Simulation of Forest Fire Spread with Different Methods

Group 11

October 2024

1 Progress Description

In this report, we list the models used for fire spread, and their descriptions

1.1 Partial Differential Equation (PDE)

While Cellular Automata focuses on the transition law between neighboring cells within certain landscape [1], PDE emphasizes more on the side of physical law. By generalizing the forest fire phenomenon with certain PDEs based on dynamical principles, we can solve the forest fire problem with a deeper understanding of its dynamics. By far, many PDE models have been suggested, as shown in [2, 3, 4, 5, 6], with most of them having finite difference and Euler methods applied to solve spatial correlation and to integrate over time. However, as noted by Martin & Torres [1], it is challenging to reproduce the codes of methods mentioned above. Therefore, we adopt a relatively more simplified and accessible one as outlined in [1]. More details of the mathematical equations will be discussed in the later submission of the proposal.

1.1.1 Mathematical Equations

In a 2D map Ω, we have the pixel $\boldsymbol{x}$ to be:

$$\boldsymbol{x} = (x, y) \in \Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}], \tag{1}$$

with the boundary $\partial\Omega$ defined as:

$$\partial\Omega \equiv \Gamma. \tag{2}$$

Now the PDEs describing the state of forest fire are:

$$\boldsymbol{v}(\boldsymbol{x}, t) = \boldsymbol{w}(t) + \nabla Z(\boldsymbol{x}), \tag{3}$$

$$u_t = \kappa \Delta u_t - \mathbf{v} \cdot \nabla u_t + \mathcal{H}(u_t - u_{pc}) \beta \exp\left(\frac{u_t}{1 + \varepsilon u_t}\right) - \alpha u_t, \tag{4}$$

$$\beta_t = -\mathcal{H}(u_t - u_{pc}) \frac{\varepsilon}{q} \beta \exp\left(\frac{u_t}{1 + \varepsilon u_t}\right), \tag{5}$$

where:

- $u_t = \dot{u}(t)$ is the temperature changing rate.
- $\beta_t = \dot{\beta}(t)$ is fuel changing rate.
- $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}, t)$.
- $\boldsymbol{w}(t)$ is the effect of wind.
- $Z(\boldsymbol{x})$ is the topography.
- κ is the diffusion coefficient.
- ε is the inverse of activation energy.
- α is the natural convection.
- u_{pc} is the phase change threshold.

• ∇ is the Gradient operator and Δ is the Laplacian operator.
• $\mathcal{H}()$ is Heaviside step function.

The initial conditions include:

$$\begin{aligned} &\boldsymbol{w}(t), Z(\boldsymbol{x}), \\ &\boldsymbol{u}(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \\ &\boldsymbol{\beta}(\boldsymbol{x}, 0) = \beta_0(\boldsymbol{x}). \end{aligned} \tag{6}$$

In another words

With a pre-setting that forest fire will not spread outside Ω, we have the Dirichlet boundary conditions as follows:

$$u_\Gamma(\boldsymbol{x}, t) = \beta_\Gamma(\boldsymbol{x}, t) = 0 \tag{7}$$

A summary of the state variables and map can be seen in Fig. 1

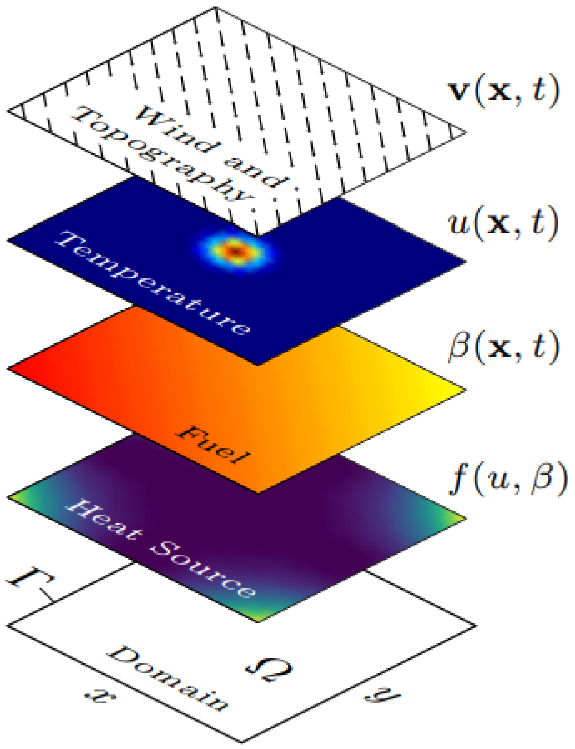


Figure 1: Visualization of the state variables and spatial map in the forest fire simulation using the PDE method [1].

1.1.2 Numerical Solver

To solve the PDEs, we need to deal with both spatial and temporal parts. For spatial part, we use spatial discretization. For temporal part, we could simply use forward eular if the interaction time step dt is small enough.

1.1.3 PDE: New Innovations

Instead of solely solving PDEs with numerical methods, we are conducting a comparative analysis of three methods, evaluating their computational efficiency and accuracy. What's more, by connecting the parameters used in the PDE method with those in the other two, we aim to explore parameter sensitivity to gain a deeper understanding of the physical mechanism of the forest fire.

1.2 Agent-Based Modeling (ABM)

To perform the agent based simulation of wildfire spread, we start off by using AgentPy package in python for ease of creating models, environments, and visualizations. Our vanilla model starts from including tree density as the only agent. The model is based on the classic NetLogo FireSimple model, which captures the effect of tree density on the likelihood of a fire spreading across a forest. We outline the base model that we start with, and state how we build complexity off of this model.

The forest is represented as a grid, with each cell containing a tree agent that can be in one of three states: Alive (0): The tree has not caught fire. Burning (1): The tree is on fire, spreading to neighboring cells. Burned (2): The tree is completely burned and no longer spreads the fire. The fire starts on the left edge of the grid and spreads through neighboring cells in the four cardinal directions (north, south, east, west), based on the state of each adjacent cell. This basic model simulates the movement of fire through a static forest without wind, which means that fire can only spread through connected areas of trees.

The following are the key Parameters that we start to use in the model. (i) Tree Density: This defines the percentage of cells that contain trees, affecting how connected and continuous the forest is. When the density is high, the fire is more likely to move across the grid, while lower densities may create natural breaks that stop the fire from spreading. (ii) Grid Size: The dimensions of the forest. (iii) Steps: The maximum time steps for a simulation run, though the model stops early if the fire extinguishes. We now represent the simulation in terms of mathematical notation using relevant equations and conditionals.

The forest is represented on a 2D grid of size $N \times N$, where each cell (i, j) represents a location that can contain a tree or be empty.

Let:

- D be the tree density parameter, so the total number of trees is approximately $D \cdot N^2$.
- The state of each cell is given by $S_{i,j}$:

$$S_{i,j} = \begin{cases} 0 & \text{if the tree is alive (green)} \\ 1 & \text{if the tree is burning (red)} \\ 2 & \text{if the tree is burned (gray)} \end{cases}$$

The fire starts by setting trees in the left-most columns of the grid to a burning state:

$$S_{i,j} = 1 \quad \text{for initial cells along the left edge.}$$

At each time step t, for every cell (i, j) with $S_{i,j}(t) = 1$ (burning trees):

- The fire attempts to spread to each of the four neighboring cells (north, south, east, and west):

$$S_{i',j'}(t+1) = 1 \quad \text{if} \quad S_{i',j'}(t) = 0 \quad \text{and} \quad (i', j') \in \{\text{neighbors of } (i, j)\}.$$

  Each neighboring cell with an alive tree ($S_{i',j'} = 0$) becomes burning with a fixed probability of spread.

- Once a tree has spread fire to its neighbors, it transitions to the burned state:

$$S_{i,j}(t+1) = 2.$$

The simulation stops if there are no burning trees left:

$$\sum_{i,j} \mathbf{1}(S_{i,j} = 1) = 0$$

where $\mathbf{1}(S_{i,j} = 1)$ is an indicator function that is 1 if $S_{i,j} = 1$ (burning) and 0 otherwise.

At the end of the simulation, the percentage of trees burned is calculated as:

$$\text{Burned Percentage} = \frac{\sum_{i,j} \mathbf{1}(S_{i,j} = 2)}{\text{Total trees}}.$$

This metric provides a measure of the fire's impact across the forest grid.

If wind is added, it can be represented by a directional bias, $w \in \{\text{north, south, east, west}\}$, which influences the probability of spread to neighboring cells based on the wind direction. The probability of spread could be adjusted as follows:

$$P_{\text{spread}}(i', j') = \begin{cases} p & \text{if spread is against wind direction} \\ 1.5p & \text{if spread is with wind direction} \end{cases}$$

where p is the baseline spread probability.

2 An Update of the Current States

2.1 Show of Progress

2.1.1 PDE

A basic setup codes for PDE method is shown in fig. 2 and we will implement more details in next checkpoint.

2.1.2 Agent-based model

We show a preliminary implementation of the agent based model in fig. 3 on which we build futher complexity.

To enhance the agent-based model and align it more closely with the Partial Differential Equation (PDE) model, we can incorporate additional agents and factors such as fuel availability, temperature, wind influence, and topography instead of just deterministically/probabilistically propagating the fire across the cells.

Each cell has defined fuel β and temperature u. Fuel depletes as the fire spreads, with an exponential decay influenced by temperature, moderated by the Heaviside function $\mathcal{H}(u - u_{pc})$ to limit burning only to cells exceeding the ignition threshold. Including topography introduces a gradient $Z(\mathbf{x})$, directing fire spread according to elevation changes. Finally, moisture content modifies each cell's ignition threshold u_{pc}, creating natural barriers by requiring higher ignition temperatures in regions with higher moisture.

In a fire spread model, intervention scenarios can be applied to mitigate the progression of the fire through modifications to fuel levels, ignition thresholds, or wind effects.

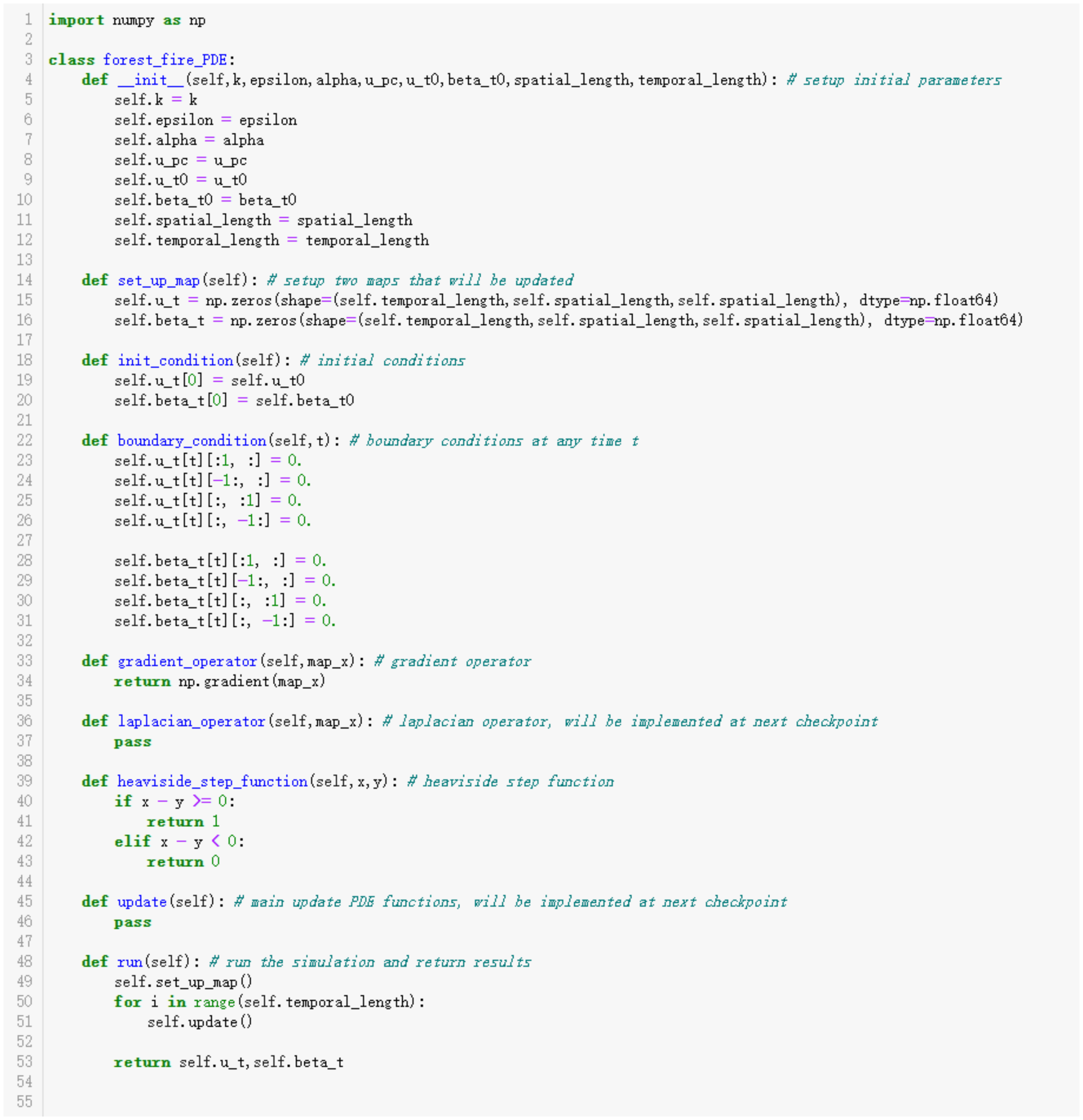
To simulate fuel reduction (via interventions or natural agents) in specific areas, such as creating firebreaks or executing controlled burns, we define a reduction in fuel levels in regions $A_{\text{intervene}} \subset \Omega$ (where Ω represents the forest area). Mathematically, this is represented as:

$$\beta(\mathbf{x}) = \beta_{\text{original}}(\mathbf{x}) - \gamma \cdot \mathcal{I}_{A_{\text{intervene}}}(\mathbf{x}),$$

where γ is the fuel reduction coefficient, and $\mathcal{I}_{A_{\text{intervene}}}(\mathbf{x})$ is an indicator function, equal to 1 if $\mathbf{x} \in A_{\text{intervene}}$ and 0 otherwise.

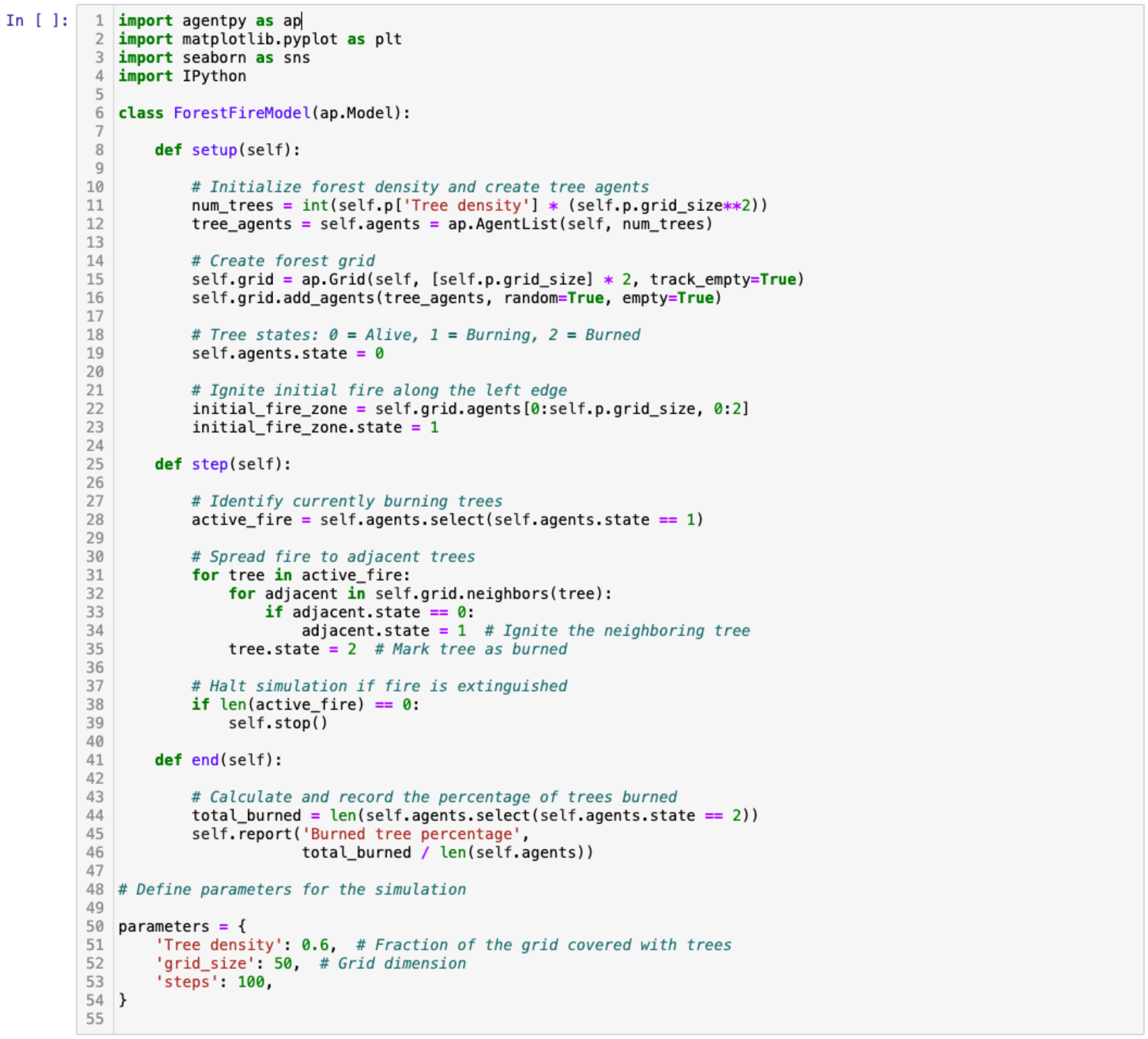
For moisture control through hydration interventions, such as applying water or retardants, we raise the ignition threshold u_{pc} in designated areas $A_{\text{hydration}}$ to slow down fire spread:

$$u_{pc}(\mathbf{x}) = u_{pc,\text{original}}(\mathbf{x}) + \delta \cdot \mathcal{I}_{A_{\text{hydration}}}(\mathbf{x}),$$



```python
import numpy as np

class forest_fire_PDE:
    def __init__(self,k,epsilon,alpha,u_pc,u_t0,beta_t0,spatial_length,temporal_length): # setup initial parameters
        self.k = k
        self.epsilon = epsilon
        self.alpha = alpha
        self.u_pc = u_pc
        self.u_t0 = u_t0
        self.beta_t0 = beta_t0
        self.spatial_length = spatial_length
        self.temporal_length = temporal_length

    def set_up_map(self): # setup two maps that will be updated
        self.u_t = np.zeros(shape=(self.temporal_length,self.spatial_length,self.spatial_length), dtype=np.float64)
        self.beta_t = np.zeros(shape=(self.temporal_length,self.spatial_length,self.spatial_length), dtype=np.float64)

    def init_condition(self): # initial conditions
        self.u_t[0] = self.u_t0
        self.beta_t[0] = self.beta_t0

    def boundary_condition(self,t): # boundary conditions at any time t
        self.u_t[t][:1, :] = 0.
        self.u_t[t][-1:, :] = 0.
        self.u_t[t][:, :1] = 0.
        self.u_t[t][:, -1:] = 0.

        self.beta_t[t][:1, :] = 0.
        self.beta_t[t][-1:, :] = 0.
        self.beta_t[t][:, :1] = 0.
        self.beta_t[t][:, -1:] = 0.

    def gradient_operator(self,map_x): # gradient operator
        return np.gradient(map_x)

    def laplacian_operator(self,map_x): # laplacian operator, will be implemented at next checkpoint
        pass

    def heaviside_step_function(self,x,y): # heaviside step function
        if x - y >= 0:
            return 1
        elif x - y < 0:
            return 0

    def update(self): # main update PDE functions, will be implemented at next checkpoint
        pass

    def run(self): # run the simulation and return results
        self.set_up_map()
        for i in range(self.temporal_length):
            self.update()

        return self.u_t,self.beta_t


```

Figure 2: Base implementation of PDE for forest-fire spread



```python
import agentpy as ap
import matplotlib.pyplot as plt
import seaborn as sns
import IPython

class ForestFireModel(ap.Model):

    def setup(self):

        # Initialize forest density and create tree agents
        num_trees = int(self.p['Tree density'] * (self.p.grid_size**2))
        tree_agents = self.agents = ap.AgentList(self, num_trees)

        # Create forest grid
        self.grid = ap.Grid(self, [self.p.grid_size] * 2, track_empty=True)
        self.grid.add_agents(tree_agents, random=True, empty=True)

        # Tree states: 0 = Alive, 1 = Burning, 2 = Burned
        self.agents.state = 0

        # Ignite initial fire along the left edge
        initial_fire_zone = self.grid.agents[0:self.p.grid_size, 0:2]
        initial_fire_zone.state = 1

    def step(self):

        # Identify currently burning trees
        active_fire = self.agents.select(self.agents.state == 1)

        # Spread fire to adjacent trees
        for tree in active_fire:
            for adjacent in self.grid.neighbors(tree):
                if adjacent.state == 0:
                    adjacent.state = 1  # Ignite the neighboring tree
            tree.state = 2  # Mark tree as burned

        # Halt simulation if fire is extinguished
        if len(active_fire) == 0:
            self.stop()

    def end(self):

        # Calculate and record the percentage of trees burned
        total_burned = len(self.agents.select(self.agents.state == 2))
        self.report('Burned tree percentage',
                    total_burned / len(self.agents))

# Define parameters for the simulation

parameters = {
    'Tree density': 0.6,  # Fraction of the grid covered with trees
    'grid_size': 50,  # Grid dimension
    'steps': 100,
}

```

Figure 3: Base implementation of ABM for forest-fire spread

where δ represents the increase in the ignition threshold in these regions, making it harder for fire to initiate and spread across hydrated cells.

Each intervention acts locally on the parameters of the model, adjusting the spatial characteristics of fire dynamics in real-time.

2.2 Github Link

We have set up a public Git Repository for this project, and the link is https://github.gatech.edu/xan37/CSE-6730-Group-Project. We modified the permissions so that TAs and the professor are the collaborators.

2.3 Division of Labors

The division of labors is shown as follows:

- PDE: An, Xiaodong (Modeling and Writing), Avasarala, Srikanth (Parameter Searching)
- ABM: Ray, Pranoy (Modeling and Writing), Avasarala, Srikanth (Parameter Searching)
- Git Repository: An, Xiaodong (Setup and Collaborator), Avasarala, Srikanth (Collaborator), Ray, Pranoy (Collaborator)

References

[1] D. San Martin and C. Torres, “2d simplified wildfire spreading model in python: From numpy to cupy,” *CLEI electronic journal*, vol. 26, no. 1, pp. 5–1, 2023.

[2] R. Montenegro, A. Plaza, L. Ferragut, and M. Asensio, “Application of a nonlinear evolution model to fire propagation,” *Nonlinear Analysis*, vol. 30, no. 5, pp. 2873–2882, 1997.

[3] M. Asensio and L. Ferragut, “On a wildland fire model with radiation,” *International Journal for Numerical Methods in Engineering*, vol. 54, no. 1, pp. 137–157, 2002.

[4] L. Ferragut, M. Asensio, and S. Monedero, “Modelling radiation and moisture content in fire spread,” *Communications in Numerical Methods in Engineering*, vol. 23, no. 9, pp. 819–833, 2007.

[5] L. Ferragut, M. I. Asensio, and S. Monedero, “A numerical method for solving convection–reaction–diffusion multivalued equations in fire spread modelling,” *Advances in Engineering Software*, vol. 38, no. 6, pp. 366–371, 2007.

[6] J. Mandel, L. S. Bennethum, J. D. Beezley, J. L. Coen, C. C. Douglas, M. Kim, and A. Vodacek, “A wildland fire model with data assimilation,” *Mathematics and Computers in Simulation*, vol. 79, no. 3, pp. 584–606, 2008.