**Cell-DEVS**

**Corsica Forest Fire in Cadmium V2**

<https://github.com/murf85/Forest_Fire>

Prepared by:

Mark Murphy

Student ID: 100603616

Assignment 2  
SYSC 5104 – Winter 2025

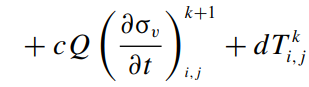
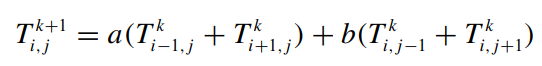
**Part I: Description of the Conceptual Model**

Cell-Devs Forest Fire

Modelling and simulation of forest fires is extremely important to provide decision makers in government information about how to best expend resources to protect civilians and fight the fire. Forest fires are extremely complex and cannot be solved analytically due to the large number of variables and unknowns. Within the fire itself there are phenomena such as radiation, convection and diffusion; there are environmental factors such as wind, humidity and precipitation; and there are terrain factors such as slope, vegetation burn rate and vegetation volume.

The “Corsica” model was first proposed by Muzy et al. [1] to model forest fires as cellular automata (CA) using Cell-DEVS and simulated using CD++. The model was reproduced by Chreyh and MacLeod, they attempted to improve on the computation time using Quantized DEVS in [2]. The researchers were limited by CD++ and the computing power available at the time, Cadmium has many advantages over CD++, and the advances in computing over the last twenty years make it possible to make meaningful advances to the Corsica model.

In the Corsica model the change in temperature of the cell is modelled using this partial differential equation:



Temperature of the cell at location (i,j) at time k+1 is a function of it’s own temperature, the temperature of the cells directly above, below and beside it, and the combustion enthalpy (Q) multiplied by the change in vegetable surface mass (σv), where a,b,c and d are constants which depend on mesh size and the time step chosen. The Corsica implementation uses the Von Neumann neighbourhood, the proposed model will add the diagonal cells, resulting in Moore’s neighbourhood, to facilitate expansion of the model for wind and slope.

This equation intends to model the following curve for the burning of a cell:

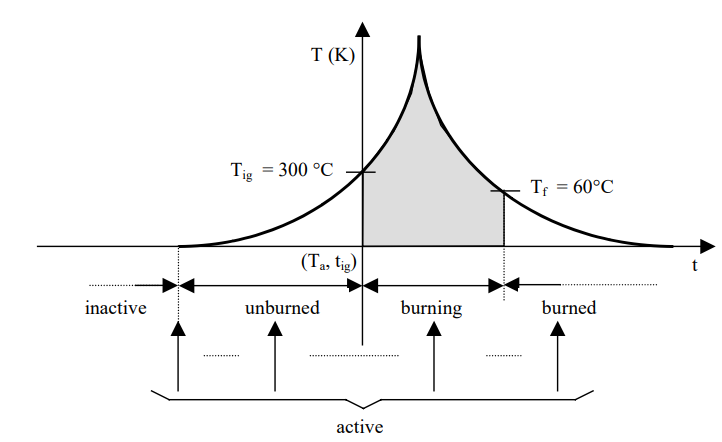


Figure 1. Simplified temperature curve for a cell. [1]

In this model a cell is considered unburned below 300°C, and thus doesn’t include the enthalpy term (Q). The cell ignites above 300°C and the enthalpy term causes the fire to accelerate, and as the fire spreads the neighbouring cells feed off of each other. As is true for a real fire, once the vegetable surface mass is sufficiently depleted, the cell can no longer maintain its extremely high temperatures and cools off, eventually hitting 60°C, at which point it’s considered burned.

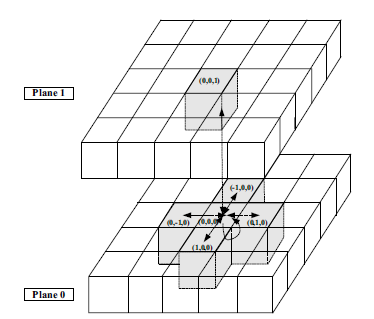
The modelling of the temperature curve follows the same formula used for in the original specification [1], but updated for the expanded neighbourhood:

(1)

Where tig is the time of ignition, and the constants ai,j can be different for each cell in the neighbourhood. The values for the constants were modified slightly from the original model with contribution from the adjacent cells reduced by one quarter and taken from the diagonal cells instead. The proposed implementation allows further adjustment of the constants for wind speed and direction.

**Part II: Modelling in Cadmium**

Due to the limitations of C++, both the original Corsica model and the attempted improvements were required to use multiple planes to implement the model. CD++ only allowed a single state variable per cell, so a second layer was required to signify whether the cell was burning, and if it was burning it contained the ignition time. In addition to the Von Neumann neighbourhood, the cell in the plane above, (0,0,1), was also part of the neighbourhood. The Cell-DEVS model looked like this:



In Cadmium\_v2 each cell can hold multiple state variables, therefore the numbers of cells is reduced by half. The logic of the rules can also be simplified, in the original implementation the rules relied upon whether the temperature was rising or falling to know what stage of burning the cell was in; these steps are no longer required.

This implementation takes advantage of the ability to add factors to the atomic cell model, which is used for wind speed and wind direction. The constants ai,j, introduced in the previous section, are modelled using the vicinity option.

Although Cadmium is supposed to handle the quiescence of cells, an explicit time advance function was added to mimic the initial model, and to overcome unwanted passivation issues which were causing large log files and strange behaviour.

Formal Specification:

|  |
| --- |
| **Corsica Forest Fire Atomic Cell Model <X, Y, S, N, d, τ, δint, δext, λ, ta>** |
| X = Ø  Y = Ø  S = temp (Kelvin) - T **∈** R0+  fire\_status = {0,1,2,3,4} where:  0 - Border cell  1 - Inactive  2 - Unburned  3 - Burning  4 - Burned  t\_ig – t **∈** R0+  sigma – t **∈** R0+  N = {(-1,-1), (-1, 0), (-1, 1),  (0, -1), (0, 0), (0, 1),  (1,-1), (1, 0), (1, 1)} (Moore’s Neighourhood)  d = 1 time unit  τ = Rule 1: (Border cells stay constant and passivate)  if s.fire\_status = 0 then:  s.temp = 300  s.sigma = ∞  Rule 2: (inactive cells remain inactive until temp changes)  if s.fire\_status = 1 then:  Update s.temp using eq. (1)  If new s.temp = 300 then:  s.sigma = ∞  Else:  s.fire\_status = 2  s.sigma = 1.0  Rule 3: (active cells remain active until the reach 300K (passivate) or 573K (ignite))  if s.fire\_status = 2 then:  If s.temp < 573 then:  Update s.temp using eq. (1)  If new s.temp = 300 then:  s.fire\_status = 1  s.sigma = ∞  Else:  s.sigma = 1.0  Else if s.temp >= 573 then:  Update s.temp using eq. (1)  s.fire\_status = 3  s.sigma = 1.0  s.t\_ig = 1.0 (set t\_ig for time t+1)  Rule 4: (Cells burn using enthalpy term until they drop to 333K)  if s.fire\_status = 3 then:  If s.temp > 333:  Update s.temp using eq. (1)  s.sigma = 1.0  s.t\_ig += 1.0  Else if s.temp <= 333:  Update s.temp using eq. (1) (without enthalpy term)  s.fire\_status = 4  s.sigma = 1.0  Rule 5: (Cells cool until they reach 300K, then passivate)  if s.fire\_status = 4 then:  Update s.temp using eq. (1)  If new s.temp = 300:  s.sigma = ∞  Else:  Update s.temp using eq. (1) (without enthalpy term)  s.sigma = 1.0  δint, δext, λ and ta are defined using Cell-DEVS specifications. |

**Part III: Experiments and Results**

In order to verify the model it can be compared to the results listed in the original Corsica model [1] and its recreation for comparison with new updates in [2].

The first test to verify the model uses the scenario presented in [2] where there is a column of fire in a 10x10 grid. The small size of the grid renders the scenario useless for studying the spread of the fire due to the border conditions; however, the authors published the start conditions and temperatures at a number of times. A scenario was build with the Von Neumann neighbourhood, no wind, and identical starting cells, the results were found to be identical, with temperatures of individual cells easily found using the Cell-DEVS viewer:

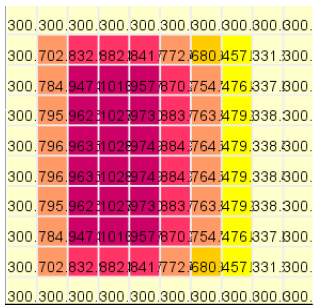
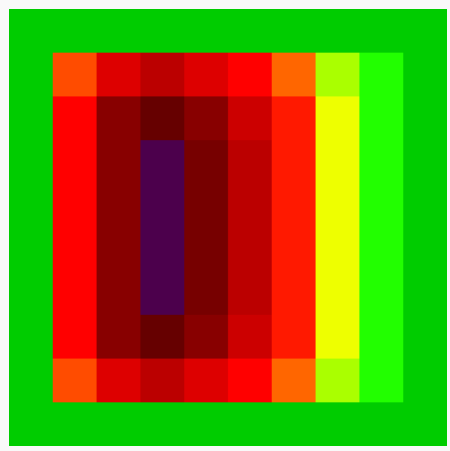


Figure 2: Comparison of proposed model results (left) vs those found in [2] at t =100

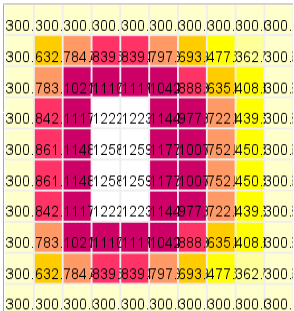
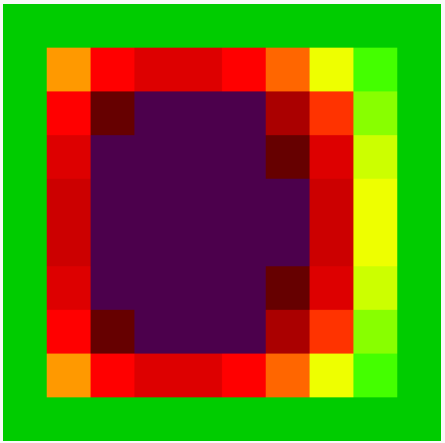


Figure 3: Comparison at t=500

The propagation of a fire outwards from an ignition point is explored in [1], the results are compared to the Von Neumann and Moore neighbourhoods for the proposed implementation:

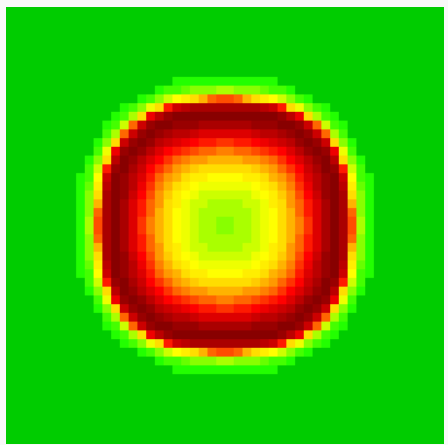
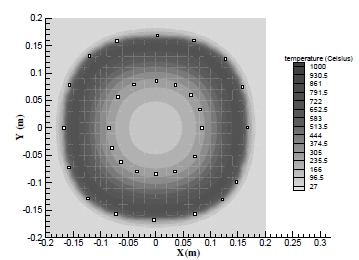


Figure 4: Fire propagation, [1] on left, compared to Von Neumann neighbourhood at t=5000

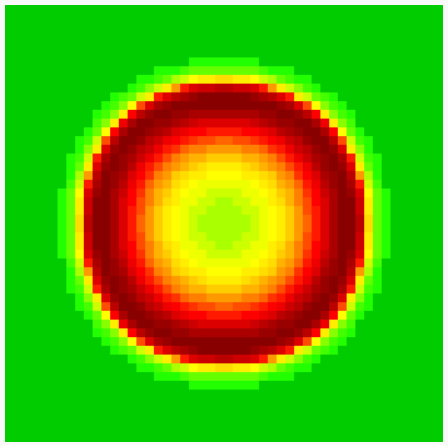
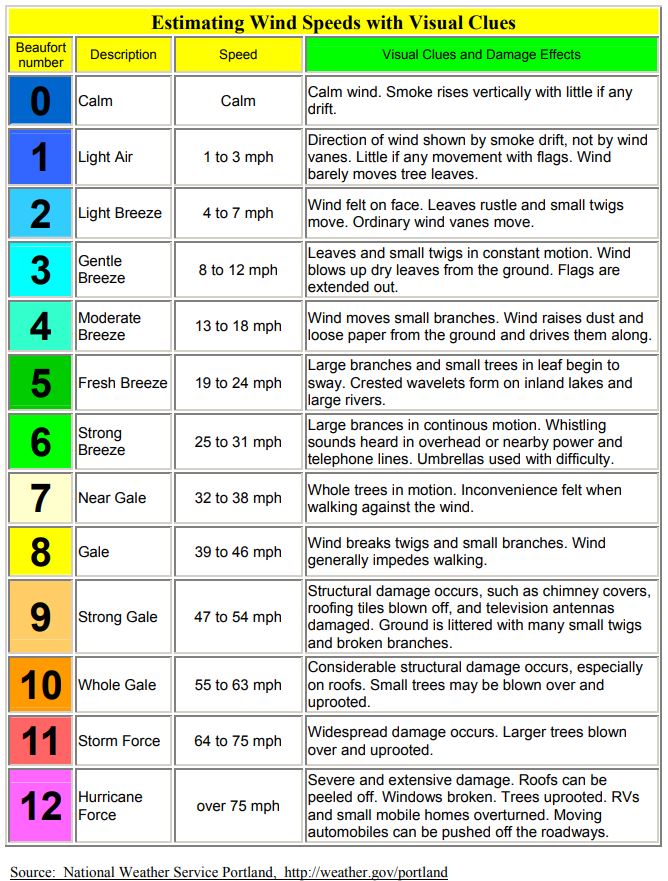


Figure 5: Rounder shape produced using Moore neighbourhood at t = 5000

A switch to Moore’s neighbourhood for the proposed model was necessary for the implementation of wind. This improvement uses a baseline of 30mph, which is based on the chart below, produced by the National Weather Service Portland office:



In order to implement the effects of the wind, the direction of the wind is broken down in to cartesian vectors. Each cell in the neighbourhood’s relative position is multiplied by the cartesian vectors and summed together, this value is multiplied by the scaled wind speed, (windspeed/30), added to 1. The wind factor is then multiplied to modify the constant ai,I. The wind factor is represented with this equation:

(1 + (windspeed/30)(x\*x\_rel + y\*y\_rel)

For example, a 15mph wind from the east produces the following results for various cells:

Wind from the east is expressed as 090°, which is 0° in polar coordinates.

x = cos(ϴ) = cos(0) = 1

y = sin(ϴ) = sin(0) = 0

wind factor from upwind cell (1,0) = (1+0.5((1)(1)+(0)(0)) = 1.25

wind factor from downwind cell (-1,0) = (1+0.5((-1)(0)+(0)(0)) = 0.75

Wind factor from orthogonal cell (0,1) = (1+0.5((0)(1)+(0)(1)) = 1

Wind factor from downwind diagonal cell (-1,-1) = (1+0.5((-1)(1)+(-1)(0)) = 0.75

This example will produce the results in Figure 6 (note that the Cell-DEVS web viewer has the rows and columns switched, so the result below is actually created with a wind from the south - 180°)

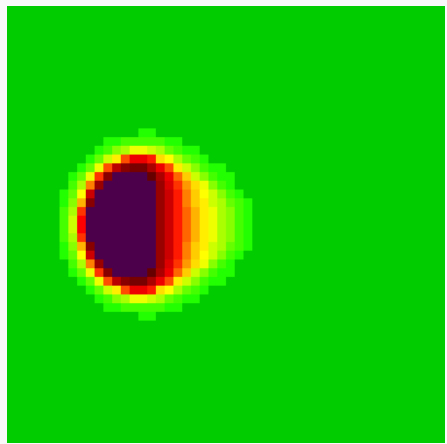


Figure 6: 15mph wind from the east at t=2000

Wind from the south-east is not affected by the flip in rows and columns in the cell viewer, this was used to confirm the proper functioning of diagonal wind and to test the wind speed. Figure 7 shows that the fire moves faster at a high wind speed, as expected.

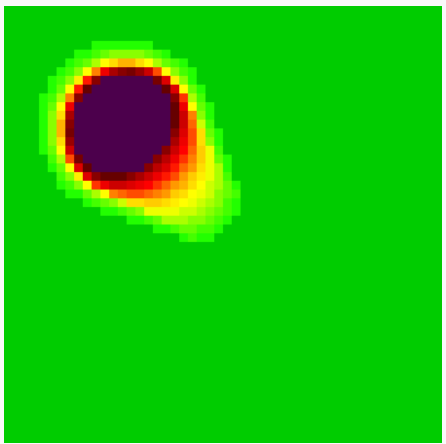
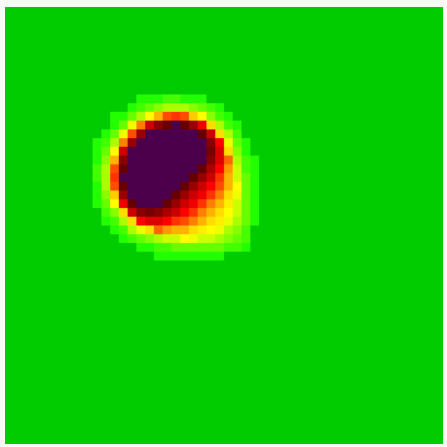


Figure 7: Comparison of wind from SE at 20mph(left) and 40mph at t=1500

**Part IV: Model Limitations and Future Work**

The proposed model demonstrates how wind can be added to the simulation. The wind formulas should be modified further to make them more realistic, especially for the diagonal directions. The example with the wind from the east has an increased effect on the diagonal cells, when logically it shouldn’t. A baseline wind speed was chosen to be 30mph without much rigour, at that speed there is no propagation of the heat upwind at all; in the 15mph example the wind was enough to overcome the upwind spread of fire. With Cadmium it is easier to implement more complex mathematical models for the burn rate and propagation of fire.

A next step for the model is to expand the model to asymmetric spaces, and to couple models together for easier representation of complex land masses. Future work can also involve integrating weather data such as wind and precipitation or terrain data such as slope, elevation and vegetation composition into the models. With the tools of Cadmium there are many options to add new information to the cells, such as vicinity, state variables or config factors. All of these options are easily changed using the json files. For example, the forest fire model proposed by Soulier and Tratnik, <https://github.com/jsoulier/wildfire_simulator/>, uses government maps to load terrain data into json files for use in a cadmium fire simulator. A possible end goal is a tool that can input terrain an weather data to quickly create sim scenarios which can provide decision makers probable fire spread rates and paths for at risk areas.

**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] M. MacLeod, R. Chreyhm, “Project Report: Improving the Corsican Fire Spreading Cell-DEVS Model.” Submitted for SYSC 5104, Carleton University, May 2005.