel approach to represent models of real life systems as cell spaces, which uses the DEVS formalism [7]. This technique permits building discrete-event cell spaces, and improves their definition by making the timing specification more expressive. In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS model. The DEVS formalism provides a framework for the construction of hierarchical modular models, allowing for model reuse, reducing development and testing times. In DEVS, basic models (called *atomic*) are specified as black boxes, and several DEVS models can be integrated together forming a hierarchical structural model (called *coupled*). DEVS not only proposes a framework for model construction, but also defines an abstract simulation mechanism that is independent of the model itself.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

A DEVS atomic model defined as

$$\text{DEVS} = \langle X, Y, S, \delta_{\text{ext}}, \delta_{\text{int}}, \lambda, ta \rangle$$

X is the input events set;
S is the state set;
Y is the output events set;
$\delta_{\text{int}} : S \to S$ is the internal transition function;
$\delta_{\text{ext}} : Q \times X \to S$, is the external transition function; where $Q = \{(s,e)/s \in S$, and $e \in [0, D(s)]\}$;
$\lambda : S \to Y$ is the output function; and
$D : S \to R_0^+ \cup \infty$ is the elapsed time function.

In the absence of external events, a DEVS model will remain in state $s \in S$ during ta(s). Transitions that occur due to the expiration of ta(s) are called internal transitions. When an internal transition takes place, the system outputs the value λ(s) $\in Y$, and changes to the state defined by $\delta_{\text{int}}(s)$. Upon reception of an external event, $\delta_{\text{ext}}(s,e,x)$ is activated using the input value $x \in X$, the current state s and the time elapsed since the last transition e. Coupled models are defined as

$$\text{CM} = \langle X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} \rangle$$

X is the set of input events;
Y is the set of output events;
$D \in N$, $D < \infty$ is an index for the components of the coupled model, and $\forall i \in D$, M_i is a basic DEVS model, where $M_i = \langle I_i, X_i, S_i, Y_i, \delta_{\text{inti}}, \delta_{\text{exti}}, ta_i \rangle$ or $M_i = \langle \text{X}_i, Y_i, D_i, \{M_d\}_i, \{I_d\}_i, \{Z_{i,j}\}_i \rangle$
I_i is the set of influences of model i, and $\forall j \in I_i$, and
$Z_{ij} : Y_i \to X_j$ is the i to j translation function.

Coupled models consist of a set of basic models (M_i, atomic or coupled) connected through the models' interfaces. Component identifications are stored into an index (D). A translation function (Z_{ij}) is defined by using an index of influences created for each model (I_i). The function defines which outputs of model M_i are connected to inputs in model M_j.

Cell-DEVS defines a cell as a DEVS atomic model and a cell space as a coupled model. Each cell of a Cell-DEVS model holds a state variable and a computing function, which updates the cell state by using its present state and its neighborhood. A Cell-DEVS atomic model is defined as

$$\text{TDC} = \langle X, Y, S, N, \text{delay}, d, \delta_{\text{int}}, \delta_{\text{ext}}, \tau, \lambda, D \rangle$$

$X \subseteq T$, $\#T < \infty \wedge T \in \{N, Z, R, \{0,1\} \cup \{\phi\}\}$; $Y \subseteq T$; $S \subseteq T$;
$\theta = \{(s, \text{phase}, \sigma\,\text{queue}, f, \sigma)/s \in S$ is the status value for the cell, phase $\in \{\text{active}, \text{passive}\}$, $\sigma\,\text{queue} = \{((v_1, \sigma_1), \ldots, (v_m, \sigma_m))/m \in N \wedge m < \infty) \wedge \alpha(i \in N, i \in [1, m]), v_i \in S \wedge \sigma_i \in R_0^+ \cup \infty\}$; $f \in T$; and $\sigma \in R_0^+ \cup \infty\}$;
$N \in S^{\eta+\mu}$; $d \in R_0^+$, $d < \infty$ (with non-deterministic behavior if $d = 0$);
δ_{int}: $\theta \to S$;
δ_{ext}: $Q \times X^b \to \theta$, $Q = \{(s,e)/s \in \theta \times N \times d$; $e \in [0, D(s)]\}$;
δ_{con}: $\theta \times X^b \to S$;
τ: $N \to S \times \{\text{inertial}, \text{transport}\} \times d$;
λ: $S \to Y^b$; and
$D : \theta \times N \times d \to R_0^+ \cup \infty$.

A cell uses a set of N input values to compute its future state, which is obtained by applying the local function τ. A delay function is associated with each cell, after which, the new state value is sent out. There are two types of delays: inertial and transport. When a transport delay is used, every value scheduled for output will be transmitted. Inertial delays use a preemptive policy: any previous scheduled output will be preempted unless its value is the same as the new computed one. After the basic behavior for a cell is defined, a complete cell space can be built as a coupled Cell-DEVS:

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

$$\text{GCC} = \langle Xlist, Ylist, I, X, Y, n, \{t_1, \ldots, t_n\}, N, C, B, Z \rangle$$

Ylist is the output coupling list;
Xlist is the input coupling list;
X is the set of external input events;
Y is the set of external output events;
n is the dimension of the cell space;
$\{t_1, \ldots, t_n\}$ is the number of cells in each of the dimensions;
N is the neighborhood set;
C is the cell space, with $C = \{C_C / c \in I \wedge C_C = \langle I_C, X_C, Y_C, S_C, N_C, d_C, \delta_{\text{int}_C}, \delta_{\text{ext}_C}, \delta_{\text{con}_C}, \tau_C, \tau_{\text{con}_C} \lambda_C, D_C \rangle\}$, where C_c is a Cell-DEVS atomic model, and $I = \{(i_1, \ldots, i_n) / (i_k \in N \wedge i_k \in [1, t_k])\ \forall k \in [1, n]\}$;
B is the set of border cells; and
Z is the translation function.

A coupled Cell-DEVS is composed of an array of atomic cells (C), each of which is connected to the cells in the neighborhood (N). The border cells (B) can have a different behavior than the rest of the space. The Z function defines the internal and external coupling of cells in the model. This function translates the outputs of mth output port in cell C_{ij} into values for the mth input port of cell C_{kl}. Each output port will correspond to one neighbor and each input port will be associated with one cell in the inverse neighborhood. The *Xlist* and *Ylist* are used for defining the coupling with external models.

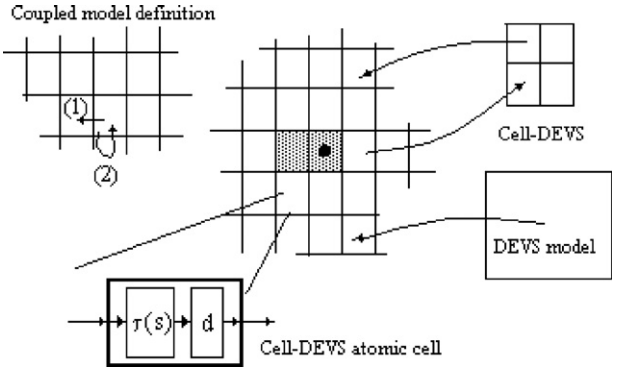
Using Cell-DEVS has different advantages. First, we have asynchronous model execution, which results in improved execution times. Timing constructions permit defining complex conditions for the cells in a simple fashion. As DEVS models are closed under coupling, seamless integration with other types of models in different formalisms is possible. The independent simulation mechanisms permit these models to be executed interchangeably in single-processor, parallel or real-time simulators without any changes.

The CD++ tool [8] was created for the simulation of DEVS and Cell-DEVS models based on their formal specifications. This version of the tool and the formalism was used to study a variety of models including traffic, forest fires, biological systems and experiments in chemistry. CD++ was built to implement DEVS and Cell-DEVS theory. The tool allows defining models according to the specifications introduced in the previous section. DEVS atomic models can be incorporated into a class hierarchy in C++, while coupled models are defined using a built-in specification language. The tool also includes an interpreter for a specification language that allows describing Cell-DEVS models.

The behavior specification of a Cell-DEVS atomic model is defined using a set of rules, each indicating the future value for the cell's state if a precondition is satisfied. The local computing function evaluates the first rule, and if the precondition does not hold, the following rules are evaluated until one of them is satisfied or there are no more rules. The behavior of the local computing function is defined using a set of rules with the form: `VALUE DELAY {CONDITION}`. These indicate that when the `CONDITION` is satisfied, the state of the cell changes to the designated `VALUE`, and its output is `DELAY`ed for the specified time. The main operators available to define rules and delays include: boolean, comparison, arithmetic, neighborhood values, time, conditionals, angle conversion, pseudo-random numbers, error rounding and constants (i.e. gravitation, acceleration, light, Planck, etc.). Fig. 2 shows the specification of a Cell-DEVS model in CD++. The specification follows Cell-DEVS coupled model's formal definitions. In this case, $Xlist = Ylist = \{\emptyset\}$. The set $\{m, n\}$ is defined by *width–height*, which specifies the size of the cell space (in this example, $m = 20$, $n = 40$). The N set is defined by the lines starting with the *neighbors* keyword. The border B is wrapped. Using this information, the tool builds a cell space, and the Z translation function following Cell-DEVS specifications. The local computing function executes very simple rules. The first one indicates that, whenever a cell state is 1 and the sum of the state values in N is 8 or 10, the cell state remain in 1. This state change is spread to the neighboring cells after 100 ms. The second rule states that, whenever a cell state is 0 and the sum of the inputs is larger or equal to 10, the cell value changes to 1. In any other case (t = true), the result remains unchanged, and it is spread to the neighbors after 150 ms.

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

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006



Q5 Fig. 1. Informal definition of Cell-DEVS [7].

```
[ex]
width : 20     height : 40    border : wrapped
neighbors : (-1,-1) (-1,0) (-1,1) (0,-1)  (0,0)  (0,1) (1,-1)  (1,0)  (1,1)
localtransition : tau-function

[tau-function]
rule: 1 100 {(0,0)=1 and (truecount=8 or truecount=10)}
rule: 1 200 {(0,0) = 0 and truecount >= 10 }
rule: (0,0) 150 { t }
```

Fig. 2. A Cell-DEVS specification in CD++.

sections will be devoted to show how Cell-DEVS can be used to describe FEM models. The idea is to describe the model in terms of cell behavior, discrete-event interaction and timing delays.

3. Mapping FEM into Cell-DEVS

3.1. One-dimensional heat transfer problem

We will show how FEM models can be mapped into Cell-DEVS using a traditional example found in [1]. This model represents steady-state heat transfer with convection from a fluid into a composite wall of different materials (resemble the heat flow through a wall of a furnace to ambient air). Heat is transferred by convection from the hot air to the 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 temperature distributions change over time.

In order to get the updating rules for each 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. Then, generalize the solution to the complete problem. In Fig. 3(a), we show two layers of the wall, which are connected 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 denoted as T_2, T_1, and T_0, as shown in Fig. 3(a). Each layer can be represented by one finite element, as showed in Fig. 3(b). Elements 1 and 2 contain two nodes, one at each end, and they are connected through their nodes, making the middle node shared between both of them as in Fig. 3(b). Every node represents a surface of a wall, and the corresponding node value represents its surface temperature. In our Cell-DEVS model, we use a cell to represent each node shown in Fig. 3.

To obtain the cell-updating rules for our Cell-DEVS model, we use the basic laws for heat transfer and energy conservation as explained in detail in the Appendix. From these mathematical manipulations, we

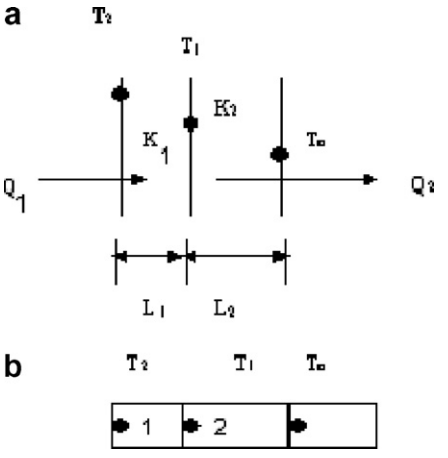


Fig. 3. (a) Two elements physically connected. (b) Elements represented as finite elements/nodes.

get Eqs. (4) and (5) that would be used as the updating rules for our Cell-DEVS model. Eq. (4) describes the heat conduction-rule inside the material. It specifies temperature of the middle node T_1 as a function of its two adjacent nodes and constant material properties. Eq. (5) describes temperature of the middle node as a function of adjacent nodes of fluid temperature T_∞, and inner node temperature T_0 inside the material. This represents the case as at a convective boundary, and T_1 is the surface temperature. Every cell value is a function of its right cell value, its 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. After the deduction procedure as shown in the Appendix we obtain the temperature formulae as follows:

$$T_1 = \frac{\frac{K_1}{L_1}T_0 + \frac{K_2}{L_2}T_2}{\frac{K_1}{L_1} + \frac{K_2}{L_2}} \quad \text{For heat conduction,} \qquad T_1 = \frac{hT_\infty + \frac{K_1}{L_1}T_0}{h + \frac{K_1}{L_1}} \quad \text{For heat convection.}$$

3.2. Two-dimensional heat transfer problem

Let us consider now how to apply the same procedure in a 2D model. We will use, as an example, a heat transfer model originally presented in [1]. This example represents a steady-state 2D heat transfer in a bar of rectangular cross section with thermal conductivity coefficient $k = 1.5\ \mathrm{W/m^2\ °C}$. Two opposite sides are kept at constant temperature of 180 °C; the third side is insulated and the other is exposed to a fluid with temperature of 25 °C and convection heat transfer coefficient $h = 50\ \mathrm{W/m^2\ °C}$. A graphical representation of the problem is depicted in Fig. 4. In order to define this model using Cell-DEVS, we define a finite grid in the bar, and we map the equations for each node on the grid into a set of rules for updating the value of a corresponding cell in a Cell-DEVS model. We have deduced the Cell-DEVS updating rules by considering two methods for the numerical solution: finite differences and finite elements.

In the Appendix, we show the full details of mathematical manipulation to deduce the following updating rules using two methods, Finite Differences and Finite Elements:

$$T_0 = \frac{T_1 + T_2 + T_3 + T_4}{4} \quad \text{Updating Rule for an internal node in the bar}$$

$$T_0 = \frac{T_1 + T_3 + 2T_2}{4} \quad \text{Updating Rule for a node on the insulated edge}$$

$$T_0 = \frac{T_1/2 + T_3/2 + T_2 + Tf \cdot h \cdot a/k}{2 + h \cdot a/k} \quad \text{Updating Rule for a node on the convective edge}$$

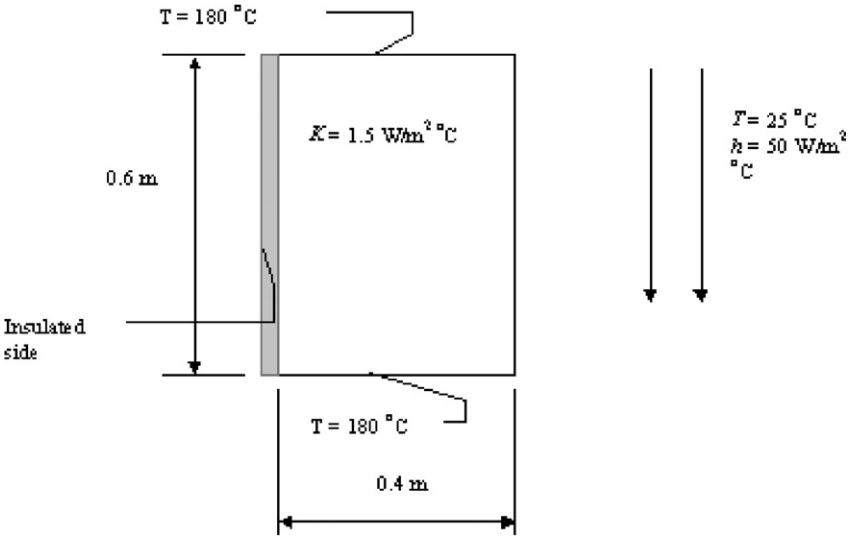


Fig. 4. Steady-state heat transfer in a long bar.

4. Cell-DEVS implementation

Once the update rules were obtained, we built the Cell-DEVS models in the CD++ environment. For the heat transfer model presented in Fig. 3, we obtained a model like the one showed in Fig. 5. Fig. 5(a) represents a composite wall of three materials with layers (numbered 1, 2 and 3). The outer temperature is $T_0 = 20$ °C. Convection heat transfer takes place on the inner surface of the wall with fluid temperature $T_4 = 800$ °C and film coefficient $h = 25$ W/m^2 °C. 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 °C, $k_2 = 25$ W/m °C, $K_3 = 20$ W/m °C, for each layer, respectively.

In Fig. 5(b), we show the structure of an equivalent Cell-DEVS coupled model for the problem defined in (a). We represent each point with temperature measure in Fig. 5(a), with a cell in the bottom row. Thus, cell (0,0), represents fluid temperature T_4, Cell (0,1) represents T_3, cell (0,2) represents T_2, cell (0,3) represents T_1, and cell (0,4) represents T_0.

Cells in row 1 store the physical properties corresponding to wall layers, as shown in Fig. 5(b). Cell (1,0) contains the value of h, (1,1) the value of (K_3/L_3), (1,2) the value K_2/L_2, cell (1,3) the value k_1/l_1, (cell (1,4)

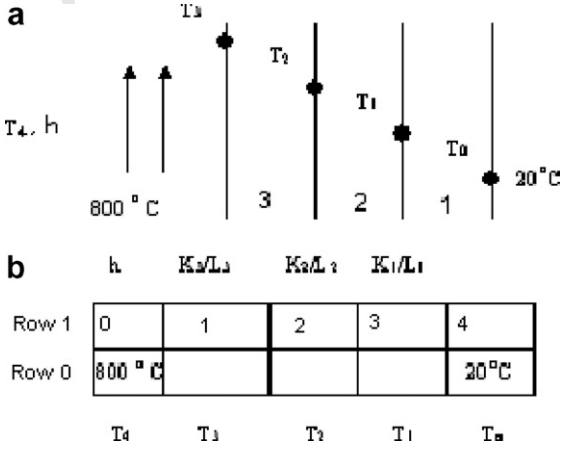


Fig. 5. (a) Steady-state heat transfer through a composite wall. (b) The problem as a Cell-DEVS space.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

value is not used in calculations). Cells in Row 1 are kept constant during the simulation, as physical properties are independent of temperature distribution. Cells (0,0) and (0,4) contain constant temperature, which constitute the boundary conditions. Each element properties are stored in a cell above it. For example, physical properties of element (0,0) would be in (−1,0), and for element (0,−1) would be in (−1,−1) of the neighborhood as in Fig. 6.

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 as defined in CD++. The first rule, [*conduction-rule*] implements Eqs. (4) and (5). Although its name implies a conduction-rule, it also works for convection as it uses film coefficient value instead of thermal conductivity at cells adjacent to fluid cells. The second rule, [*Constants*] defines the updating rule for cells in row zero. The third rule, [*Boundary*] defines the updating rule for boundary cells. These also are kept fixed during the simulation.

The specification of the Cell-DEVS coupled model is shown in Fig. 8. In this figure *heatcond* is a Cell-DEVS component, built as a bi-dimensional grid (2 rows and 5 columns) using *transport* delay. Its *border* is non-wrapped (we run specific rules for the boundary conditions in the border cells). The *neighbors* definition show the cell space neighborhood, which consists of five cells as shown in Fig. 6. *Localtransition* defines the rule used for local transition function for a cell (defined previously in Fig. 7). *Zone* defines group of cells that constitute a zone with its own updating rule that differs from the general updating rule. Here, we have defined two zones: one for the constants stored in upper row cells, and another for cells containing boundary conditions values.

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 Eq. (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 °C inside the wall. Fig. 9 shows the results for this model. The final step shows $T_3 = 304.75$ °C, $T_2 = 119.03$ °C, $T_1 = 57.14$ °C, corresponding to nodes temperatures as in Fig. 3. This same example were solved in [1] using Finite Elements Analysis and gave the following values: $T_3 = 304.6$ °C, $T_2 = 119.0$ °C, and $T_1 = 57.1$ °C.

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 consecutive iterations. This process is in fact equivalent to solving a

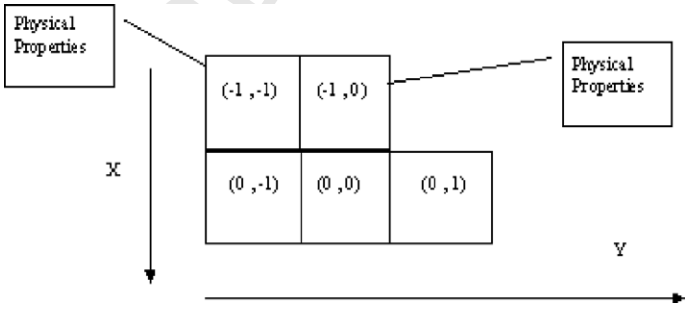


Fig. 6. Neighborhood cells.

```
[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.
```

Fig. 7. CD++ rules for 1D heat transfer model.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

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, if we have a system of n linear equations on the form $Ax = b$, and if the ith 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 $x_i = (b_i - \sum_{j \neq i} a_{i,j}x_j)/a_{i,i}$ 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 Eqs. (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 for the model shown in Fig. 8 for the problem in Fig. 5, 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. We converge into the final correct answer after 30 steps. These are 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. 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 °C. 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° 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 steps.

Finally, we initialized the cells with arbitrary values (−3000 °C). 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 (see Figs. 10–12).

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

Fig. 8. Cell-DEVS coupled model definition.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

```
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|
+-------------------------------------------------------------------+
```

Fig. 9. Simulation results for initial temperature = 20 °C.

```
Line : 30 - Time: 00:00:00:000
         0        1        2        3        4
  +-----------------------------------------------+
 0|  25.0000   66.6666   200.0000  333.3333  1.0000 |
 1| 800.0000 3000.0000 3000.0000 3000.0000 20.0000|
  +-----------------------------------------------+
...
```

```
Line : 638 - Time: 00:00:00:031
         0       1        2       3        4
  +-----------------------------------------------+
 0|  25.0000   66.6666 200.0000 333.3333   1.0000 |
 1| 800.0000  304.782  119.0754   57.1532   0.0000 |
  +-----------------------------------------------+
```

Fig. 10. Simulation results for initial temperature = 3000 °C.

```
Line : 31 - Time: 00:00:00:000
         0       1         2         3         4
   +-----------------------------------------------+
  0|  25.0000  66.6666  200.0000  333.3333   1.0000|
  1| 850.0000 304.7619 119.0476  57.1428   20.0000|
   +-----------------------------------------------+
        . . .
```

```
Line : 459 - Time: 00:00:00:020
         0         1        2         3        4
   +-----------------------------------------------+
  0|  25.0000   66.6666  200.0000  333.3333 1.0000|
  1| 850.0000  323.0113 125.3939   59.5214 20.0000|
   +-----------------------------------------------+
```

Fig. 11. Changing boundary conditions and initial temperature = 20 °C.

```
Line : 30 - Time: 00:00:00:000
         0         1         2        3       4
   +-----------------------------------------------+
  0|  25.000    66.6666  200.0000  333.33   1.0000|
  1| 850.000 -3000.000 -3000.000 -3000.00 20.0000|
   +-----------------------------------------------+
        . . .
```

```
Line : 610 - Time: 00:00:00:035
      0        1        2       3              4
   +-----------------------------------------------+
  0| 25.0000   66.6666  200.0000  333.33    1.0000|
  1| 850.0000  323.0111  125.3903   59.5213 20.0000|
   +-----------------------------------------------+
```

Fig. 12. Changing boundary conditions and arbitrary initial temperature.

4.1. Two-dimensional heat transfer problem solution with Cell-DEVS

To solve the problem shown in Fig. 4, we defined a 2D version of this model using Cell-DEVS. To do so, we divide the bar previously presented in Fig. 4 as shown in Fig. 13, which defines a grid of 6×4 (containing 7×5 nodes). Nodes are located at every lines intersection in the grid. We use the updating rules obtained previously to model this grid with Cell-DEVS in which, each cell in the model would represent a node on the grid.

Fig. 14 represents the model definition in CD++, considering that cells on the boundaries are initialized with a constant value of 180 °C.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

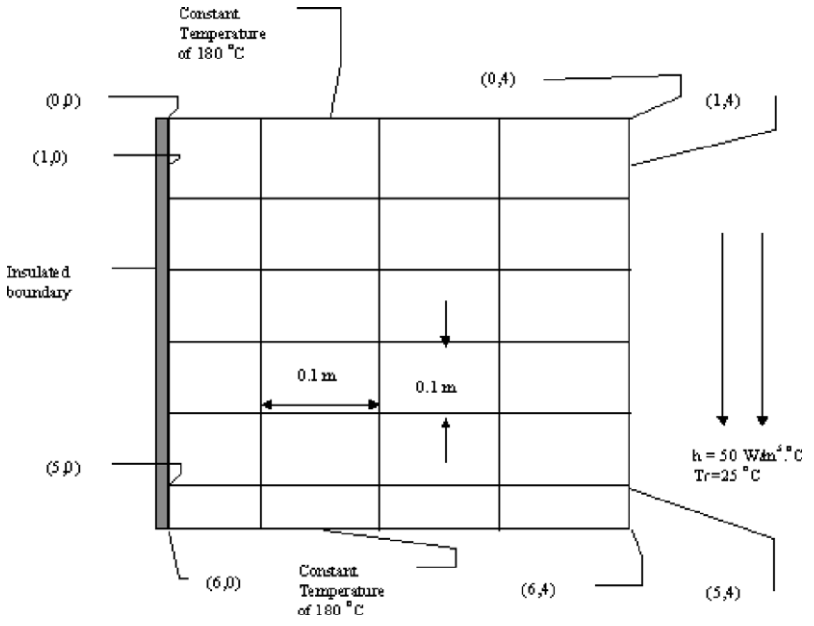


Fig. 13. 7×5 nodes grid of finite differences.

```
[heatcond]
type : cell         dim : (7,5)          delay : transport          border : nowrapped
neighbors : (-1,0) (0,0) (0,1) (1,0) (0,-1)
localtransition : conduction-rule
zone : Insulated-Boundary { (1,0)..(5,0) }
zone : Constant-Temp  { (0,0)..(0,4) }
zone : Constant-Temp { (6,0)..(6,4) }
zone : Convective { (1,4)..(5,4) }

[conduction-rule]
rule : { ((0,1)+(-1,0)+(0,-1)+(1,0)) / 4  } 1 { t }

[Insulated-Boundary]
rule : {((-1,0)+(1,0)+2*(0,1)) / 4} 1 {t}

[Constant-Temp]
rule : {(0,0)} 1 {t}

[Convective]
rule : {( ((1,0)/2)+((-1,0)/2)+(0,-1)+ (25 * (10/3)) )/(2 + (10/3)) }  1 {t}
% Fluid temperature: 25, and h.a/k is (10/3)
```

Fig. 14. Model definition in CD++ for 7×5 nodes grid.

In the figure, *heatcond* is a two-dimensional Cell-DEVS space with seven rows and five columns using `transport` delays. The border is `nowrapped`, hence, border cells are not connected to the opposite side border cells. Then, we define the neighborhood shape for each cell. `Localtransition` specifies the name used for the local computing function, and it is defined as a set of updating rules for each cell depending on its location on the grid. By default, each cell would follow the conduction-rule, unless it is located at one of the defined `zones`, which define the border conditions. Therefore, `[conduction-rule]` uses updating rule 1 defined above. Likewise, `[Insulated-Boundary]` uses updating rule 2, and `[Convective]` uses updating rule 3.

When we executed this model in CD++, we have obtained the results shown in Fig. 15. In Fig. 16, we show CD++ outputs on grid like the one in Fig. 13 for easy reading. The results shown in Fig. 15 include the sim-

```
Line : 91 - Time: 00:00:00:000
         0     1     2     3     4
    +---------------------------------+
   0|  180.0 180.0 180.0 180.0 180.0 |
   1|   20.0  20.0  20.0  20.0  20.0 |
   2|   20.0  20.0  20.0  20.0  20.0 |
   3|   20.0  20.0  20.0  20.0  20.0 |
   4|   20.0  20.0  20.0  20.0  20.0 |
   5|   20.0  20.0  20.0  20.0  20.0 |
   6|  180.0 180.0 180.0 180.0 180.0 |
    +---------------------------------+
     .  .  .
```

```
Line : 29358 - Time: 00:00:00:204
         0     1     2     3     4
    +---------------------------------+
   0|  180.0 180.0 180.0 180.0 180.0|
   1|  158.4 155.3 144.1 118.7  58.7|
   2|  143.3 138.4 122.6  92.0  42.1|
   3|  137.9 132.6 115.6  84.8  39.4|
   4|  143.3 138.5 122.6  92.0  42.1|
   5|  158.4 155.3 144.1 118.7  58.7|
   6|  180.0 180.0 180.0 180.0 180.0|
    +---------------------------------+
```

Fig. 15. Results for 7×5 nodes grid of finite differences.

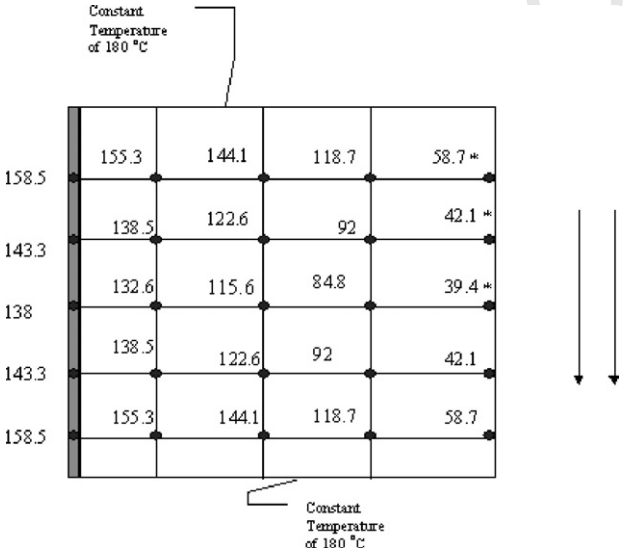


Fig. 16. 7×5 nodes grid using finite differences Cell-DEVS.

ulation time and cell's values at each time step. In this output file, each cell value is printed at its corresponding row and column. Thus, at the start of the simulation, cells $(0,0)$ to $(0,4)$ and $(6,0)$ to $(6,4)$ all had a value of 180, and the rest of the cells have a value of 20. At the final two steps, cell values do not change, causing the simulation to end (there is no cell that would be triggered by a changing neighbor). At this step, cell values would represent the solution of the problem, namely, the temperature distribution though the bar.

The type of diffusion problems as defined with previous PDE is practically solved using numerical methods such as FEM or finite differences for the reason that exact analytical solutions can only be obtained for simple geometries and boundary conditions. Such an analytical exact solution would be an ideal choice to compare precision of results obtained from CD++. Obtaining an analytical solution for a problem similar to one defined in Fig. 1, need to consider solving conduction heat transfer PDE in the bar, with applying an insulated boundary condition at one end, and a convective boundary condition at another. Solving these equations analytically is difficult and usually not practical for real problems. Due to this difficulty and to compare our method with FEM, we have compared results obtained in CD++ with those obtained using the FEM. To solve the problem with FEM, we have used a proven software tool included in [1], and used the results to validate our simulation. To show convergence of FEM and Cell-DEVS, we solved the problem with FEM twice, first with a mesh of six elements (we show the execution results in Fig. 17) and again with 192 elements with 117 distinct nodes as shown in Fig. 18. Similarly, we solved it with Cell-DEVS twice, first using 35 nodes as shown in Fig. 13, and again using 96 cells with 117 nodes as the results shown in Table 1.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

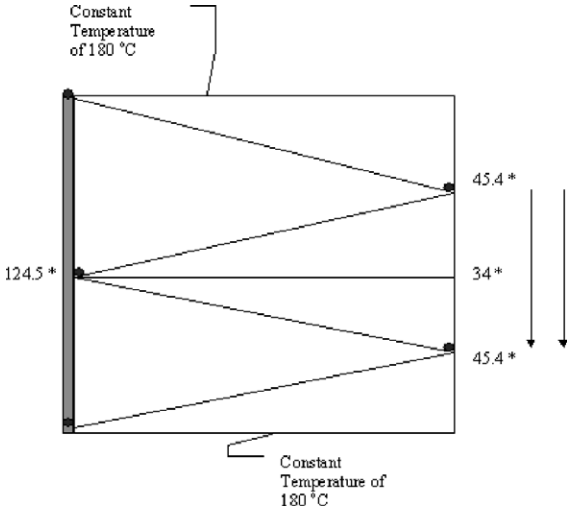


Fig. 17. Results for a mesh of six triangular finite elements.

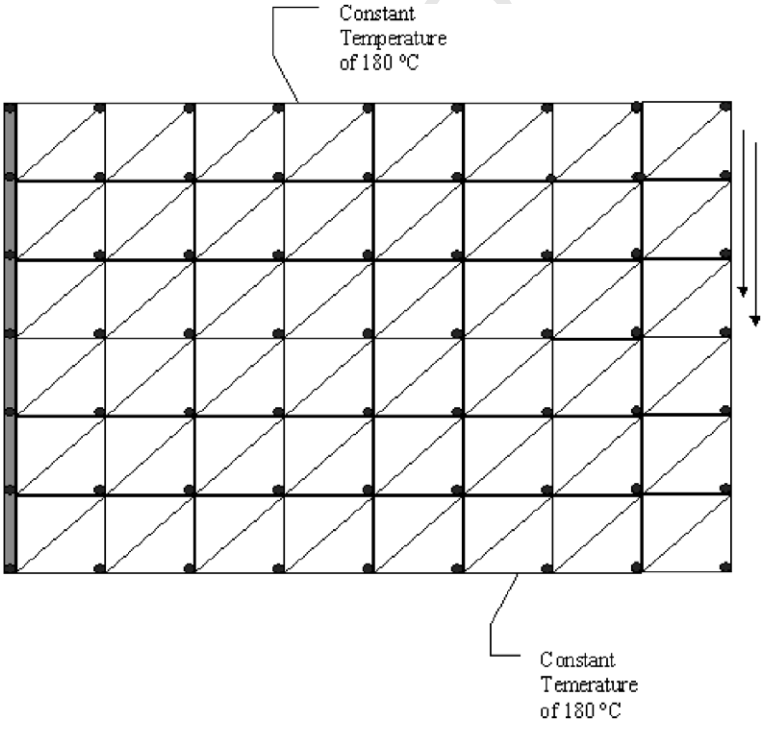


Fig. 18. Half of the bar divided to 96 triangular finite elements.

The result of FEM second simulation is shown in Table 1 for half of the bar (as it is symmetric around its middle horizontal line). Every cell in the table contains the corresponding node temperature.

To compare the results shown in Fig. 16 (35 nodes Cell-DEVS) and those in Table 1 we marked corresponding rows of nodes with an asterisk (*). Same points in the bar shown in the table and on the figure are further written in bold font in Table 1. Temperature values from Table 1 and Fig. 16 match on most nodes;

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

Table 1
Tabular form of simulation results for FEM mesh in Fig. 18

180	180	180	180	180	180	180	180	180
169	168.6	167.3	164.8	160.8	154.3	143.7	118.7	68.1
158.9	158	**155.6**	151.2	**144**	132.5	**122**	94.9	**52***
150.3	149.3	146	140.3	131.6	119.2	103.2	79	45.1
143.9	142.6	**138.9**	132.4	**122.7**	109.6	**92.9**	72.5	**37.3***
139.9	138.5	134.5	127.6	117.4	103.7	86	63.3	42.6
138.5	137.1	**133**	126	**115.7**	101.8	**84.1**	62.6	**39.6***

Table 2
96 Cell-DEVS finite differences cells with 117 nodes

180	180	180	180	180	180	180	180	180
169	168.6	167.3	165	161.1	154.9	144.1	123.5	76.9
158.9	158.1	**155.7**	151.4	**144.6**	134.1	**118.2**	93.0	**53.4***
150.3	149.3	146.0	140.3	131.6	119	101.4	77.1	45.4
143.8	142.6	**138.8**	132.3	**122.5**	109.0	**91.2**	68.6	**41.7***
139.8	138.5	134.4	127.4	117.2	103.4	85.7	64.3	40
138.5	137.1	**132.9**	125.8	**115.4**	101.5	**84**	63	**39.5***
139.8	138.5	134.4	127.4	117.2	103.4	85.7	64.3	40
143.8	142.6	138.8	132.3	122.5	109.0	91.2	68.6	41.7
150.3	149.3	146.0	140.3	131.6	119	101.4	77.1	45.4
158.9	158.1	155.7	151.4	144.6	134.1	118.2	93.0	53.4
169	168.6	167.3	165	161.1	154.9	144.1	123.5	76.9
180	180	180	180	180	180	180	180	180

409 nevertheless, there is a slight deviation between the two results for nodes on the convective side of the bar. This
410 deviation results from the fact that this side has the steepest temperature gradient, we need to divide the bar to
411 more elements to capture details of this steep gradient.
412 To enhance Cell-DEVS Finite Differences results, and to verify the conversion of the Cell-DEVS simulation
413 with that of FEM software, we ran another simulation with Cell-DEVS increasing the number of nodes to
414 13×9. The results of this simulation are shown in Table 2. The corresponding cells to those in previous solu-
415 tions (representing node temperatures) are marked with bold font for comparing the results.
416 To get the results in Table 2, we changed the model definition from the one shown in Fig. 14 to the one
417 shown in Fig. 19. We have changed model dimension and zone definitions to express the increased number
418 of cells in the new model. The results shown in Table 2 represent temperature distribution in the bar under
419 study. Each cell contains the temperature value corresponding to a node in a mesh of 12 rows and eight col-
420 umns of rectangular elements. Results of the simulation in Table 2 are symmetric around the horizontal cen-
421 terline of the bar due to its symmetry in both geometry and boundary conditions. The difference in values

```
[heatcond]
type : cell
dim : (13,9)
%delay : transport
delay : inertial
border : nowrapped
neighbors : (-1,0) (0,0) (0,1) (1,0) (0,-1)
initialvalue : 20
localtransition : conduction-rule
zone : Insulated-Boundary { (1,0)..(11,0) }
zone : Constant-Temp-Boundary { (0,0)..(0,8) }
zone : Constant-Temp-Boundary { (12,0)..(12,8) }
zone : Convective-Boundary { (1,8)..(11,8) }
```

Fig. 19. Model definition in CD++ for 13×9 nodes grid.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

between the FEM software results, our Cell-DEVS Finite Differences results in Table 1 and those in Table 2 on the convection side comes from the fact that we used a linear triangular element, which is a rough estimate for the varying field. In this element, temperature gradient is assumed constant throughout the element. This element is good to study a slow varying field over the material. Other types of elements can be used to capture the steep variation more precisely, like a quadratic triangular element. In this element, we assume the temperature gradient is linear throughout the element.

5. Conclusion

We showed how to use Cell-DEVS to model problems that traditionally have been solved with other methods as Finite Elements and Finite Differences. Modeling physical and engineering problems with Cell-DEVS have some advantages over traditional methods.

- Cell-DEVS enables building coupled systems that are composed of many atomic or coupled components. These components can simulate discrete or continuous systems. This property can be very useful when simulating engineering systems as they often compose of many continuous and discrete components and this property is an enabler to our proposed methodology as described above in this paper.
- Simulation of large and complex models can use the ability of Cell-DEVS models to execute in parallel. This can be advantageous over other methods of solving continuous systems as FEM, as their parallel execution may not be straightforward. Parallel execution of Cell-DEVS does not need to be specified by the user, but it comes as a result of the asynchronous nature of Cell-DEVS technique.
- Advantages in modeling and improved development facilities. Definition of complex equations can be easily done using the rule specification techniques of CD++, as it could be seen with the examples presented here. 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.

Engineering disciplines could benefit from using Cell-DEVS formalism to build reusable components that can be used as building blocks to build models for their products. Those models can then be executed to study system behavior. One or more components in the model can then be replaced with other alternative designs to reach a desired optimum system behavior.

When using our proposed methodology in engineering design, modeling finalized component designs (or a final product) would best be done with DEVS, as the component operating characteristics is already known. For components still under study and continuing optimization, it would be modeled with Cell-DEVS. The system would then be built using all the components and simulated for different design criteria. When optimum system performance point is reached, those components modeled with Cell-DEVS would then be in their optimum designs. The behavior of those components would be studied more to get their characteristics under normal operating conditions, and then final models for those components can be done using DEVS and integrated into the overall system model. The process could be repeated to build a large system incrementally with components possibly from different vendors as long each vendor would supply a DEVS model with their product component.

Appendix

Obtaining updating rules for one-dimensional heat transfer model

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 an 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

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

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

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 °C). $T = T(x, y)$ is the temperature field in the medium (a function in x and y), and $\partial T/\partial x$ 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

$$q = Ah(T_\infty - T_s) \tag{2}$$

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

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

where T_h, T_l are the high and low temperatures of its ends, respectively. From (3), by assuming a linear temperature distribution along the elements shown in Fig. 3, we get

$$Q_1 = K_1/L_1 \cdot (T_2 - -T_1); \quad Q_2 = K_2/L_2 \cdot (T_1 - -T_0)$$

Having the conservation of energy equation over a control volume containing both elements 1 and 2 (input heat flux equals output heat flux, and similarly, when we study two elements in which one is a convective and the other is conductive), we have

$$Q_1 = Q_2 \tag{3a}$$

thus

$$T_1 = \frac{\frac{K_1}{L_1}T_0 + \frac{K_2}{L_2}T_2}{\frac{K_1}{L_1} + \frac{K_2}{L_2}} \quad \text{for heat conduction} \tag{4}$$

and

$$T_1 = \frac{hT_\infty + \frac{K_1}{L_1}T_0}{h + \frac{K_1}{L_1}} \quad \text{for heat convection} \tag{5}$$

Obtaining updating rules for two-dimensional heat transfer model

(a) Using finite differences method [2].

With the aim of deducting equations from Finite Differences, we wrote the general partial differential equations for a steady-state heat transfer through a material, and then we approximated these equations using finite differences. We applied the basic equations of heat transfer shown earlier, to a one-dimensional element, assuming a linear temperature distribution along its length. The resulting heat flux equation is $q = k(T_h - T_l)$, where T_h, and T_l are the high and low temperatures of its ends, respectively. A steady heat transfer without heat generation in the body in 2D is represented by the following diffusion equation [2]:

$$k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = 0$$

To solve this equation, we need to get the second derivative of the temperature gradient. To do so, we study a steady-state heat transfer in a long rod as represented in Fig. 20. In this figure, we study a very small section of one-dimensional rod. Points A, B and C along the rod has corresponding temperatures of T_1, T_2, and T_3, respectively. Distances between points in the section are as indicated on the figure. To get the temperature gradient along small section, we assume a linear temperature change in x-direction over the very small finite space Δx.

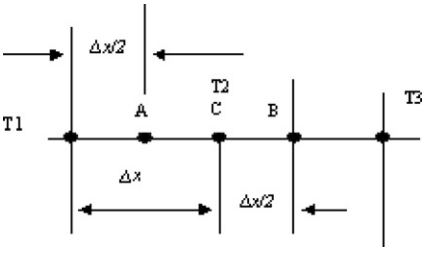


Fig. 20. Heat transfer in one dimension.

$\frac{\partial T}{\partial x} = (\frac{T_1 - T_2}{\Delta x})$ is the temperature gradient at point A, $\frac{\partial T}{\partial x} = (\frac{T_2 - T_3}{\Delta x})$ is the temperature gradient at point B, Thus, the temperature gradient at point C is: $\frac{\partial^2 T}{\partial x^2} = \frac{[(\frac{T_1 - T_2}{\Delta x}) - (\frac{T_2 - T_3}{\Delta x})]}{\Delta x} = \frac{T_1 + T_3 - 2T_2}{\Delta x^2}$.

By applying the previous result in a two-dimensional space, we can approximate the solution of the previous PDE applied to the grid in Fig. 21 as: $k[(\frac{T_1 + T_3 - 2T_0}{a^2}) + (\frac{T_2 + T_4 - 2T_0}{a^2})] = 0$ From which we obtain: $T_1 + T_2 + T_3 + T_4 - 4T_0 = 0$. This equation relates nodes temperatures of the grid. Thus, giving the updating rule for an internal node (not on any of the sides of the bar) as node 0 to be: $T_0 = \frac{T_1 + T_2 + T_3 + T_4}{4}$ *Updating Rule 1*.

These results are for a steady-state heat transfer in a homogeneous medium with constant coefficient of thermal conductivity k in all directions. We still need the updating rules for a point on the insulated surface, or on the convective side of the rod. In order to get these rules, we study the grids shown in Figs. 22 and 23, respectively.

Using a similar procedure (described in detail in [9]), we have deduced the rules for a node on the insulated boundary and on the convective boundary as follows:

$$T_0 = \frac{T_1 + T_3 + 2T_2}{4} \quad \text{Updating Rule 2}$$

We evaluate the heat balance at the two closed cells of the grid in Fig. 22 with insulated boundary, i.e. for the zone of the two closed cells. From energy conservation law, the summation of all heat fluxes entering and

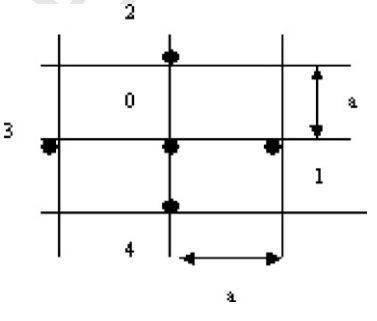


Fig. 21. Finite differences grid.

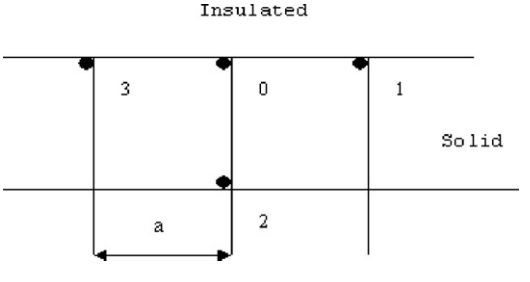


Fig. 22. Insulated boundary grid.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

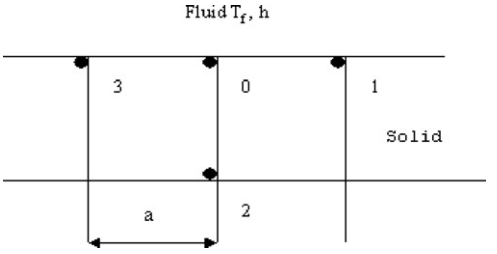


Fig. 23. Convective boundary grid.

those exiting at steady-state case would equal to zero as no energy is accumulated or lost. Similarly, by applying the heat balance equation on the two closed cells at the grid in Fig. 23, we get the updating rule for node 0 on the convective boundary as

$$T_0 = \frac{T_1/2 + T_3/2 + T_2 + Tf \cdot h \cdot a/k}{2 + h \cdot a/k} \quad \text{Updating Rule 3}$$

(b) Using finite elements method [1].

We have defined a method to deduce Cell-DEVS model updating rules from Finite Elements model for 2D problems, and we applied it to a triangular finite element with equations of steady-state heat transfer. The element shape is shown in Fig. 24. We used a linear change function of the field under study over the element. After getting the matrix equation for one element, we construct a mesh of elements interconnected, and we apply equilibrium equations, obtaining the node temperature as a function of its surrounding nodes. This function used as local computing function in Cell-DEVS model. These updating rules are used for internal nodes inside the bar. Similarly, we construct another mesh to study boundary nodes on the convective side.

The element has three nodes (each at a vertex of the triangle) each containing a value for the potential under study (for example temperature values). This type of element (called Constant Strain Triangular) was used historically to analyze body strain problems. Any internal point in the element as P can be evaluated as a function of the three nodes. We assume a linear function over the element, that is: $T_p = N_1T_1 + N_2T_2 + N_3T_3$, where N_1, N_2, N_3 are linear shape functions. With further manipulations, we can get temperature gradient inside the element as $\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = BT^e$ where $B = \frac{1}{2A}\begin{bmatrix} y_{23} & y_{31} & y_{12} \\ x_{32} & x_{31} & x_{21} \end{bmatrix}$, $T^e = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$, $y_{23} = y_2 - y_3$, $x_{32} = x_2 - x_3, \ldots,$ and A is the area of the triangular element. The final steady-state equations for all elements in a structure can be represented in a matrix format as: $\underline{K}T = \underline{R}$, where $\underline{K}$ is the global stiffness matrix of the structure, T is the node temperatures vector, and R is the heat rate vector at each node (the underline denotes a Matrix). For detailed manipulations, please refer to [1,10].

To get the global conductivity matrix $\underline{K}$ for a mesh of elements in a structure, we need to construct it from elemental k with a simple summation procedure as described in [1]. For a single element: $\underline{K_e} = kA_e\underline{B}^T\underline{B}$, where

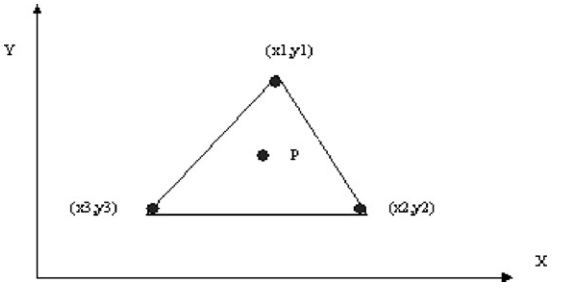


Fig. 24. Triangular element.

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

$\underline{K}_e$ is element conductivity matrix, k is the element material thermal conductivity coefficient, A_e is the element area. The final matrix K for a single element is

$$K = \frac{k}{2}\begin{bmatrix} y_{23}^2 + x_{32}^2 & y_{23}y_{31} + x_{32}x_{13} & y_{23}y_{12} + x_{32}x_{21} \\ y_{31}y_{23} + x_{31}x_{32} & y_{31}^2 + x_{13}^2 & y_{31}y_{12} + x_{13}x_{21} \\ y_{12}y_{23} + x_{21}x_{32} & y_{12}y_{31} + x_{21}x_{13} & y_{12}^2 + x_{21}^2 \end{bmatrix}$$

Q3 Therefore, for a triangular element we have

$$\frac{k}{2}\begin{bmatrix} y_{23}^2 + x_{32}^2 & y_{23}y_{31} + x_{32}x_{13} & y_{23}y_{12} + x_{32}x_{21} \\ y_{31}y_{23} + x_{31}x_{32} & y_{31}^2 + x_{13}^2 & y_{31}y_{12} + x_{13}x_{21} \\ y_{12}y_{23} + x_{21}x_{32} & y_{12}y_{31} + x_{21}x_{13} & y_{12}^2 + x_{21}^2 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} \quad (6)$$

where T_1, T_2, T_3 are node temperatures, and R_1, R_2, R_3 are initial heat rate values applied at the three nodes. We construct a mesh of elements to represent a recurring pattern inside the structure under study, as in Fig. 25.

This figure shows a mesh composed of six elements marked as $E_1, E_2, \ldots, E_6$. Those elements have their nodes numbered as shown from zero to six. The middle node 0 is shared among all the elements; thus, its value is a function of all the other elements. By studying this structure, we were able to deduce the updating rules for node 0, which were then repeat for all similar internal nodes.

We will use the following notation to name the nodes. Each of three nodes in each element is numbered *locally*, and it is referred by *global* numbers as in Fig. 25. Table 3 shows the relation between local node and global node numbers for each element (local node numbers are located at each column header, while the global node numbers are in table cells).

As a convention, local nodes for an element are numbered in counterclockwise direction. For example, the first local node of "Element 1" corresponds to the node globally numbered as 5 as in Fig. 25. The second local node corresponds to the node globally numbered 0 and its third local node corresponds to the node globally

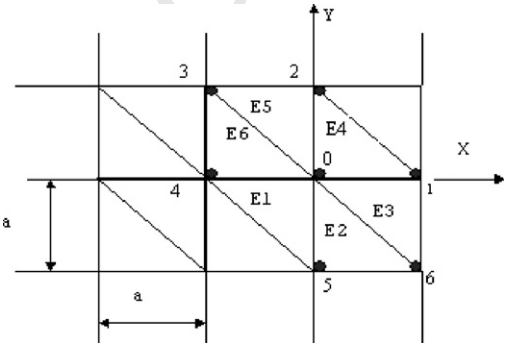


Fig. 25. Internal mesh of triangular elements.

Table 3
Mapping of local nodes to global nodes

	Local nodes		
	1	2	3
Element 1	5	0	4
Element 2	0	5	6
Element 3	6	1	0
Element 4	2	0	1
Element 5	0	2	3
Element 6	3	4	0

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

numbered 4. This mapping is shown in Table 3 for all six elements in Fig. 25. We will use this table whenever we construct the matrix equation for an element as Eq. (6) above. Using Table 3, and considering the previous mesh in Fig. 25, we get

$$\frac{k}{2}\begin{bmatrix} a^2 & -a^2 & 0 \\ -a^2 & 2a^2 & -a^2 \\ 0 & -a^2 & a^2 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} R_5 \\ R_0 \\ R_4 \end{bmatrix} \quad \text{Element 1}$$

where the first row corresponds to the local node 1 (which is globally numbered as 5, as seen in Table 3). The second row corresponds to global node 0, and the third row corresponds to global node 4:

$$\frac{k}{2}\begin{bmatrix} a^2 & -a^2 & 0 \\ -a^2 & 2a^2 & -a^2 \\ 0 & -a^2 & a^2 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} R_0 \\ R_5 \\ R_6 \end{bmatrix} \quad \text{Element 2}$$

where the first row corresponds to local node 1 (which is globally numbered as 0, as seen in Table 3), the second row corresponds to global node 5, and the third row corresponds to global node 6. Subsequently, we find the matrix equations for every other element similar to those above (due to symmetry of shape and numbering scheme for local nodes). The only difference would be the correspondence between rows of the matrix equation and different global nodes. By taking the summation of heat rate at node 0, we get

$$\begin{aligned} &-a^2T_5 + 2a^2T_0 - a^2T_4 + a^2T_0 - a^2T_5 - a^2T_1 + a^2T_0 - a^2T_2 + 2a^2T_0 - a^2T_1 + a^2T_0 - a^2T_2 - a^2T_4 + a^2T_0 \\ &= R^{(1)}0 + R^{(2)}0 + R^{(3)}0 + R^{(4)}0 + R^{(5)}0 + R^{(6)}0 \end{aligned} \tag{7}$$

where $R^{(1)}0$ is the heat rate into node 0 from element 1. In steady state, the input heat rate in any node is equal to the output from that node (i.e. $\sum_{i=1}^{i=6} R^{(i)}0 = 0$). Therefore, by solving Eq. (7) we obtain: $T_0 = \frac{T_5+T_1+T_2+T_4}{4}$, which resembles the result obtained using the finite differences method for an internal node. In order to get the updating rules for boundary conditions, we need to construct the mesh in the same way at the insulated and convective boundaries.

Obtaining node updating rules for node displacements in solid mechanics with two Degrees Of Freedom (DOF)

The previous element used for heat transfer models has only one degree of freedom (DOF) for each node (the temperature). Hence, it is sufficient to represent each node of the triangular element with one cell of Cell-DEVS model. Another example, is of solid mechanics, however, has two DOF for each node (strain in X- and Y-directions).

To solve the solid mechanics problem, we follow the same procedure to obtain the rules of the local computing function in our Cell-DEVS. In problems of solid mechanics, the potential is the displacement, and the field is the mechanical stress. Engineers study this type of problems to identify points inside the material where unsafe stresses could occur, thus causing mechanical or structural failure. We used a simple element as shown in Fig. 26, and worked as in the previous example to get element equations. Then, we built a mesh of elements and used force equilibrium equations to obtain internal node displacement as a function of surrounding nodes displacements. This would constitute our updating rules for node displacement in our Cell-DEVS model. From displacements of each node, stress inside each element can then be calculated.

In the element shown in Fig. 26, we assume a constant strain through the element. Each node has two DOF, thus it has a strain component in X-direction u_x and another in Y-direction u_y. All external forces that may act on an element (either applied by other elements, or is an external load applied on the structure under study) are assumed concentrated at the nodes.

The mesh showed in Fig. 27 represents a sub-structure that we need to examine. We build the force–strain relation for each element, and then we apply force equilibrium principle at node numbered 1 to obtain required relations between strains at node 1 as a function of surrounding nodes strains. Due to equilibrium,

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

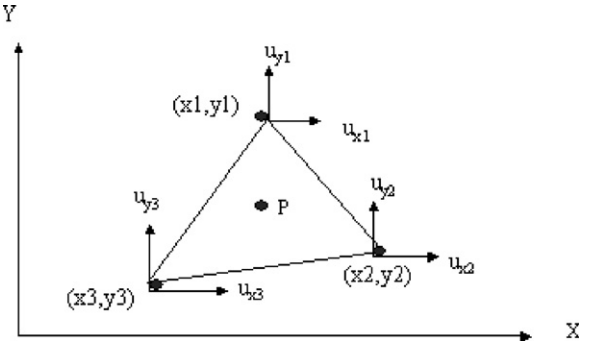


Fig. 26. Constant strain triangular element.

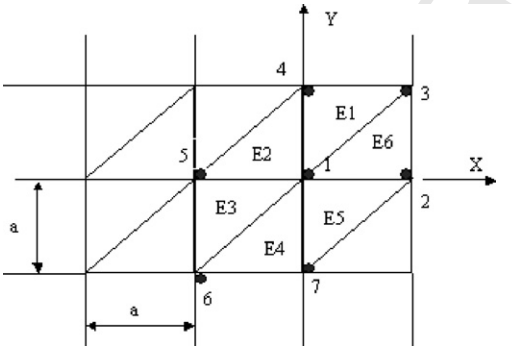


Fig. 27. Mesh of constant strain triangular elements.

summation of all forces acting on node 1 in X-direction should be zero (i.e. $\sum_{i=1}^{i=6} f_x^i = 0$). The same procedure is repeated for node 1 in Y-direction ($\sum_{i=1}^{i=6} f_y^i = 0$).

We discuss the algebraic manipulation for this model later in this Appendix to deduce the displacement at node 1 in X and the displacement of node 1 in Y-direction. Deducing these rules is done as follows: From [10], we have the following relation between stress–displacement and force on an element in a matrix form:

$$\underline{K} \cdot \underline{D} = \underline{F} \tag{1}$$

where K is the 6×6 stiffness matrix for the element, D is the 6×1 displacement vector for nodal displacements as shown in Fig. 26 and F is 6×1 vector for forces acting on each node in X- and Y-directions. From [10], $K = tAB^{\mathrm{T}}EB$, where t is element thickness; A is area of the element, $E = \frac{E}{1-v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & 1-v/2 \end{bmatrix}$. In addition, E is Young's modulus.

$$B^{\mathrm{T}} = \begin{bmatrix} y_{23} & y_{23}v & \frac{x_{32}(1-v)}{2} \\ x_{32}v & x_{32} & \frac{y_{23}(1-v)}{2} \\ y_{31} & y_{31}v & \frac{x_{13}(1-v)}{2} \\ x_{13}v & x_{13} & \frac{y_{31}(1-v)}{2} \\ y_{12} & y_{12}v & \frac{x_{21}(1-v)}{2} \\ x_{21}v & x_{21} & \frac{y_{12}(1-v)}{2} \end{bmatrix}$$

$$K=\frac{tE}{4A(1-v^2)}\begin{bmatrix} y_{23}^2+x_{32}^2\frac{1-v}{2} & y_{23}x_{32}\frac{1+v}{2} & y_{23}y_{31}+x_{32}x_{13}\frac{1-v}{2} & y_{23}x_{13}v+x_{32}y_{31}\frac{1-v}{2} & y_{23}y_{12}+x_{32}x_{21}\frac{1-v}{2} & y_{23}x_{21}v+x_{32}y_{12}\frac{1-v}{2} \\ y_{23}x_{32}\frac{1+v}{2} & x_{32}^2+y_{23}^2\frac{1-v}{2} & x_{32}y_{31}v+y_{23}x_{13}\frac{1-v}{2} & x_{32}x_{13}+y_{23}y_{31}\frac{1-v}{2} & x_{32}y_{12}v+y_{23}x_{21}\frac{1-v}{2} & x_{32}x_{21}+y_{23}y_{12}\frac{1-v}{2} \\ y_{23}y_{31}+x_{32}x_{13}\frac{1-v}{2} & y_{31}x_{32}v+x_{13}y_{23}\frac{1-v}{2} & y_{31}^2+x_{13}^2\frac{1-v}{2} & y_{31}x_{13}v+x_{13}y_{31}\frac{1-v}{2} & y_{31}y_{12}+x_{13}x_{21}\frac{1-v}{2} & y_{31}x_{21}v+x_{13}y_{12}\frac{(1-v)}{2} \\ x_{13}y_{23}v+y_{31}x_{32}\frac{1-v}{2} & x_{13}x_{32}+y_{31}y_{23}\frac{1-v}{2} & x_{13}y_{31}v+y_{31}x_{13}\frac{1-v}{2} & x_{13}^2+y_{31}^2\frac{1-v}{2} & x_{13}y_{12}v+y_{31}x_{21}\frac{1-v}{2} & x_{13}x_{21}+y_{31}y_{12}\frac{1-v}{2} \\ y_{12}y_{23}+x_{21}y_{32}\frac{1-v}{2} & y_{12}x_{32}v+x_{21}y_{23}\frac{1-v}{2} & y_{12}y_{31}+x_{21}x_{13}\frac{1-v}{2} & y_{12}x_{13}v+x_{21}y_{31}\frac{1-v}{2} & y_{12}^2+x_{21}^2\frac{1-v}{2} & y_{12}x_{21}v+x_{21}y_{12}\frac{1-v}{2} \\ x_{21}y_{23}v+y_{12}x_{32}\frac{1-v}{2} & x_{21}x_{32}+y_{12}y_{23}\frac{1-v}{2} & x_{21}y_{31}v+y_{12}x_{13}\frac{1-v}{2} & x_{21}x_{13}+y_{12}y_{31}\frac{1-v}{2} & x_{21}y_{12}v+y_{12}x_{21}\frac{1-v}{2} & x_{21}^2+y_{12}^2\frac{1-v}{2} \end{bmatrix}$$

Therefore, the stiffness matrix for each element of Fig. 27 would be

$$K=\frac{tE}{4A(1-v^2)}\begin{bmatrix} a^2+a^2\frac{(1-v)}{2} & -a^2\frac{(1+v)}{2} & -a^2\frac{(1-v)}{2} & a^2v & -a^2 & a^2\frac{(1-v)}{2} \\ -a^2\frac{(1+v)}{2} & a^2+a^2\frac{(1-v)}{2} & -a^2\frac{(1-v)}{2} & -a^2 & a^2v & -a^2\frac{(1-v)}{2} \\ -a^2\frac{(1-v)}{2} & a^2\frac{(1-v)}{2} & a^2\frac{(1-v)}{2} & 0 & 0 & -a^2\frac{(1-v)}{2} \\ a^2v & -a^2 & 0 & a^2 & -a^2v & 0 \\ -a^2 & a^2v & 0 & -a^2v & a^2 & 0 \\ a^2\frac{(1-v)}{2} & -a^2\frac{(1-v)}{2} & -a^2\frac{(1-v)}{2} & 0 & 0 & a^2\frac{(1-v)}{2} \end{bmatrix}$$

$A=$ Area of element$=\frac{1}{2}(x_{13}y_{23}-x_{23}y_{13})$. Displacement D and forces F vectors would be

$$D=\begin{bmatrix} u_{x_1} \\ u_{y_1} \\ u_{x_2} \\ u_{y_2} \\ u_{x_3} \\ u_{y_3} \end{bmatrix},\quad F=\begin{bmatrix} f_{x_1} \\ f_{y_1} \\ f_{x_2} \\ f_{y_2} \\ f_{x_3} \\ f_{y_3} \end{bmatrix}$$

By applying the previous matrix equation for each element in the mesh in Fig. 27, we obtain six equations for node 1. From which, we reach the updating rule for the strains in node 1 as a function of strains in neighboring nodes. From the equilibrium of forces in X-direction we have $\sum_{i=1}^{i=6} f_x^i=0$ at node 1, that is total of forces from each element in X-direction is zero. Similarly, we have in Y-direction $\sum_{i=1}^{i=6} f_y^i=0$. From Eq. (1), we construct the matrix equation for each of the 6 elements, and we then get the forces affecting at node 1 from each element as

$$f_x^1=\frac{tE}{4A(1-v^2)}\left[-a^2\frac{1-v^2}{2}u_{x_4}+a^2\frac{1-v}{2}u_{y_4}+a^2\frac{1-v}{2}u_{x_1}-a^2\frac{1-v}{2}u_{y_3}\right]$$

where f_x^1 represents force from element 1, u_{x_4} is displacement in X-direction of node 4. Similarly, u_{y_4} is the displacement of node 4 in the X-direction:

$$f_x^2=\frac{tE}{4A(1-v^2)}\left[a^2\frac{3-v}{2}u_{x_1}-a^2\frac{1+v}{2}u_{y_1}-a^2\frac{1-v}{2}u_{x_4}+a^2vu_{y_4}-a^2u_{x_5}+a^2\frac{1-v}{2}u_{y_5}\right]$$

For element 2:

$$f_x^3=\frac{tE}{4A(1-v^2)}[-a^2u_{x_5}+a^2vu_{y_5}-a^2vu_{y_6}+a^2u_{x_1}]\quad\text{Element 3}$$

$$f_x^4=\frac{tE}{4A(1-v^2)}\left[-a^2\frac{1-v}{2}u_{x_7}+a^2\frac{1-v}{2}u_{y_7}+a^2\frac{1-v}{2}u_{x_1}-a^2\frac{1-v}{2}u_{y_6}\right]$$

$$f_x^5=\frac{tE}{4A(1-v^2)}\left[a^2\frac{3-v}{2}u_{x_1}-a^2\frac{1+v}{2}u_{y_1}-a^2\frac{1-v}{2}u_{x_7}+a^2vu_{y_7}-a^2u_{x_2}+a^2\frac{1-v}{2}u_{y_2}\right]$$

$$f_x^3=\frac{tE}{4A(1-v^2)}[-a^2u_{x_2}+a^2vu_{y_2}-a^2vu_{y_3}+a^2u_{x_1}]$$

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006

With algebraic manipulations and by applying the equilibrium condition $\sum_{i=1}^{i=6} f^i x = 0$, we reach the updating rule for the displacement at node 1 in X-direction to be

$$u_{x_1} = \frac{2u_{x_4}(1-v) + 2u_{y_1}(1+v) + (1+v)(u_{y_3} + u_{y_6}) + 4(u_{x_5} + u_{x_2}) + u_{x_7}(1-v) - (1-v)(u_{y_4} + u_{y_5} + u_{y_7} + u_{y_2})}{2(5-2v)}$$

Similarly, we get the displacement of node 1 in Y-direction as

$$u_{y_1} = \frac{2u_{x_1}(1+v) + 4u_{y_4} + u_{x_2}(1+v) + 2u_{y_5}(1-v) + u_{x_6}(1+v)4u_{y_7} + 2u_{y_2}(1-v) - (1+v)(u_{x_4} + u_{x_5} + u_{x_7} + u_{x_2})}{2(5-2v)}$$

References

[1] T. Chandrupatla, A. Belegundu, Introduction to Finite Elements in Engineering, second ed., Prentice Hall, Upper Saddle River, NJ, c1997.
[2] D.H. Bacon, Basic Heat Transfer, Butterworth & Co. Publishers Ltd., 1989.
[3] S. Wolfram, A New Kind of Science, Wolfram Media, Inc., 2002.
[4] M. Sipper, The emergence of cellular computing, IEEE Computer (July) (1999) 18–26.
[5] D. Talia, Cellular processing tools for high-performance simulation, IEEE Computer (2000) 44–52.
[6] G. Wainer, N. Giambiasi, *N*-dimensional Cell-DEVSDiscrete Events Systems Theory and Applications, vol. 12, No. 1, Kluwer, 2002, pp. 135–157.
[7] B. Zeigler, T. Kim, H. Praehofer, Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, Academic Press, 2000.
[8] G. Wainer, CD++: a toolkit to define discrete-event models, in: G. Wainer (Ed.), Software Practice and Experience, vol. 32, No. 3, Wiley, 2002, pp. 1261–1306.
[9] H. Saadawi, G. Wainer, Defining models of complex 2D physical systems using Cell-DEVS, Technical Report SCE-04-04, Carleton University, 2004.
[10] Y. Liu, Lecture Notes: Introduction to the Finite Element Method. <http://urbana.mie.uc.edu/yliu/FEM-525/FEM_Lecture_-Notes_Liu_UC.pdf> (accessed April 2003).

Please cite this article in press as: H. Saadawi, G. Wainer, Modeling physical systems using finite element Cell-DEVS, Simulat. Modell. Pract. Theory (2007), doi:10.1016/j.simpat.2007.08.006