METHODOLOGIES FOR DISCRETE EVENT MODELLING AND SIMULATION

SYSC 5104

Project Report:

Improving the Corsican Fire Spreading Cell-DEVS Model

Matthew MacLeod (100265888)
Rachid Chreyh (100350664)

12/5/2005

1.0 Introduction

The complexity of studying the spread of fire has made it the target of several studies in the modelling and simulation field. Mathematical models for this phenomenon are too complex to allow an analytical solution; therefore researched have turned to simulation to provide some success in predicting fire behaviour.

This paper is intended to enhance an existing fire spreading model: the Corsican Fire Spreading Cell-DEVS model. In this model the physical area of interest is divided into cells, with each cell exhibiting the same behaviour. The model uses a simple set of equations to determine the temperature of each cell at regular time intervals. The temperature of a non-burning cell is an averaging function of its own temperature and that of its neighbours. Once ignited the temperature of burning cells, on the other hand, is also a function of time – the cell's temperature increases to a peak and then falls back down, modelling the exhaustion of fuel in the cell.

The biggest problem with the existing Corsican Fire Spreading Cell-DEVS model is that it takes a long time to execute. This is mainly due to the large number of messages exchanged amongst all the cells in the simulation every single time step. Our contribution is to explore modifications to the existing model to speed up the simulation. The problem was attacked in two major ways, using 'dead reckoning' to vary the length of the time steps taken by each cell and using Quantized DEVS (Q-DEVS) to quantize the cells' output. Using Q-DEVS generally reduces the number of messages exchanged between the cells, as messages are only sent when the output of a cell passes a quantization threshold. In the other case, we are stepping away from the traditional way of looking at the fire equations – deriving an equation that gives you the temperature at each given time step. Instead, we try to find to do the opposite, and find an equation (or set of equations) that determine the time the next quanta will be passed as a function of the current temperature. In other words we will only change the temperature of the cells at the quantum boundaries, by using an equation that determines at what time the next quantum will be reached. This paper explores how using one or both of the methods together can sometimes decrease the number of messages sent (and hence the execution time), but with definite issues with the resulting accuracy. Also, the likely reasons for much unexpected poor behaviour of both methods will discussed, and possible solutions proposed.

2.0 Background

The spread of fire is a complex phenomenon that many have tried to study over the years. As one can imagine, the spread of fire depends on many different variables such as the material being burned, the geography of the area, and the weather. It has been determined that finding an analytical solution for mathematical models of fire spread is almost impossible, and therefore many have looked to simulation as an attractive alternative. Simulations have been found that represent the way in which fire spreads reasonably accurately, and are now generally the preferred solution for predicting the behaviour of fire. This goal of predicting fire behaviour is important to firefighters, for example, because having a tool that is able to predict where the fire will be and how it will move will enable them to better plan strategies to control the fire quickly and safely. An aspect of great importance in such a tool is that it has to be able to predict the fire behaviour (at the very minimum) faster than the fire itself moves, preferably much faster. In a real life situation, if we want to use a tool to help us predict how the fire will spread we have to be confident that it will give us a reasonable result on the order of minutes. Otherwise valuable time will be lost as the fire spreads further, regardless of the accuracy of the results.

The authors of [1] have shown how Cell-DEVS and CD++ can be used to model fire spread. In this project we have started with the Cell-DEVS Corsican Fire Spread model and have modified it to achieve a faster simulation time (in some cases), with a somewhat large loss of accuracy. First let us quickly go over how the original Corsican Cell-DEVS model worked. Below is a simplified diagram of the temperature curve used in the Corsican model. The temperature curve is divided into four stages, and at any given instant each cell in the model will be in one of these stages. The first is the inactive stage in when a cell has no neighbours with a temperature higher than the ambient temperature (T_a). The second is the unburned stage in which the temperature of the cell is increasing due to heat from the neighbouring cells; during this stage the cell's temperature is between the ambient temperature and the ignition temperature (300 ºC). The third is the burning stage in which the cell has reached the ignition temperature and fuel in the cell starts to burn. During this stage the cell's temperature increases until it reaches a peak

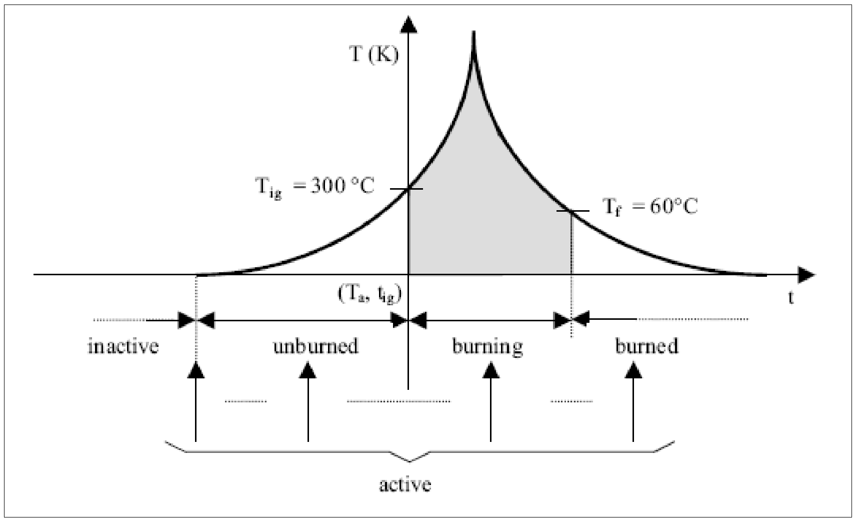


Figure 1: Simplified Temperature curve [1]

temperature, and then falls back down to 60 ºC. It has now entered the fourth and final stage, burned. Because it has exhausted its fuel, it can no longer reignite and is considered inactive.

The Cell-DEVS implementation contains two planes to store the state for each cell. The first represents the fire spread itself, in which each cell calculates its temperature. The additional plane is used to store the ignition times for the cells (this can be considered merely a state variable of the true cells of interest). The cells in the ignition temperature plane have a simple rule – record the current simulation time when the corresponding cell in the fire spread plane reaches the ignition temperature. As can be expected, the cells in the fire spread plane have more complex rules. When a cell is in the unburned phase its temperature is calculated as the weighted average of the current cell’s temperature with its neighbours' temperatures. When a cell is in the burning phase its temperature is again calculated as the weighted average of the temperatures around it, but with the addition of the result of a decaying exponential function of time describing the temperature behaviour. In the other two phases (inactive and burned) the cell’s temperature does not change and thus these cells should remain in the passive state (the inactive cells will of course respond to any temperature changes in their neighbourhood, so they may eventually ignite). One of the advantages of using Cell-DEVS is that if the rules are written properly all cells in the inactive or burned phase will remain passive and thus the calculations will be confined to the fire front, saving on execution time. Burned cells do this effectively in this model, but unburned cells do not. The Corsican Cell-DEVS model itself is explaneed more thoroughly in [1].

3.0 Discussion of the Corsican Cell-DEVS Model

Running the simulation of the existing Corsican Cell-DEVS led to several important observations. First and the most important deficiency of the current simulation is that it takes far too long to run. In the original examples we ran it took around ten minutes to run 15 seconds of simulation time. The log files produced by the simulation were also very large (around 300 Megabytes (MB) for the 15 seconds of simulation time). For the purposes of easy comparison with different techniques we reduced the size of the cell space from 60x60 to 10x10 for all experiments described below. Secondly, it was observed that during the burning phase the temperature curves of all the cells are very similar. This was taken to imply that the effect of the neighbouring cells can be ignored when calculating the temperature of a cell during the burning phase. Whether or not this was a valid assumption will be explored below.

The reason for the slow execution time of the simulation is mainly due to the high number of messages being exchanged between cells. In the current model each cell in the unburned or burning phase will update its temperature once every 1 ms and will as a result send messages to its neighbours. To remedy this problem a solution must be found that decreases the number of messages exchanged. The solution we propose is described in the following section.

4.0 Proposed Fire Spread Cell-DEVS Model Definition

4.1 **The Conceptual Model**

As mentioned above, the problem we are trying to address is the speed of the simulation. To increase the speed we propose two main simplifications. Firstly, if we are able to keep the unburned cells completely in the passive state until they reach the ignition temperature, we would reduce the number of cells that send out messages to their neighbours. The second is to use quantization to reduce the number of messages exchanged among the cells. This proposal is explaneed in more detail below.

The model we constructed still has the same four phases of inactive, unburned, burning and burned. One main difference however is that in our model, in addition to the burned phase, the unburned phase cells also remain passive. We have found that a cell in the unburned phase will start to climb the temperature curve above the ambient temperature when: a) one of its neighbours has reached a temperature above 650 °K , or b) two of its neighbours have reached a temperature above 474 °K. As a result we are able to keep all cells in the passive state until they reach the ignition temperature as indicated above. From the provided data from the original model we found that all cells more or less exhibit the same temperature curve when they are in the burning phase, implying that the temperatures of neighbouring cells do not have a big impact on a

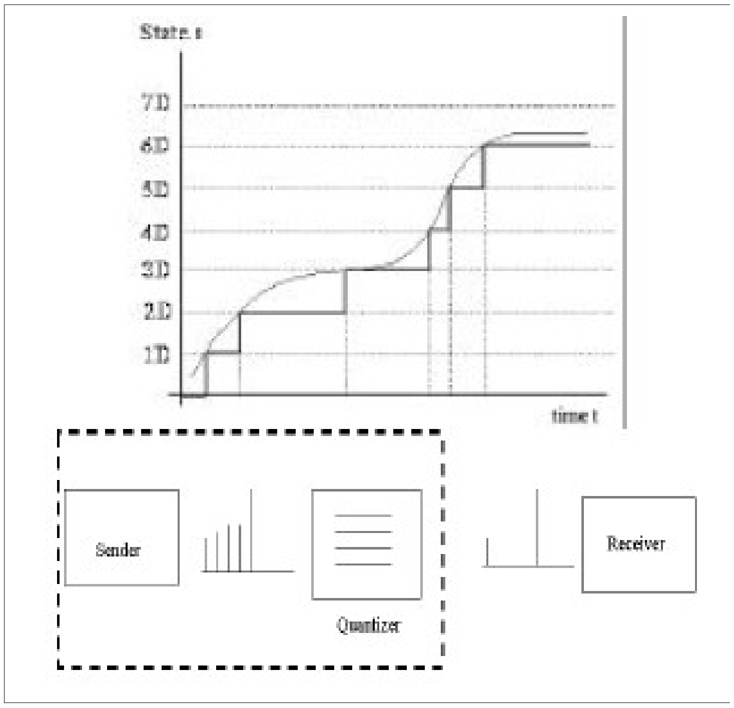


Figure 2: Q-DEVS Quantization [3]

burning cell's temperature. As a result, when a cell reaches the ignition temperature it can calculate its temperature by merely following the temperature curve determined from experimentation. By doing this we have restricted the majority of calculations to the cells in the burning phase, and removed the need for messages from the neighbours in many cases.

The first idea for reducing the messaging between cells dramatically is quantization [2]. There are two quantization ideas that we have implemented in our model.s The first is to use Q-DEVS. In Q-DEVS all cells in the model have a fixed quantum size and each cell has a quantizer (see Figure 2 above [3]). The idea is that each cell will only send output to its neighbours if its temperature has passed into the next quantum threshold. The quantizer acts as the detector that decides when a threshold has been crossed, and sends out the output only if the threshold has been crossed. By implementing quantization as described here the number of messages exchanged between cells will be reduced thus increasing the speed of the simulation, but also the accuracy of the simulation will be reduced. The key is to select a quantum size that strikes a good balance between speed increase and accuracy reduction.

The second quantization idea involves looking at the data from another perspective. As mentioned above, the Corsican model calculates the temperature curve as a function of time. Using this function we input a value for time and it returns the temperature of the burning cell. However, it was proposed to look at the inverse of the curve – i.e. as a function that gives us the time to reach a specific temperature. If we have such a function, we can specify a quantum value (for the purposes of this paper we will call it the *hard coded* quantum) for temperature and use the function to calculate what the time will be when the cell's temperature crosses the next quantum. If we do this the cells will be "asleep" until they reach the next quantum threshold where they will wake up, calculate the next time at which they will cross the quantum threshold and then go back

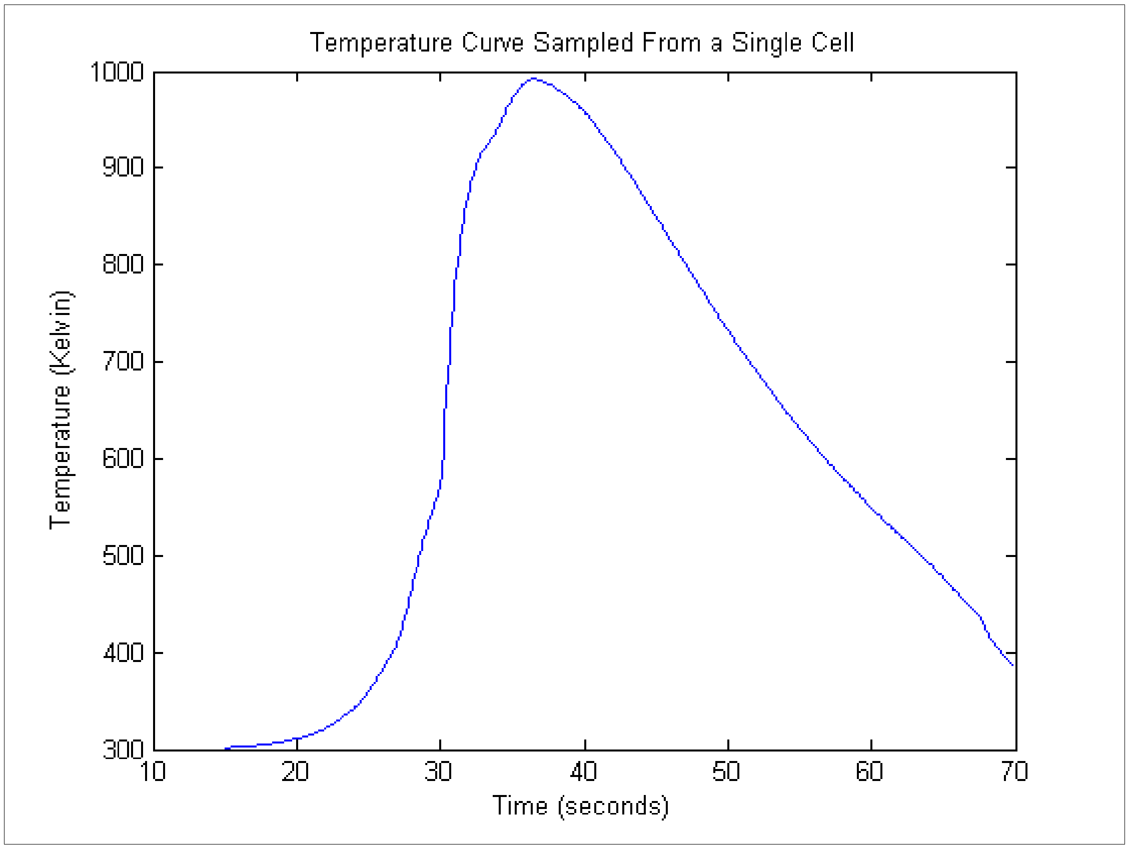


Figure 3: Temperature Curve of a Single Cell in the Corsican Model

to sleep until that time has been reached. This will save a lot of execution time since the cells will only be waking up on the significant event of crossing a threshold of interest.

To get the required function to do our simulation we started with a normal temperature curve of a cell in the burning phase (see Figure 3 above). As can easily be seen this function fails the horizontal line test [4], and therefore is not directly invertible. First we must divide it into increasing and decreasing components, giving us two individually invertible functions (See Figures 4 and 5). A state variable can then be used to choose between them during execution. Using Matlab we fit functions that approximate the two data cruves. Note that for the scope of our project we are not overly concerned with how accurate the functions we have are in emulating the real curve. The focus of this study was to analyze the performance of our proposed model, which if successful could be refined by fire experts to the desired level of accuracy. Collecting data for this model would also be more efficient, as instead of sampling every 'cell' of the real model every 1 ms, samples would only have to be recorded at threshold crossings. This would potentially save much data storage and make better use of network bandwidth in the test bed.

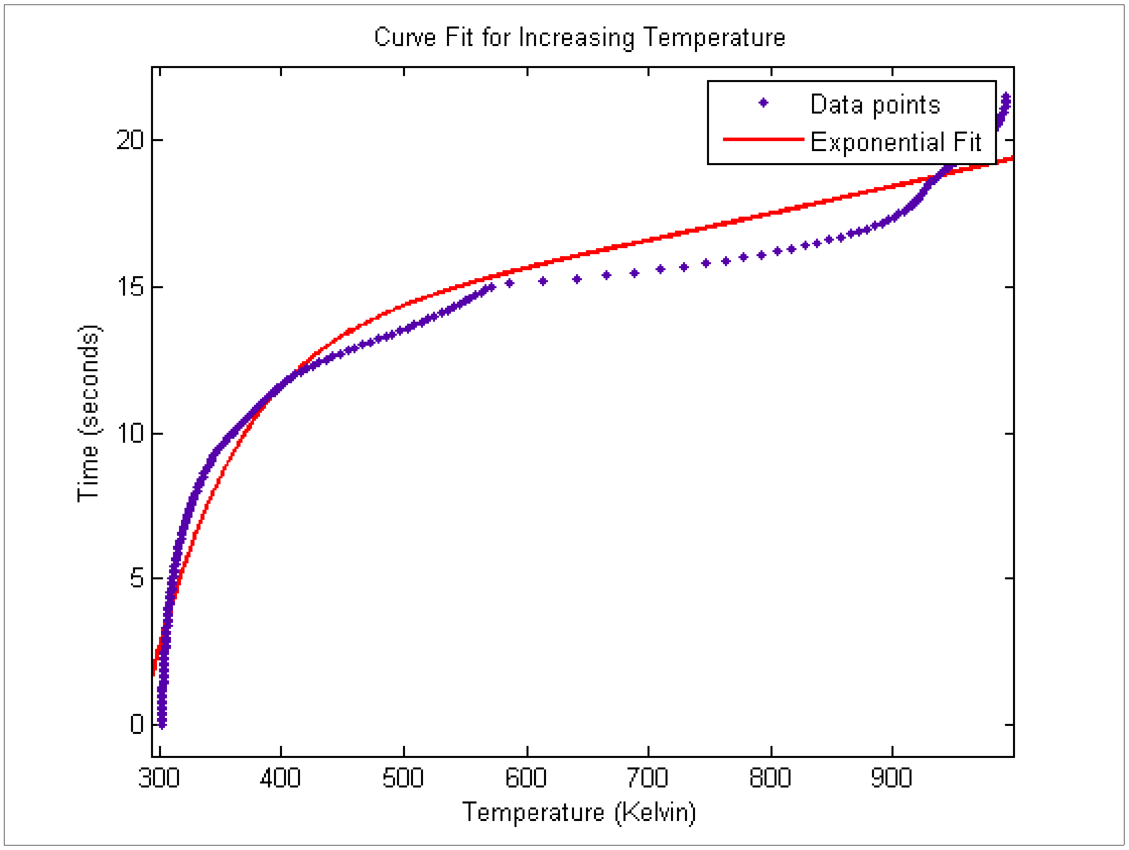


Figure 4: Inverted Increasing Temperature Data and Fit

The fit for the increasing temperature portion of the curve is shown above in figure 4. It uses a sum of two exponential functions:

$$f(t) = 11.56 * e^{0.0005187 * temp} - 784.7 * e^{(-0.01423 * temp)}$$

Where *temp* is the temperature and *t* is the time. Similarly, the decreasing portion (figure 5) is fit with the linear function:

$$f(t) = 0.052 * temp$$

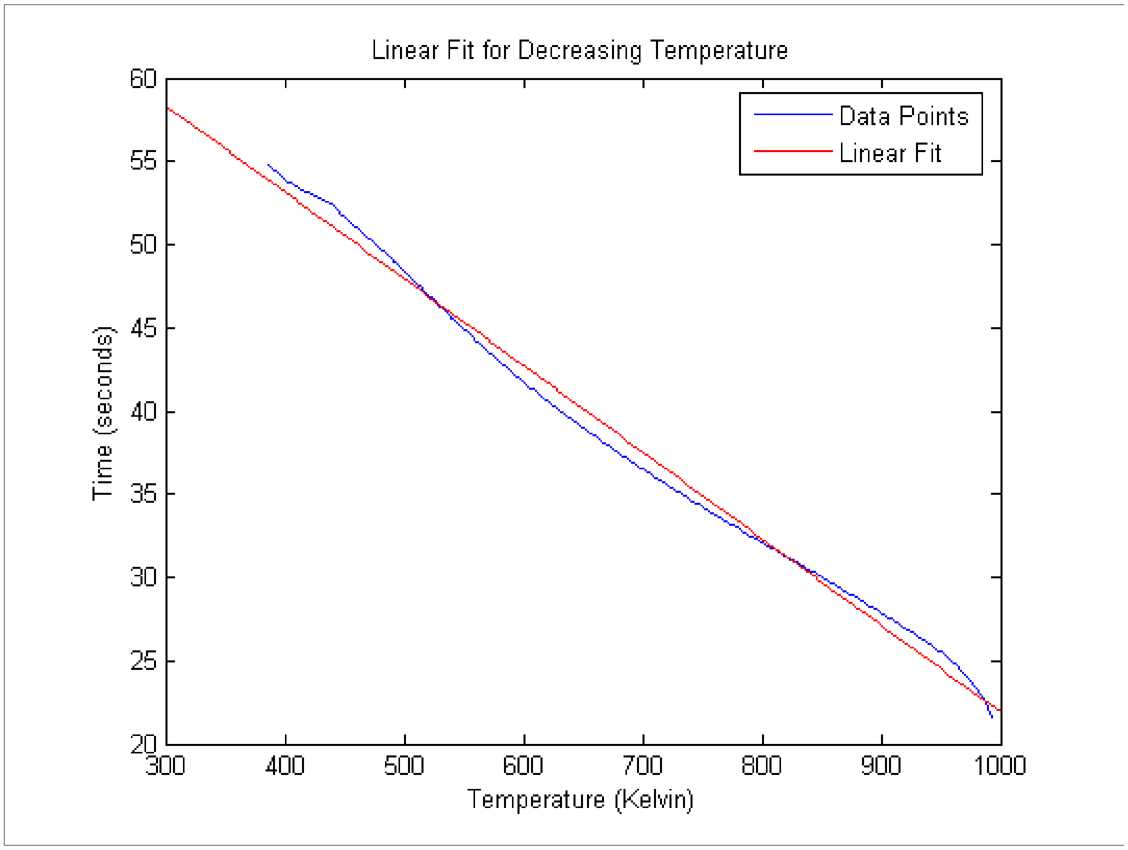


Figure 5: Inverted decreasing temperature function

There are a few things to notice about these curves. Originally higher order functions (quadratic, etc.) were used to fit the functions, and did so with greater accuracy. After some thought we realized these functions would not work – as time is of necessity strictly increasing, and the fit functions had several local minima and maxima (implying that they decreased in some regions). As we cannot advance time in a negative direction, any function of time must also pass the horizontal line test, and therefore be invertible. As this inverted function will also necessarily be invertible, this implies that we cannot model any up and down fluctuations in temperature within any of our piecewise curves. A new state for any change in direction of the temperature. This restriction causes the obtained functions to be linear or near-linear in most regions (as shown in Figure 4 the exponential curve shown has two nearly linear regions joined by a knee). These functions will be used to develop the time advance portion of the model rules.

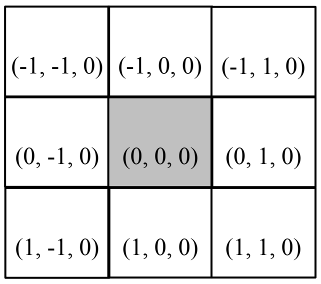
4.2 The Formal Model Specification

In our model we have added one additional plane to the existing model. Cell temperatures remain on the first plane, and ignition times are moved to the third. In the second plane we store information about each cell that will help us determine which rule to apply. The second plane has a value of 0 by default, and the following values as indicated below:

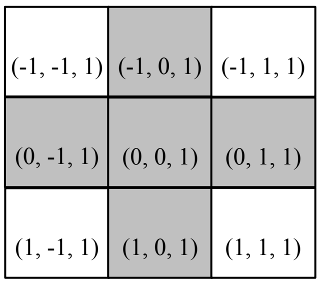
- -100 : If the temperature of the cell is between 301 and 474, meaning it is burning but not hot enough to cause a neighbour to ignite.
- -200: If the temperature of the cell is between 474 and 650, meaning it is burning and hot enough to cause a neighbour to ignite if another neighbour is also in this state.
- -300: If the temperature of the cell is above 650, meaning it is burning and is by itself hot enough to cause a neighbour to ignite.

- -400: If the temperature has reached the peak temperature (992) from our data curve and is now starting to burn with a decreasing temperature.
- -500: When the cell has burned out.

The neighbourhood for the model is as follows: looking at a cell in the first plane (the fire spread plane) each cell has its corresponding cell in the second plane (the supporting info plane) and the Von Neumann neighbourhood of that cell as its neighbours. The neighbourhood therefore looks as depicted in the flowing diagram:



Plane 1 (Fire Spread plane)



Plane 2 (Supporting Info plane)

The formal specification <X, Y, I, S, θ, N, d, τ, δint, δext, λ, ta> for the atomic Cell-DEVS model is defined as follows:

$X = \{ x \mid x \in [R^+, -100, -200, -300, -400, -500] \}$

$Y = \{ y \mid y \in [R^+, -100, -200, -300, -400, -500] \}$

$I = <6, 0, \{P^{x1}, P^{x2}, .. , P^{x6}\}, \{\}>$

$S = \{ s \mid s \in [R^+, -100, -200, -300, -400, -500] \}$ // where:

R^+ = Real number represents Cell Temperature,

-100 = Cell in other plane between 301 and 474 °K,

-200 = Cell in other plane between 474 and 650 °K,

-300 = Cell in other plane above 650 °K,

-400 = Cell in other plane reached peak temp,

-500 = Cell has burned out

$\theta = \{ (s, phase, f, \sigma)\}$

where:

- $s \in [R^+, -100, -200, -300, -400, -500]$,
- $phase \in \{passive, active\}$,
- $f \in T$
- and $\sigma \in R_0^+$ //use inertial delay

$N \in S^6$

d = 1 ms (or determined by the inverse temperature functions introduced in 4.1)

δint: internal transition function which is defined by CD++ automatically

δext: external transition function which is defined by CD++ automatically

λ: output function which is defined by CD++ automatically

ta(passive) = INFINITY

ta(active) = d

τ = The local computing function described below: (See .MA file for details)

A cell whose corresponding neighbor in above plane has values of -100, -200 or -300:

These Cells are in the Burning Up phase, meaning that they are burning and have not yet reached their peak temperature. These cells will calculate (according to the burning up function) the time delay after which they should increment their temperature by the quantum amount and then sleep for this time

A cell whose corresponding neighbor in other plane has values of - 400:

These Cells are in the Burning Down phase, meaning that they are still burning but have reached their peak temperature and their temperature is falling from here on in. These cells will calculate (according to the burning down function) the time delay after which they should decrement their temperature and then sleep for this time

A cell whose value is 0 and its corresponding neighbor in other plane has a value between 301 and 474:

These cells are in the "Supporting Info" plane. After a short time delay they are to get a value of -100 indicating that their corresponding cell has ignited but is still below 474 °K.

A cell whose value is 0 or -100, and its corresponding neighbor in other plane has a value > 474:

These cells are in the "Supporting Info" plane. After a short time delay they are to get a value of -200 indicating that their corresponding cell has ignited and has reached 474 °K. Two of these cells can cause a neighbor to ignite.

A cell whose value is 0 or -200, and its corresponding neighbor in other plane has a value > 650:

These cells are in the "Supporting Info" plane. After a short time delay they are to get a value of -300 indicating that their corresponding cell has ignited and has reached 650 °K. This cell alone can cause a neighbor to ignite.

A cell whose value is -300 and its corresponding neighbor in other plane has a value > 992:

These cells are in the "Supporting Info" plane. After a short time delay they are to get a value of -400 indicating that their corresponding cell has just reached the peak temperature and should use the burning down equation.

A cell whose value is -400 and its corresponding neighbor in other plane has a value < 332:

These cells are in the "Supporting Info" plane. After a short time delay they are to get a value of -500 indicating that their corresponding cell has Burned out.

Ignition Rules:

A cell in the "fire spread plane" that has not ignited yet (i.e. has a value of 300 °K) will ignite if at least two of its neighbors have a value of -200 or at least one neighbor with a value of -300. The cell will ignite by being assigned a temperate of 301 °K.

Border Rules:

Because the borders are not wrapped, the borders have special rules that force their values to always be constant.

The formal specification <Xlist, Ylist, I, X, Y, η, N {f, c}, C, B, Z, select> for the coupled Cell-DEVS model is defined as follows:

Xlist = {Φ}

Ylist = {Φ}

I = {Φ}

X = {Φ}

Y = {Φ} //no external inputs and outputs

η = 6

N = { (-1,0,1) , (0,-1,1) , (1,0,1) , (0,1,1) , (0,0,0) , (0,0,-1), (0,0,1) }
//neighborhood

{f,c} = {10,10} //10x10 cell space

C = { C_{ij} | i∈[2,9], j∈[2,9]}
Where C_{ij} is an atomic component defined in the previous part.

B = {C_{1j}, C_{10j}, C_{i1}, C_{i10}} //the border is Not wrapped

Where: C_{1j}, C_{10j} , C_{i1} , C_{i10} are as defined for the Border Rules in the previous part.

Select = {(-1,0,1) , (0,-1,1) , (1,0,1) , (0,1,1) , (0,0,0) , (0,0,-1), (0,0,1) }

5.0 Simulation Results and Comparisons

We ran the fire spread simulation in CD++ using our proposed model and we also ran the original Corsican model (FireCorse.ma) in order to compare the results of the two. A smaller cell space was used in order to reduce the simulation times to manageable levels (as noted in [5] there is a very long initialization time for large models). The initial values used were similar to that in the provided model, representing a line ignition scenario.

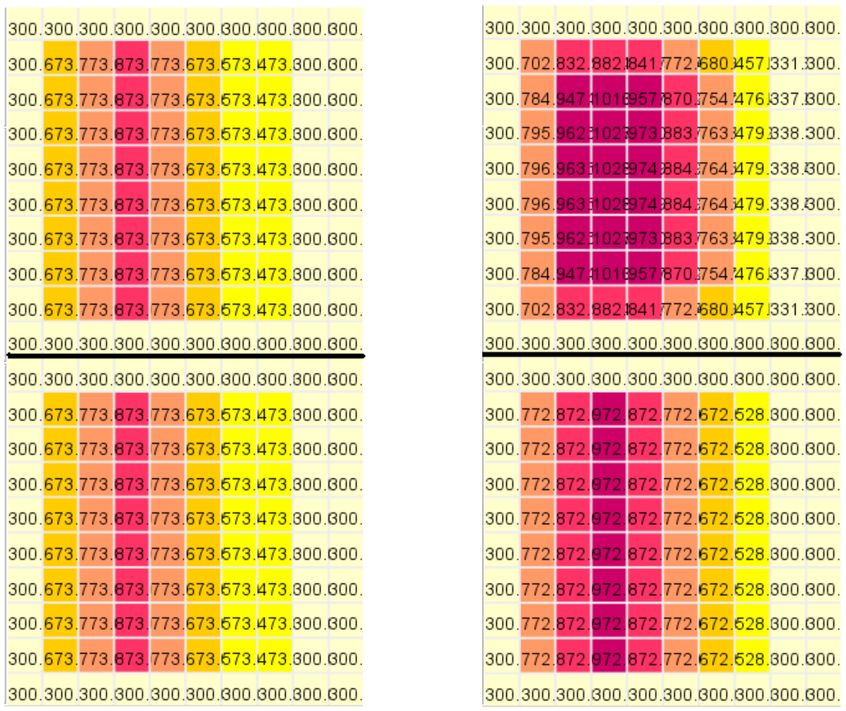
As we had noted earlier the aim of our model is to reduce the execution time of the simulation without reducing the accuracy by too much. As the running time of the model tends to be affected greatly by initialization delays as mentioned above (and is hard to measure directly on Windows) the number of messages are used as our performance metric (as measured by the size of the logs). The size of the log file was reduced by more than 50%, from 42.1 MB to 19.6 MB even when almost all cells in the model are active. Gains were even greater when only a few cells were initially activated.

The following series of diagrams depicts the results we obtained using our model and using the Corsican model. One of the first things we see when we examine the results is that the temperatures go much higher ('white hot') in the Corsican model than possible in the proposed model. This makes the proposed model not as accurate because the temperatures in the Corsican model take much longer to fall. The proposed explanation for this is the way the model deals with border cells. Because the fire is in a confined space and most of the cells started out burning, there are fewer unburned cells to average out the temperature and cool the fire. This is analogous to the diffusion of heat in real scenarios.

A further issue with the borders is evident by looking at the shape of the spread. Because of the averaging effect, cells in the centre area are higher than on the edges in the original model, as they are surrounded by very hot cells. Our curve was sampled from the original larger line ignition model near the middle, which behaved much more evenly than the distorted edges. This effect can be seen in the orderly advance of our fire front.

One final thing of note in the comparison of the models is that at some points during the simulation some cells close to the front are more than 200 degrees off in our model for the original model, but the temperature of these cells comes back closer in line with the original model's results. Some of this is due to the method of selecting initial values. These were copied from the original model, and descend by 100 degrees in each column, a perhaps somewhat arbitrary choice that is quickly smoothed out to a more realistic curve by the averaging functions used there. As our model is following a fixed curve, small inconsistencies in the initial state (especially when the temperature is rapidly increasing or decreasing) are magnified.

In general the proposed model has performed with a much better speed and producing much smaller logs. Considering the issues mentioned above the accuracy is not horrible, but definitely needs some fine tuning. However, it is probably not realistic to expect the modeller to know beforehand what sort of shape the average cell is going to take, and it is obviously affected by the initial conditions. For instance, if only a few cells have a starting temperature just above 301 Kelvin, our model will quickly spread fire everywhere, whereas the original model will not even reach a red hot state. Requiring knowledge of the shape of the curve beforehand does not seem like a practical solution.



Corsican Model

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	702.	832.	882.	841.	772.	680.	457.	331.	300.
300.	784.	947.	1016	957.	870.	754.	476.	337.	300.
300.	795.	962.	1027	973.	883.	763.	479.	338.	300.
300.	796.	963.	1028	974.	884.	764.	479.	338.	300.
300.	796.	963.	1028	974.	884.	764.	479.	338.	300.
300.	795.	962.	1027	973.	883.	763.	479.	338.	300.
300.	784.	947.	1016	957.	870.	754.	476.	337.	300.
300.	702.	832.	882.	841.	772.	680.	457.	331.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

Proposed Model

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	673.	773.	873.	773.	673.	673.	473.	300.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

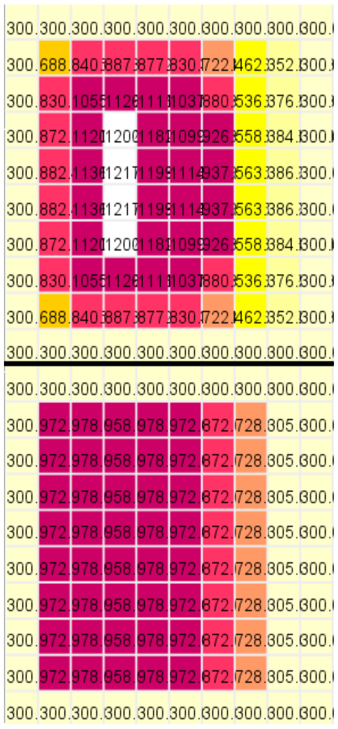
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	772.	872.	972.	872.	772.	672.	528.	300.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

At time = 0

At time = 100 ms

Corsican Model

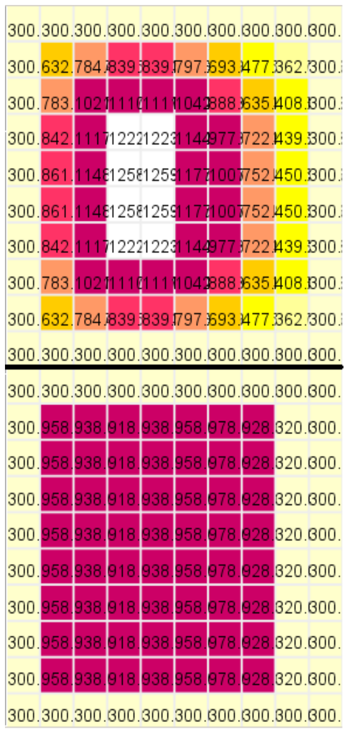
Proposed Model



300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	688.	840.	887.	877.	830.	722.	462.	352.	300.
300.	830.	1055	1128	1113	1037	880.	536.	376.	300.
300.	872.	1120	1200	1182	1099	926.	558.	384.	300.
300.	882.	1136	1217	1199	1114	937.	563.	386.	300.
300.	882.	1136	1217	1199	1114	937.	563.	386.	300.
300.	872.	1120	1200	1182	1099	926.	558.	384.	300.
300.	830.	1055	1128	1113	1037	880.	536.	376.	300.
300.	688.	840.	887.	877.	830.	722.	462.	352.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	972.	978.	958.	978.	972.	872.	728.	305.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

At time = 300 ms



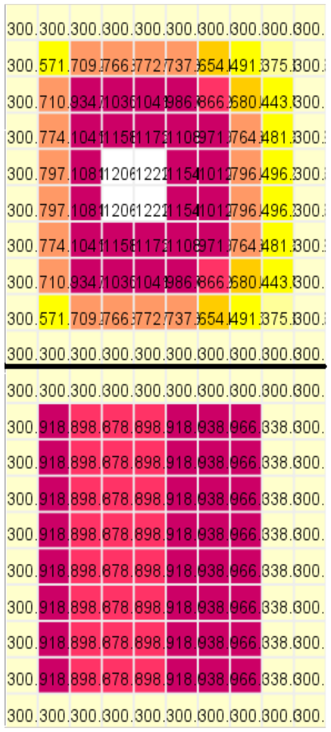
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	632.	784.	839.	839.	797.	693.	477.	362.	300.
300.	783.	1021	1110	1111	1042	888.	635.	408.	300.
300.	842.	1117	1222	1223	1144	977.	722.	439.	300.
300.	861.	1148	1258	1259	1177	1007	752.	450.	300.
300.	861.	1148	1258	1259	1177	1007	752.	450.	300.
300.	842.	1117	1222	1223	1144	977.	722.	439.	300.
300.	783.	1021	1110	1111	1042	888.	635.	408.	300.
300.	632.	784.	839.	839.	797.	693.	477.	362.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	958.	938.	918.	938.	958.	978.	928.	320.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

At time = 500 ms

Corsican Model

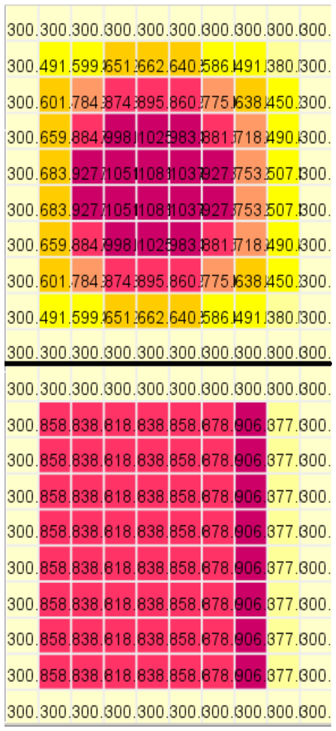
Proposed Model



300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	571.	709.	766.	772.	737.	654.	491.	375.	300.
300.	710.	934.	1036	1041	986.	866.	680.	443.	300.
300.	774.	1041	1158	1172	1108	971.	764.	481.	300.
300.	797.	1081	1206	1221	1154	1012	796.	496.	300.
300.	797.	1081	1206	1221	1154	1012	796.	496.	300.
300.	774.	1041	1158	1172	1108	971.	764.	481.	300.
300.	710.	934.	1036	1041	986.	866.	680.	443.	300.
300.	571.	709.	766.	772.	737.	654.	491.	375.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	918.	898.	878.	898.	918.	938.	966.	338.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

At time = 700 ms



300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	491.	599.	651.	662.	640.	586.	491.	380.	300.
300.	601.	784.	874.	895.	860.	775.	638.	450.	300.
300.	659.	884.	998.	1025	983.	881.	718.	490.	300.
300.	683.	927.	1051	1081	1037	927.	753.	507.	300.
300.	683.	927.	1051	1081	1037	927.	753.	507.	300.
300.	659.	884.	998.	1025	983.	881.	718.	490.	300.
300.	601.	784.	874.	895.	860.	775.	638.	450.	300.
300.	491.	599.	651.	662.	640.	586.	491.	380.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	858.	838.	818.	838.	858.	878.	906.	377.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

At time = 1000 ms

6.0 Quantizing the Model – Issues and Experiences

Several other important issues related to quantization made themselves clear during the initial model development. On initial launch with a hard coded quanta of 1 ms, many of the temperatures were jumping up immediately to 984 degrees. After some analysis it became clear that up until this region on the increasing temperature curve the calculated time step per degree was less than 1 ms. CD++ truncates these values to 0, so a great many transitions were happening at the initial time of 0:0:0:0 and not being displayed. This not only decreased the accuracy greatly, but actually increased the number of messages, as a large number would occur per millisecond.

This was obviously not desirable, and was somewhat mitigated by rounding the calculated time before passing it to CD++. Looking at the data, however, it becomes clear that temperatures on the decreasing curve tend to drop around 2 degrees per millisecond, and rise even faster than that on the increasing curve. As such the original model was already in a sense provided a relatively large degree of quantization (any digital model must quantize on some level), and when a cell is burning there is a margin of error on each calculation on the order of several degrees.

As such the only cases in which a quantization value less than around 2 can actually decrease the number of calculations/updates is for the initial and final passive states. This decrease will tend to offset the increase in messages (multiple per time step for burning cells) for large, sparse models, but in more active models will decrease performance and accuracy. Based on this observation (and the loss of accuracy already apparently with a small quanta), we speculate that the time step in the original model is actually close to the ideal quanta for the performance/accuracy trade-off. Without access to the real data it is hard to compare the accuracy to an already somewhat inaccurate model.

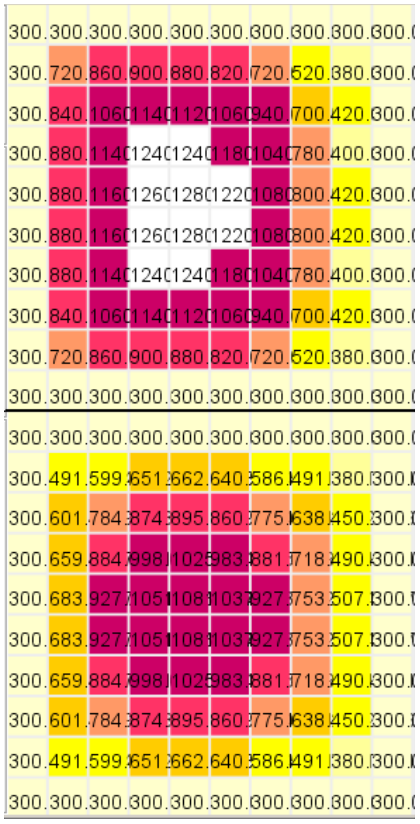
To further back up this assertion, we consider the functions used to approximate the time advance function. Because the decreasing function in particular is modelled as a linear function the time step is in fact fixed (the temperature values cancel out in the time calculation, as seen in the model). Worse than that, it is fixed at a value fractional compared to what CD++ can represent, causing compounding error accumulation (whereas temperatures are represented as real numbers). The exponential function is also usually close to a fixed value (linear) for several updates at a time. So in the end one may as well select a good, fixed time step, for each stage and calculate the temperature value properly.

This actually follows directly from the invertibility constraints referred to above – quantization often shows gains because it smoothes out small, unimportant up and down fluctuations in output, but the constraints of our calculation method preclude any fluctuations. It can also provide gains by skipping large, precalculable changes in a function from one point to another. Although our model does have these regions, there are regions near the bottom and top of the function that require greater precision. Therefore a dynamic quantum size is preferable. This is likely why the other group working on fire spreading saw little or no change to the results when running with a quantum set in Q-Devs. Unfortunately we did not have the opportunity to test with a version of CD++ that supported dynamic quanta.

7.0 Combined Model

To deal with the accuracy issues, a model that combined the aspects of both was constructed. Essentially the time function of our model was plugged into the Corsican model, with the calculating function of the original model maintained. The only modification was to add the hard coded quanta to the calculated temperature at each time step. The idea being to gain the advantage of skipping up the curve quickly, while being able to correct any inaccuracies by re-evaluating the correct temperature after the jump.

Several issues were found with this model (FireTest2.ma). One of these is that cells tend to flip back and forth between quantum values, as they're required to change a fixed amount by the time update function, but may actually want to be somewhere in between. Secondly, the number of messages is actually much greater than the original model, until both the hard coded quanta and that specified to the simulator is raised to 20. At this point, the modified model still generated 45 MB of messages, versus 42 MB from the original. As you can see from figure 6 on the right, although the shape of the front remains pretty much the same, the temperature values of the new model (upper graph) off by around 200 degrees. Watching the simulation showed it tending to lag behind the original values. This model does not seem to be a good trade-off either. The lag may be caused by



300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	720.	860.	900.	880.	820.	720.	520.	380.	300.
300.	840.	1060	1140	1120	1060	940.	700.	420.	300.
300.	880.	1140	1240	1240	1180	1040	780.	400.	300.
300.	880.	1160	1260	1280	1220	1080	800.	420.	300.
300.	880.	1160	1260	1280	1220	1080	800.	420.	300.
300.	880.	1140	1240	1240	1180	1040	780.	400.	300.
300.	840.	1060	1140	1120	1060	940.	700.	420.	300.
300.	720.	860.	900.	880.	820.	720.	520.	380.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

300.	300.	300.	300.	300.	300.	300.	300.	300.	300.
300.	491.	599.	651	662.	640.	586.	491.	380.	300.
300.	601.	784	874	895.	860.	775.	638.	450.	300.
300.	659.	884.	998.	1025	983.	881.	718.	490.	300.
300.	683.	927.	1051	1081	1037	927.	753.	507.	300.
300.	683.	927.	1051	1081	1037	927.	753.	507.	300.
300.	659.	884.	998.	1025	983.	881.	718.	490.	300.
300.	601.	784	874	895.	860.	775.	638.	450.	300.
300.	491.	599.	651	662.	640.	586.	491.	380.	300.
300.	300.	300.	300.	300.	300.	300.	300.	300.	300.

Figure 6: Modified Corsican vs. Original @ 1 ms

being unable to precalculate correctly what your temperature is going to be after the temp step, but only what it should be now.

8.0 Proposed Solutions

Although simply using dynamic Q-Devs may likely be the easiest and best trade-off, we propose another possible solution using the dead reckoning approach. Using the observation that the temperature curve is generally close to linear other than in a few transition ratios, really what one needs is to track the current slope of the function and extrapolate from there to predict the next change. Accurate calculations of the current temperature could be made after either every jump, or perhaps a state variable could be kept to limit it to every *N* jumps. Otherwise it would not be able to detect changes in direction without knowing where they were *a priori*, which as discussed above is an unrealistic expectation.

This sort of technique has already been applied to distributed simulation as shown in [6] and other references. One of the authors remembers coming across it in discussion of the HLA.

9.0 Conclusion

Our study concurred that the original Corsican model was too slow in execution time, due to the large number of messages exchanged among the cells in the model. However, our techniques to improve the performance came at too much of an accuracy penalty. We theorize that the original model actually was quantized about as much as it could be to maintain reasonable accuracy, so there was little room for improvement. However, if a more precise model is desired our techniques may come into play.

Out first proposed model used a form of hard coded quantization to reduce the number of messages between the cells and thus increase the speed of the simulation. Quantization was implemented by calculating the time steps between temperatures, instead of the temperatures at time steps. We found equations describing the time when a cell reaches a specific temperature during the burning phase. These were used equations in our model to find the time it takes for the cell to increase (or decrease) in temperature by a quantum amount, thus achieving the goal of keeping all cells "asleep" until a significant event takes place. The effect of neighbouring cells were ignored, as in previous test runs all cells were seen to develop similarly. Another modification was to keep cells in the unburned state "passive" until they are seen to reach the ignition temperature. The combination increased performance, but had problems with accuracy and requiring some prior knowledge of how the fire would develop to obtain good equations and initial values. We found that the general direction and speed of fire spread was maintained by our model, although some finer details such as peak temperatures and temperatures of cells at the fire front were not very accurate.

To improve the accuracy we attempted to combine our solution more directly with the original model, by using the original equations involving the neighbours. Unfortunately the inaccuracies seemed to cause more fluctuations and therefore updates than they eliminated, and the model performed poorly as a result. A great deal of inaccuracy had to be tolerated to even match the performance of the original model.

We conclude that the highly dynamic nature of a fire remains difficult to model correctly with approximations. This will be even more evident if slope and weather are taken into account. Our recommendation is that methods of dynamic quantization be explored, rather than further simplification of the model, whether using built-in support in future versions of CD++ or rule based dead reckoning approximations. Also, our techniques should be reevaluated if a more precise model is desired, as simpler quantization methods may be more useful in that scenario.

10.0 References

[1] A. Muzy, E. Innocenti, A. Aiello, J. Santucci, and G. Wainer, "Specification of Discrete Event Models for Fire Spreading," *The Society for Modeling and Simulation International*, vol.81, issue 2, February 2005.

[2] G. Wainer and B. Zeigler, "Experimental Results of Timed Cell-DEVS Quantization," in *Proceedings of AIS'2000*. Tucson, Arizona, USA, 2000.

[3] G. Wainer and A. Muzy, "Cell-DEVS Quantization Techniques in a Fire Spreading Application," in *Proceedings of the 2002 Winter Simulation Conference*.

[4] Wikipedia, Wikimedia Foundation, Inc., "Horizontal Line Test," December 2005, http://en.wikipedia.org/wiki/Horizontal_line_test.

[5] A. Muzy, G. Wainer, E. Innocenti, A. Aiello, J.-F. Santucci, "Comparing Simulation Methods for Fire Spreading Across a Fuel Bed," Computer Modeling, University of Corsica, Corti, France, Tech. Rep. SPE - UMR CNRS 6134.

[6] K.-C. Lin, "Dead reckoning and distributed interactive simulation," in *Distributed interactive simulation systems for simulation and training in the aerospace environment; Proceedings of the Conference*. Orlando, FL, USA, April 1995, pp. 16-36.