A Flow Injection Model Using Cell-DEVS

Alejandro Troccoli
Javier Ameghino

Departamento de Computación, Pabellón I. Ciudad Universitaria (1428). Buenos Aires. Argentina.

Fernando Iñón

INQUIMAE, Universidad de Buenos Aires. Pabellón II. Ciudad Universitaria (1428). Buenos Aires. Argentina.

Gabriel Wainer

Department of Systems and Computing Engineering, Carleton University. 1125 Colonel By Drive, K1S 5BE. Ottawa, Canada

e-mail: gwainer@sce.carleton.ca

Abstract

Cell-DEVS is an extension to the DEVS formalism that allows the definition of cellular models. Complex physical systems can be defined using simple rules, reducing the development. We present the definition of a model of flow injection using Cell-DEVS. The simulation validation results showed a margin of error within the expected values for the experiment, showing how to employ the formalism in analyzing physical systems.

1. Introduction

Simulation is a powerful tool to understand complex physical systems. Simulation models of complex physical systems have been developed for years, generally using difference equations (see, for instance, [1, 2]). In recent years, many simulation models of real systems have been represented as cell spaces [3, 4]. Cellular Automata [5] is a well-known formalism to describe these systems. Cellular automata are defined as an 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).

Cellular automata usually require large amounts of compute time, mainly due to its synchronous nature. The use of a discrete time base is also a constrain to the precision of the model. Timed Cell-DEVS solves these problems by using the DEVS (Discrete EVents Systems specifications) formalism [6] to define a cell space where each cell is defined as a DEVS model [7]. The goal is to build discrete-event cell spaces, improving their definition by making the timing specification more expressive. DEVS formalism was proposed to model discrete events systems. A DEVS model is built using a set of behavioral models called **Atomic**, which can be combined to form **Coupled** ones. A DEVS atomic model is defined as:

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >`.$$

Input external events in ***X*** are received in input ports. When an event arrives, the model executes the external transition function δ_{ext} to produce a state change. Each state has an associated lifetime **ta**. When this time is consumed the internal transition function δ_{int} is activated to produce internal state changes. The internal state **S** can be used to provide model outputs ***Y***, which are sent through the output ports. They are sent by the output function λ, which executes before the internal transition.

A DEVS coupled model is defined as:

$$CM = < X, Y, D, \{Mi\}, \{I_i\}, \{Z_{ij}\} >.$$

Each coupled model consists of a set of **D** basic models $\mathbf{M_i}$ connected through input/output ports. The list of influencees $\mathbf{I_i}$ of a given model is used to determine the models to which outputs must be sent. These sets are used to build the translation function $\mathbf{Z_{ij}}$, in charge of translating outputs of a model into inputs for the others. An index of influencees is created for each model (I_i). For every *j* in the index, outputs of model M_i are connected to inputs in model M_j.

In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS using transport or inertial delays.

Each cell is seen as having a set of N inputs to compute its future state. Each input (generally received from the neighboring cells) is received through the model's interface, and is used to activate the local function. A delay can be associated with each cell, allowing deferring the transmission of the execution results. A **transport** delay allows us to model a variable commuting time for each cell with anticipatory semantics (every scheduled event is executed). Using **inertial** delays, the semantics is preemptive: some scheduled events are not executed due to a small interval between two input events. Therefore, the outputs of a cell are not transmitted instantaneously, but after the consumption of the delay. The model advances through the activation of the internal, external, output and state's duration functions, as in other DEVS models.

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 τ. The new value of the cell is transmitted to the neighbors after the consumption of the delay function. **Delay** defines the kind of delay for the cell, and **d** its duration. This behavior is defined by the $\boldsymbol{\delta_{int}}$, $\boldsymbol{\delta_{ext}}$, λ and **D** functions.

Once each cell is defined, they can be put together to form a coupled model composed of an array of atomic cells. Each of them is connected to its neighborhood. As the cell space is finite, the borders should be provided with a different behavior than the rest of the space. Otherwise, the space can be defined as wrapped, meaning that cells in a border are connected with those in the opposite one.

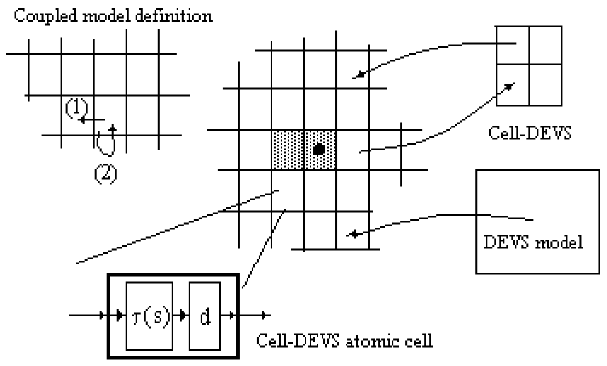

Figure 1. Informal Definition of Cell-DEVS [8].

A Cell-DEVS coupled model is defined by:

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

A cell space **C** defined by this specification 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. This function translates the outputs of output port *m* in cell *Cij* into values for the *m* input port of cell *Ckl*. The input/output coupling lists can be used to interchange data with other models.

The **CD++** tool [9] was developed following the formal definitions of the Cell-DEVS formalism. CD++ is a tool to simulate both DEVS and Cell-DEVS models. 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. The behavior of a cell (the τ function of the formal specification) is defined using a set of rules of the form:

VALUE DELAY CONDITION

When an external event is received, the rule evaluation process is triggered to calculate the new cell value. Starting with the first listed rule, the *CONDITION* is evaluated. If it is satisfied, the new cell state is obtained by evaluating the *VALUE* expression. The cell will change to this new state after a *DELAY* time, and when it changes, it sends output messages to all its neighbors. If the condition is not valid, the next rule is evaluated repeating this process until a rule is satisfied. If no rule *CONDITION* statement is satisfied, the simulation is aborted.

The specification language has a wide range of functions and operators. The most common operators are included: boolean (*AND*, *OR*, *NOT*, *XOR*, *IMP* and *EQV*), comparison (=, !=, <, >, <= and >=), and arithmetic (+, -, * and /). In addition, different types of functions are available: trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, G.C.D. and L.C.M. Other existing functions allow to check if a number is integer, even, odd or prime. Also, some common constants are defined. Cellular models can be integrated to a standard DEVS hierarchy. Therefore, input/output ports can be defined for the cell space.

In [7, 8], we showed the usefulness of the formalism by introducing different models of complex physical systems using Cell-DEVS. In this case we have focused in the solution of a flow injection analysis problem. The idea was to depart from the standard discrete Cellular Automata models, and use an approach based on Cell-DEVS. The idea is to describe the model behavior in terms of cell behavior, discrete event interaction and timing delays related to the model. The following section introduces the problem to be solved. Then, we show the definition of this model using Cell-DEVS and the related tools. Finally, we present simulation results of these experiences.

2. Flow Injection Analysis (FIA)

Flow-injection methods are analytical methods used for automated sample analysis of liquid samples. In a flow injection analyzer, a small, fixed volume of a liquid sample is injected as a discrete zone using an injection device into a liquid carrier, which flows through a narrow tube. As a result of convection at the beginning, and later of axial and radial diffusion, this sample is progressively dispersed into the carrier as it is transported along the tube. The addition of reagents at different confluence points (which mix with the sample as a result of radial dispersion) produces reactive or detectable species, which can be sensed by flow-through detection devices. The following figure presents a simple flow-injection apparatus.

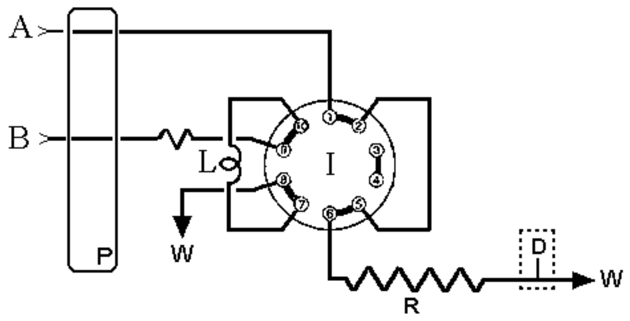

Figure 2. A FIA manifold. P: pump; A,B: carrier and reagent lines; L: sample injection; I: injection valve; R: reactor coil; D: flow through detector; W: waste line.

This device (called a FIA manifold) consists of a peristaltic pump (P) that adds carrier solution (A) into a valve (I) that connects to a tube-shaped reactor (R). At the end of the tube a detector (D) is placed to sense a specific property of the flowing solution. The valve can be turned to allow the flow of the sample (B, C) into the reactor. The sample is held in the loop (L) and when the valve is rotated, its content flows into the reactor, where chemical activity will usually take place between the sample and the carrier solution. As a result, a change will be observed in the signal produced by a detector (D), making it possible to quantify the sample after comparing the results with those obtained by known samples.

In a flow injection system, convective transport yields a parabolic velocity profile with molecules at the tube walls having speed zero and those at the center having twice the average velocity. At the same time, the presence of concentration gradients develops axial and radial diffusion of sample molecules. It has been reported that in FLOW INJECTION systems of practical interest, axial molecular diffusion has almost no influence in the overall dispersion, but radial diffusion is the main contributor. For a pump proving a net flow of *q* ml/min in a coil of radius *a*, the average flow velocity is given by:

$$V_a = \frac{q}{60 \cdot (p \cdot a^2)} \qquad \textbf{(Equation 1)}$$

At a point at distance *r* from the center, the flow velocity is described by:

$$v(r) = 2 \cdot V_a \cdot \left(1 - \frac{r^2}{a^2}\right) \qquad \textbf{(Equation 2)}$$

As mentioned in [12], it is very difficult, if not impossible, to correlate the experimentally obtained response curve with the actual spatial mass distribution of the system. This is a consequence of the selected method of measurement, which fixes spatially and temporally the point of detection. Under these circumstances, any event occurred before the detection point is inferred from the response curve profile. Therefore, this detection approach is a powerful tool for predicting response curves, but ignores the processes leading to the generation of such response.

In [12] a method for continuously monitoring a flow injection system was proposed. A flow injection system using nitric acid as the carrier solution, water as the injected sample and a digital conductimeter with a couple of wires at both ends of the carrier stream detector was used to follow the radial mass distribution of the sample zone (Figure 3).

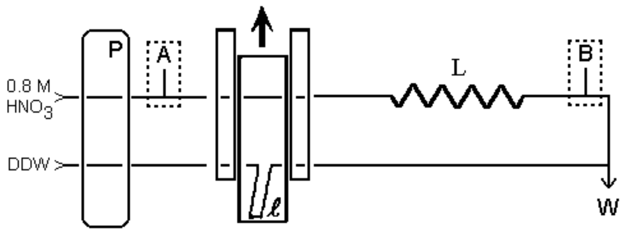

Figure 3. FIA manifold for continuously monitoring. P = pump; l = loop; L = reactor; W = waste; A, B = detection points. Punctual detection: suitable detector in point B; integrated detection: Pt wires located at points A-B.

While the system is in the position shown in Figure 3, only the carrier solution flows into the reactor L. The water sample is injected by sliding the valve V upward, acting as a blocking disc. At this point, no electric conductance is measured between points A and B. As convective transport and diffusion gradient forces the water sample to be released from the walls inside the reactor L, a reduction of the blocking area is produced. This allows electric current to flow, enabling measuring conductivity values different from zero. Figure 4 shows the characteristic conductivity curve obtained by such a system.

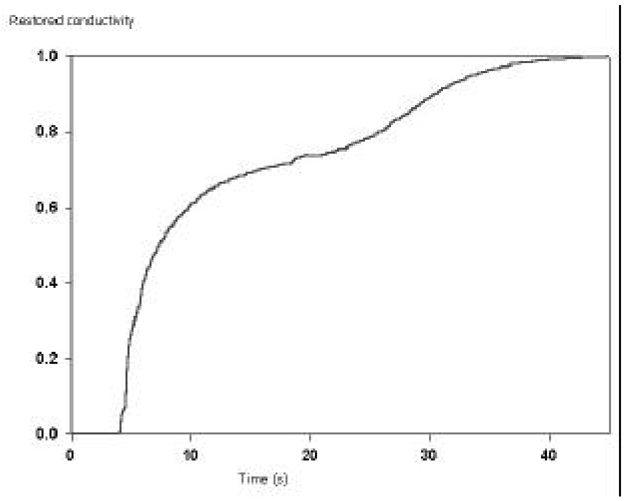

Figure 4. Characteristic conductivity curve [12].

The original work used Random Walk simulation to approximate the differential equations of this system [12]. Using this technique, the initial coordinates of the molecules in the loop are assigned at random. The coordinates of each molecule are computed using Brownian movement and longitudinal transport. The loop is divided in cells in order to compute the concentration, which is used to compute the conductivity. The next position of each particle is computed using a differential equation and a random variable.

Cell-DEVS is perfectly tailored for this kind of application, and can improve the development cycle of this kind of problems. It also enables making more complex experiments using simple rules. The n-dimensional features of Cell-DEVS, and the ability of including different delays for each of the cells can improve the definition of the FIA system. In the following section we will show how the FIA model can be implemented using Cell-DEVS.

3. Cell-DEVS model for a FIA system

As mentioned, it is impossible to analyze the detailed behavior of the changes in the mass distribution profile. Therefore, we decided to build a Cell-DEVS model describing the integrated conductivity flow-injection system (ICM) in detail. In this way, the internal complex behavior can be analyzed by studying the simulated results. The ICM system consists of a 0.025 cm radius tube, a 10.75 cm loop and a 9,25 reactor coil. We assumed the total tube length of the tube to be of 20cm. For this system, a cell space of 25 rows and 200 columns was defined, each cell representing a 0.001 x 0.1cm of a half tube section. Row 0 represents the center of the tube and row 24 the section of the tube touching its walls and the value of each cell will represent the nitric acid concentration.

Tube wall

			Row 24
...	...	......	
Center			Row 0
...	...	...	...

Tube wall

Figure 5. Correspondence between the cell-space and the actual tube

Figure 5 shows in light gray a tube section representing a cell. This is a longitudinal cut of the tube. The final aim is to build a 3 dimensional space representing a cylindrical section of the tube, but in this case each cell represent a flat section.

To deal with convective transport and radial diffusion at the same time, the model reacts in two phases: transport and diffusion. The local computing function simulates the transport phase, and all cells are connected to an external generator sending an event, which triggers the diffusion phase. The model is built as a coupled DEVS model with two components: a Cell-DEVS (named *fia*) representing the tube, and an atomic model (named *generator*). The generator has one output port (*out*) to send the diffusion triggering event. This port is mapped to the *diffuse* input port of the *fia* model (line 2). This means all output events sent through the *out* port will be received as external events by the *fia* model through the *diffuse* port.

```
00 [Top]
01 components :  fia generator@ConstGenerator
02 link : out@generator diffuse@fia
03
04 [generator]
05 frequency : 00:00:00:014
```

Figure 6. Components of the DEVS model

The frequency of diffuse events is defined by Equation 3. This equation computes the characteristic distance a particle of a given solution of diffusion coefficient *c* will travel in *dt* seconds.

$$ds = \sqrt{2 \cdot c \cdot dt} \qquad \textbf{(Equation 3)}$$

Solving the equation for $c = 3{,}5 \times 10^{-5}$ cm/s and ds = 0.001 cm, we obtain a dt of 14ms. We used for the ds value the cell height to find out how long it would take for two cells to diffuse homogeneously. We did not take into account the cell width because axial diffusion can be ignored.

```
05 [fia]
06 in : diffuse
07 width : 200
08 height : 25
09 delay :   inertial
10 border :   nowrapped
11 neighbors : fia(-1,-1) fia(-1,0) fia(-1,1)
12 neighbors : fia(0,-1) fia(0,0) fia(0,1)
13 neighbors : fia(1,-1) fia(1,0) fia(1,1)
14 localtransition :  transport
```

Figure 7. Definition of the FIA coupled cell model

Figure 7 shows the definition of the parameters for the coupled Cell-DEVS *fia*. Line 6 defines the *diffuse* input port, and lines 7 and 8 define the cell space dimensions. Line 9 sets the cell delay type to inertial. An inertial delay cell that has a scheduled future value f will preempt this value if upon receiving an external event and evaluating the local transition rules a new future value f_1,with $f \neq f_1$, is obtained. In this case, f_1 will be scheduled as the future value with a given delay d. Line 10 defines non-wrapped borders and lines 11 to 13 define a cell's neighborhood shape. Finally, line 14 defines the sets the local transition function rules. The behavior of each cell is defined by this function, defined in Figure 8.

```
18 [transport]
19 rule : { (0,-1) } { 0.1 / ( 22.57878 * ( 1 -
power( cellPos(0) * 0.001 + 0.0005 , 2)
 / 0.000625 )) * 1000 } { cellPos(1) != 0 }
20 rule : { 0.8 } { 0.1 / ( 22.57878 * ( 1 -
power( cellPos(0) * 0.001 + 0.0005 , 2) /
 0.000625 )) * 1000 } { cellPos(1) = 0 }
```

Figure 8. Definition of the border cells.

The convective transport has arbitrarily been defined in the direction of increasing column values, so that in visual representations the carrier will be seen flowing from left to right. Being this the case, a local transition rule for the transport phase should set a cell's value to the current value of its (0,-1) neighbor cell. The rate at which this is done depends on the velocity of the flow at the cell, which, as mentioned before, has its maximum at the center of the tube and decreases towards its walls. This is stated in the first transport rule in line 19. As mentioned in section 2, a local transition rule has three components, a value, a delay and a condition. For this rule, this components are:

Value: `{(0,-1)} //The cell's left neighbor`

Delay: `{ 0.1 / (22.57878 * ( 1 - power( cellPos(0) * 0.001 + 0.0005, 2) / 0.000625)) * 1000 }`

Condition: `{ cellPos(1) != 0 }`

The delay is calculated using equations 1 and 2. For a pump with a constant flow of 1,33ml/min, the average speed is 11,29 cm/s. This value can be substituted in equation 2 and multiplied by 2 to yield the number 22.57878 shown in the delay expression. In addition, for equation 2 to be solved, we also need to know the distance to the center of the tube. CD++ provides a built in function called *cellPos* that returns a requested coordinate of the cell whose value is being sought. For a 2 dimensional model, *cellPos(0)* returns the cell's row. Consequently,

```
cellPos(0) * 0.001 + 0.0005
```

is the distance of the center of the cell to the center of the tube and therefore,

```
( 22.57878 * ( 1 - power( cellPos(0) * 0.001 +
0.0005 , 2) / 0.000625 ))
```

is the solution to equation 2, for a = 0.025 cm. Having the velocity of flow $v(r)$, the delay will be the time in milliseconds for a particle moving at speed $v(r)$ cm/s to travel across a 0.1 cm cell. This time is given by the expression

`0.1 /` $v(r)$ * 1000

concluding our explanation for the delay component of the rule.

The generic rule we have just given is only valid for all cells that have a valid (0,-1) neighbor. The left border cells (those in column 0) do not satisfy this prerequisite, stated in the condition component `cellPos(1) != 0`, and should therefore have a different rule.

The rule in line 20 is the rule for the left border cells. It simply states that for these cells the new value should be 0.8, which corresponds to the concentration of the carrier solution being pumped into the tube.

Table 1 shows the results of applying equation 2 to calculate the delays for each row. It is important to notice that some adjacent rows have different delay values, as is the case of rows 2 and 3. This might lead to the presumption that the convective transport behavior will not be preserved due to an early preemption a cell's scheduled future value. This is not the case, as we will show.

Table 1 – Calculated delays for each row

Row	Delay (ms)
0	4
1	4
2	4
3	5
4	5
5	5
6	5
7	5
8	5
9	5
10	5
11	6
12	6
13	6
14	7
15	7
16	8
17	9
18	10
19	11
20	14
21	17
22	23
23	38
24	112

When the simulation starts at time 0, all cells will evaluate their local transition functions and schedule their next change. A cell in row 2 will schedule an internal transition at time t = 4ms and a cell in row three at t = 5ms. So at time t = 4ms, all cells in row 2 will send an output event to their neighbors. Cells in row 3 will receive this event and evaluate the local transition function, which says they should take the value of their left neighbor. But their left neighbor has not changed yet, so the new value will be the same as the previous *future value*. Therefore, they will keep their scheduled internal transition for t = 5 ms. At this time, all cells in row 2 with a scheduled internal transition will send their new value to their neighbors. A cell in row 2 receiving an input from its left neighbor will again evaluate its local transition function. In this case, the delay has already expired and there is no *future value* scheduled, so the result of this evaluation will be scheduled as the *future value* for time t = 10 ms.

Figure 9 shows the radial diffusion rules. For a cell with valid top and bottom neighbors, the diffusion rule states that the new cell value will be the average of the three cells. This is the case of the rule in line 22. A delay of 1 ms was chosen. The remaining three rules in lines 23 and 24 cover the special case of top and border cells. These cells do not have both, a valid top and bottom neighbor so instead of using three cells to obtain the average, only two are used.

```
21 [diffusion]
22 rule : { ((-1,0) + (0,0) + (1,0)) / 3 } 1 {
cellPos(0) != 0 AND cellPos(0) != 24 }
23 rule : { ((-1,0) + (0,0)) / 2 } 1 { cellPos(0)
!= 0 AND cellPos(0) = 24 }
24 rule : { ((0,0) +  (1,0)) / 2 } 1 { cellPos(0)
= 0 AND cellPos(0) != 24 }
```

Figure 9. Radial diffusion rules.

So far we have shown the diffusion rule, but we have not yet defined that this ruled should be evaluated when an external event is received through the *diffuse* input port. Figure 10 shows the statements that link the *fia* model *diffuse* input port to a cell's *diffuse* input port (line 27) and set the diffusion rule to be evaluated upon the arrival of an external event through this port (line 28).

```
[fia]
27 link : diffuse diffuse@fia(x,y)
28 PortInTransition : diffuse@fia(x,y) diffusion
```

Figure 10. External coupling of the FIA Cell-DEVS model.

4. Simulation Results

The described model was run for 10s and the state of the whole cell space was logged every 100ms. Figure 11 shows a graphical representation of five different stages the FIA model. Only half of the tube is modeled because we assumed the other half will be symmetrical. In the figure, the upper cells represent the center of the tube, and the lower cells represent the part of the solution touching the walls of the tube. The experiment starts at time 0, where the sample (white), is injected. At this moment, half of the tube contains the carrier solution (dark gray). In the following figures, the convective transport makes the sample disperse faster at the middle of the tube than near the walls. The experiment finishes when the whole tube contains the carrier solution only.

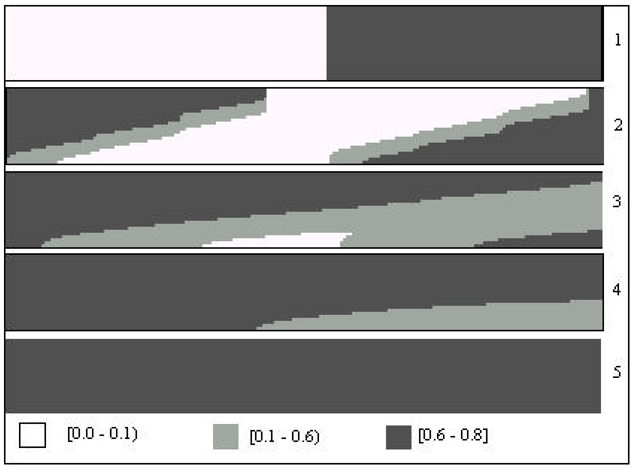

Figure 11. Different execution stages of the FIA model.

The logged results were also used to draw the conductivity curve. To obtain the conductivity of the whole system, we divided the cell space in axial segments, calculated the resistance of each, and assumed the whole resistance to be the result of combining all segments in serial mode. We took each segment to be a column of cells and calculated its resistance using equation 4.

$$R_{total} = \sum_{column=0}^{199} \left(\sum_{row=0}^{24} \frac{1}{R_{cell(row,col)}} \right)^{-1} \qquad \textbf{(Equation 4)}$$

To calculate the resistance, equation 5, which gives the conductivity of each cell, was used. The resistance of a cell can be obtained by calculating the inverse of the conductivity. All values are known, being the concentration of nitric acid the one that varies from cell to cell.

$$G_{cell} = \frac{1}{R_{cell}} = G_{HNO_3} + G_{H_2O} = \frac{Area_{cell}}{Length_{cell}} \left(k_{HNO_3} \cdot [HNO_3] \right)$$

(Equation 5)

Figure 12 shows the conductivity curve obtained. For this example the curve is quite similar to the first part of the measured curve.

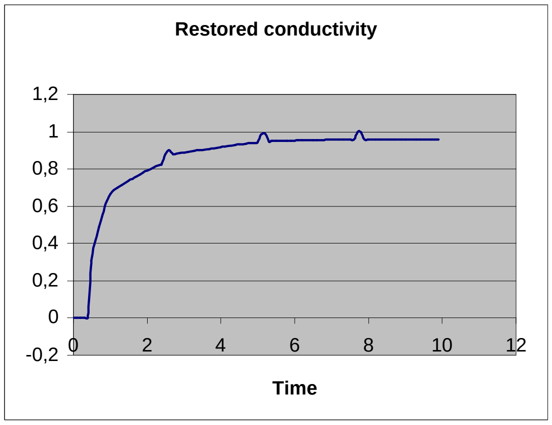

Figure 12. Conductivity curve obtained

The results of this experiments are a good starting point to simulate the whole FIA manifold. The execution results of the diffusion activities were smoother than the results obtained using Random Walks. Cell-DEVS also enables easy definition of diffusion speed in different areas of the loop using different delay functions. The extension of this model to a three dimensional version is straightforward, and would required extensive work with traditional approaches. Using Cell-DEVS the physical phenomenon can be described in a more intuitive way, improving the development times. The discrete-event nature of the formalism can also improve execution times of the simulations.

5. Conclusion

Cell–DEVS allows describing physical systems using an n-dimensional cell-based formalism. Input/output port definitions allow defining multiple interconnection between Cell-DEVS or DEVS models. Complex timing behavior for the cells in the space can be defined using very simple constructions. The CD++ tool, based on the formalism entitles the definition of complex cell-shaped models.

We introduced a Cell-DEVS model to simulate the mass distribution profile of a flow injection system. Experimental results using integrated conductimetric detection have provided valuable information to understand the system’s behavior. But to provide a greater insight of this behavior, a small Cell-DEVS model was developed and its conductivity curve was calculated. The initial curve is similar to the experimental one, it proving the feasibility of the approach.

To completely validate the model, a complete FIA manifold should be simulated. For a system with a loop of 30 cm and a reactor of 100 cm a 1300 by 25 cell space (a total of 32500 cells) would be needed. This number is quite big to be handled by a standalone PC workstation. We are now in the process of running the model using our parallel version of CD++.

6. References

[1] L. Lapidus and G. F. Pinder, Numerical Solution of Partial Differential Equations in Science and Engineering, Wiley, New York, 1982.

[2] G. Weisbuch, Complex Systems Dynamics, Addison-Wesley, 1991.

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

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

[5] S. Wolfram. “Theory and applications of cellular automata”. Vol. 1, *Advances Series on Complex Systems*. World Scientific, Singapore, 1986.

[6] B. Zeigler, H. Praehofer, T. Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2000. Academic Press.

[7] G. Wainer; N. Giambiasi,. "Timed Cell-DEVS: modelling and simulation of cell spaces". In *Discrete Event Modeling & Simulation: Enabling Future Technologies*. 2000. Springer-Verlag

[8] WAINER, G. “Improved cellular models with parallel Cell-DEVS”. In *Transactions of the SCS.* June 2000.

[9] D. Rodríguez, G. Wainer. "New Extensions to the CD++ tool". In *Proceedings of SCS Summer Multiconference on Computer Simulation*. 1-7. 1999.

[10] J. Ameghino, G. Wainer. “Application of the Cell-DEVS formalism using N-CD++.” In *Proceedings of the 2000 Summer Computer Simulation Conference.* July 2000.

[11] J. Ameghino, A. Troccoli, G. Wainer. "Modelling and simulation of complex physical systems using Cell-DEVS". In *Proceedings of the 33rd SCS Summer Multiconference on Computer Simulation*. Seattle, WA. USA. 2001.

[12] F.J. Andrade, F.A. Iñón, M. B. Tudino, O. E. Troccoli “Integrated conductimetric detection: mass distribution in a dynamic sample zone inside a flow injection manifold”, *Analytica Chimica Acta 19211, 1998.*

ALEJANDRO TROCCOLI has received his M. Sc. in Computer Sciences (2001) from the Universidad de Buenos Aires, Argentina. He is currently a first year Ph.D. student at Columbia University, New York, NY, U.S.A.

JAVIER AMEGHINO has received his M. Sc. in Computer Sciences (2000) from the Universidad de Buenos Aires, Argentina. At present he works at COMPAQ Latin America Corporation (Buenos Aires), and he is an Adjunct Lecturer at the Computer Science Dept. of the Universidad de Buenos Aires. He is the coordinator of the PARDEVS lab in the same Dept..

FERNANDO IÑON Fernando Iñón has received a Licentiate in Chemistry from the Universidad de Buenos Aires, Argentina (1995), a M.Sc. in Environmental Analysis at Imperial College, U.K. (1998), and a Ph.D. in Chemistry from the Universidad de Buenos Aires, Argentina (2001). He is currently a teaching assistant at the Universidad de Buenos Aires, and part of the research staff in the Trace Analysis lab, where he is conducting further research on Flow Injection Systems.

GABRIEL WAINER received the M.Sc. (1993) and the Ph.D. degrees (1998, with highest honours) at the Universidad de Buenos Aires, Argentina, and DIAM/IUSPIM, Université d'Aix-Marseille III, France. He is Assistant Professor at the Systems and Computer Engineering, Carleton University (Ottawa, Canada). He was Assistant Professor at the Computer Sciences Dept. of the Universidad de Buenos Aires, Argentina, and a visiting research scholar at the Arizona Center of Integrated Modelling and Simulation (ACIMS, University of Arizona). He has published more than 50 articles in the field of operating systems, real-time systems and Discrete-Event simulation. He is author of a book on real-time systems and another on Discrete-Event simulation. He has been the PI of several research projects, and participated in different international research programs. Prof. Wainer is a member of the Board of Directors of The Society for Computer Simulation International (SCS). He is the coordinator of a group on DEVS standardization. He is Associate Editor of the Transactions of the SCS. He is also a Co-associate Director of the Ottawa Center of The McLeod Institute of Simulation Sciences.