EFFICIENT ALGORITHMS FOR WILDLAND FIRE SIMULATION

by

VOLODYMYR Y. KONDRATENKO

Bachelor of Science, Taras Shevchenko Kyiv National University, 2008
Master of Science, University of Colorado Denver, 2010

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado Denver in partial
fulfillment of the requirements for the degree of
Doctor of Philosophy
Applied Mathematics
2015
This thesis for the Doctor of Philosophy degree by
Volodymyr Y. Kondratenko
has been approved for the
Department of Mathematical and Statistical Sciences
by

Jan Mandel, Advisor
Julien Langou, Chair
Loren Cobb
Adam K. Kochanski
Weldon Lodwick

April 24, 2015
Kondratenko, Volodymyr Y. (Ph.D., Applied Mathematics)
Efficient Algorithms for Wildland Fire Simulation
Thesis directed by Professor Jan Mandel

ABSTRACT

In this dissertation, we develop the multiple-source shortest path algorithms and examine their application importance in real world problems, such as wildfire modeling. The theoretical basis and its implementation in the Weather Research Forecasting (WRF) model coupled with the fire spread code SFIRE (WRF-SFIRE model) are described.

We present a data assimilation method that gives the fire spread model the ability to start the fire simulation from an observed fire perimeter instead of an ignition point. While the model is running, the fire state in the model changes in accordance with the new arriving data by data assimilation. As the fire state changes, the atmospheric state (which is strongly effected by heat flux) does not stay consistent with the fire state. The main difficulty of this methodology occurs in coupled fire-atmosphere models, because once the fire state is modified to match a given starting perimeter, the atmospheric circulation is no longer in sync with it. One of the possible solutions to this problem is a formation of the artificial time of ignition history from an earlier fire state, which is later used to replay the fire progression to the new perimeter with the proper heat fluxes fed into the atmosphere, so that the fire induced circulation is established.

In this work, we develop efficient algorithms that start from the fire arrival times given at the set of points (called a perimeter) and create the artificial fire time of ignition and fire spread rate history. Different algorithms were developed in order to suit possible demands of the user, such as implementation in parallel programming,
minimization of the required amount of iterations and memory use, and use of the rate of spread as a time dependent variable. For the algorithms that deal with the homogeneous rate of spread, it was proven that the values of fire arrival times they produce are optimal. It was also shown that starting from arbitrary initial state the algorithms have finite convergence and the amount of iterations was estimated.

Application of the method on real tests, based on the data taken from the observed fires, has shown the high accuracy of the algorithm and its usefulness. Besides wildfire modeling, this technique has a high application value in different fields, such as epidemiology, demographics control, flood modeling, etc.

The form and content of this abstract are approved. I recommend its publication.

Approved: Jan Mandel
DEDICATION

Dedicated to my loving parents and sister.
ACKNOWLEDGMENT

I would like to thank all the people whom I have met on my path of growing as a mathematician and a scientist, and without whom my thesis would never have been possible. Particularly, I wish to thank my teacher, Prof. Valentin Leifura, from Petro Mohyla Mykolaiv University, who helped me achieve high results in the Ukrainian Mathematical Olympiad competitions, and my undergraduate advisor, Prof. Yuriy Krak, from Taras Shevchenko Kyiv National University. I would like to thank my graduate adviser, Prof. Jan Mandel, from the University of Colorado, for his contribution, encouragement and mentorship that greatly helped me to grow as a scientist and made the creation of this thesis possible. I would also like to thank my lovely parents, Yuriy and Galyna, who encouraged me to take the first steps in academia, and my dearest sister Nina for all her support and companionship. Finally, I thank all the professors, fellow students and staff of the department of Mathematical and Statistical Sciences for their company and collaboration in discussing various research ideas over the past several years.
TABLE OF CONTENTS

Figures ......................................................... ix

1. Introduction ............................................. 1

2. Wildfire Modeling ........................................ 5
  2.1 Coupled Models ......................................... 5
  2.2 Propagation Methods ................................... 7
  2.3 Ignition Problem and Atmospheric Balance ........... 9
  2.4 WRF-SFIRE Software ................................... 11

3. Propagation on a Mesh .................................... 14
  3.1 Existing Methods ....................................... 14
  3.2 Distance Based Method ................................ 16
    3.2.1 Computational Results ............................ 18
    3.2.2 Conclusion ........................................ 20
  3.3 Forward Method ........................................ 22
    3.3.1 Simple Forward Propagation Method ............... 22
    3.3.2 Fast Forward Propagation Algorithm ............... 30
  3.4 Backward Method ....................................... 35
    3.4.1 Simple Backward Propagation Algorithm ......... 36
    3.4.2 Fast Backward Propagation Algorithm ............. 44
  3.5 Time Dependent Method ................................ 50
    3.5.1 Backward Propagation .............................. 50
    3.5.2 Forward Propagation ............................... 53

4. Application to Fires: Perimeter Ignition ............... 56
  4.1 Initialization from a Fire Perimeter ................. 56
  4.2 Creating Artificial Ignition Time History from the Current Fire Perimeter 56
  4.3 Encoding and Replaying the Fire History ............. 57
  4.4 Results ................................................. 58
FIGURES

Figure

2.1 One $2 \times 2$ tile with the lowest layer of the atmospheric grid and the fire mesh on the surface shown. Wind vector components $u$, $v$, $w$ are located at the midpoints of the sides of the atmospheric grid cells. Reproduced from [20].

3.1 A level set function linear on the line segments connecting the nodes of the fire mesh cannot be defined at the nodes $X$ and $Y$ consistently as the signed distance (3.1) from the interface $\Gamma$. The distance of the point $X$ from $\Gamma$ does not depend on the location of the point $Z$, while the distance of $Y$ does; yet the values of the level set function at $X$ and $Y$ are linear along the segment $XY$ and so fixed by the ratio of their distances from $W$. Reproduced from [21].

3.2 Construction of intersection of the fire perimeter and the half line originating from the Ignition point and passing through a given mesh point. Reproduced from [17].

3.3 The difference in the U and V wind components, computed at 6.1m, of the direct simulation minus the artificial propagation at 68 minutes. Reproduced from [17].

3.4 The relative error in the horizontal wind speed at 6.1m above the ground at 68 minutes. Reproduced from [17].

3.5 The direct fire simulation at 68 minutes. Reproduced from [17].

3.6 (a) The situation when both forward and backward propagations give the same results. (b) Since there are no outgoing arrows from the point $C$, the value of the fire arrival time at this point will differ for forward and backward propagations.
3.7 Fire line at step $t_k$, where the fire covered distance $d$ on the way from point B to point A. $d' = ROS(t, C, \vec{BA})$.

4.1 (a) Propagation of ignition time $t$ to a node from neighboring nodes already on fire. (b) Backtracking (propagation back in time) of ignition time to a node from neighboring nodes where the fire arrived later. Reproduced from [16].

4.2 Perimeter of the 2007 Santa Ana fires simulation October 22, 2007 1PM PDT [16]. The fire consisted of two fires, Witch and Guejito, which started on October 21, 2007 12:15 PM PDT and 22 October 2007 1:00 AMPDT, respectively, and subsequently merged. The asterisks and circles from left to right indicate the real and artificially calculated ignition points of Guejito and Witch fires respectively. Reproduced from [16].

4.3 Artificial fire arrival time found by fire propagation back in time from the fire perimeter in 4.2. The two peaks on the bottom, marked by arrows, are the two ignition locations and times, found automatically from the perimeter. The vertical axis and the false color are the time from the beginning of the simulation. Reproduced from [16].

4.4 (a) Horizontal wind at 6.1 m in the 2007 Santa Ana fires simulation on October 2007 1:00PM PDT. (b) The same wind as in (a), but with the artificial ignition time history from Fig. 4.3 until October 2007 1:00PM PDT. (c) The difference of (a) and (b). The RMSE is $1.1 ms^{-1}$, which is 8.8% of the maximal wind speed $12.53 ms^{-1}$. Reproduced from [16].
4.5  (a) Fire perimeter in the 2007 Santa Ana fires simulation on 22 October 22, 2007 9PM PDT. (b) The same perimeter as in (a), but with the artificial ignition time history from simulation data (Fig. 4.3) on October 22, 2007, 1PM PDT. The differences in the simulation 8 hours later are only minor, such as the protuberance at the northeast part of the perimeter. Reproduced from [16].

4.6  (a) The ignition point of the fire driven simulation. (b) Perimeter of the fire, taken after the fire driven simulation was running for 30 minutes. The false color represents the amount of fuel burnt. Problem setup and visualization done by Adam Kochanski.

4.7  Artificial fire arrival time found by the backward fire propagation algorithm which originated from the fire perimeter in 4.6(b) and was using no fire driven rate of spread in (a) and fire driven rate of spread in (b). The vertical axis and the false color are the time from the beginning of the simulation.

4.8  Fire arrival time of the original simulation. (b) Difference between fire arrival times of the original simulation and the simulation with no fire affect.

4.9  Location of the real ignition point, the one found with perturbed winds, and the ignition point with winds without fire in relation to the contour of the perimeter.

5.1  The morphing Kalman Filter (EnKF) with 5 ensemble members, applied to the ground heat flux from the WRF-Fire model. The ensemble size is not sufficient, the correct analysis is not even approximately in the span of the forecast, and the EnKF cannot reach it.
5.2 The morphing FFT EnKF with 5 ensemble members, applied to the
ground heat flux from the WRF-Fire model. The analysis ensemble moved
towards the data. Reproduced from [22].

6.1 Division of fire mesh cells into subcells for fuel fraction computation.
The level set function $\psi$ and the ignition time $t_i$ are given at the cen-
ters $a_1, \ldots, a_4$ of the cells of the fire grid. The integral (6.5) over the cell
$C$ with the center $a_3$ is computed as the sum of integrals over the subcells
$C_1, \ldots, C_4$. While the values of $\psi$ and $t_i$ are known at $a_3 = x_3$, they need
to be interpolated to the remaining corners $x_1, x_2, x_4$ of the subcell $C_1$
from their values at the points $a_1, \ldots, a_4$. Reproduced from [20].
1. Introduction

Forests play a very important role in our planet’s ecosystem by mitigating such ecological problems as air pollution, erosion and groundwater deficiency. However, wildfires that are a part of the natural life cycle of a forest can get out of control and cause major damage to forests, its species and endanger lives and properties.

Wildland fire is a complex multiscale process. Some aspects wildland fire behavior can be captured by the coupling of a mesoscale weather model with a simple 2D fire spread model. Weather has a major influence on the wildfire behavior. In particular, the fire spread depends on several factors, such as wind, atmospheric humidity, type of fuel, topography, etc. Sections 2.1 and 2.2 provide an introduction to different coupled atmosphere-fire models and propagation methods used in wildfire modeling.

The coupled WRF and SFIRE software, described in section 2.4, combines the Weather Research, Forecasting Model (WRF) and fire spread model (SFIRE), is intended to be faster than real time in order to deliver a forecast.

The original ignition mechanism of WRF-FIRE model allows only point and line ignitions. However, usually the initial information about the location of the fire arrives when the fire has already been burning for a certain period of time. In that case, the model either has to guess the ignition point, which significantly drops the accuracy of the model, or has to set the whole fire perimeter on fire, which causes the inconsistency between the atmospheric and fire states.

One of the possible solutions would be to generate the artificial fire history based on the given fire perimeter and time, the fuel map and the state of the atmosphere in the time period before the perimeter time. Using one of the fire propagation algorithms described in chapter 3, the artificial time of ignition history is created and is hypothesized to be close to the real fire time of ignition. We then use the artificial fire arrival time instead of the fire spread model to burn the fuel and generate the heat release to the atmosphere. Replaying the artificial fire history enables gradual
fuel burn, instead of igniting the whole area inside the fire perimeter at once. This process allows the fire-induced atmospheric circulation to develop. At the perimeter time, the complete coupled atmosphere-fire model takes over.

Our current approach consists of reversing the direction of time in a fire spread method, which will result in shrinking the fire to one or more ignition points. \( \mathcal{M} \), that defines a directed graph \( \mathcal{G}(\mathcal{M}, \mathcal{E}) \), where \( \mathcal{E} \) contains the edge \( AB \) if \( \Delta t_{AB} < \infty \). Here A and B are two neighboring points, \( A, B \in \mathcal{M} \) and \( \mathcal{E} \) has no looping edges \( (\Delta t_{AA} = 0) \). The shortest path to the given point is the minimal travel time along the neighbors from the current fire perimeter, such that the sum of weights along the edges is minimized. In chapter 3 we developed the multiple-source shortest path algorithms, which are suitable for time reversal. The methods determine the ignition time at a node as the earliest time the fire can get to that node from the nodes that are already burning. In particular, in Section 3.2 the fire arrival times inside a given convex or star-shaped perimeter were approximated based on the distance from a known ignition point to the perimeter, while use of the reinitialization equation was proposed in [21]. In the ideal case, where all the data is taken from the model simulation output, which originates from a predefined ignition point and was using the homogeneous rate of spread, the simulation results show that the fire can continue in a natural way from a defined perimeter.

Subsequently, the author develops more advanced backward propagation methods using homogeneous input data in Section 3.4 where two algorithms were developed to serve different purposes. Data assimilation techniques, which are the processes during which new data is incorporated into the model, require the realization of the forward propagation algorithms that are described in Sections 3.3.

Simple forward and backward propagation algorithms described in 3.3.1 and 3.4.1 are constructed in a way that makes them suitable for parallel computing, since most of the computations can be performed simultaneously. Fast propagation methods de-
scribed in sections 3.3.2 and 3.4.2 are more efficient in terms of speed, memory saving and decreasing the amount of iterations, however the construction of the algorithms makes it hard to implement them for a parallel computing. For all of the above algorithms it is shown that they converge to a unique fixed point in a finite amount of steps.

The advantage of the methods described in section 3.5 is that they let the rate of the fire spread to be a non homogeneous (time dependent) variable. This makes these methods applicable in the real world problems, where such variables as wind and moisture are changing continuously.

Simulation results in the real case study were performed with the 2007 Santa Ana fires (see Section 4) and the errors in ignition point location and ignition times map between the simulation results and observed data were computed. Relatively small errors demonstrated that the model that uses the artificial fire history has minor differences from the original model that starts at the given ignition point.

In section 4.4.2 we propose the simulation of an ideal case presenting how the algorithm performs in case of absence of actual fire propagation simulation. The results of using the fire rate of spread data, not affected by the fire, in Extended Fast Backward propagation algorithm, are not as precise, comparing to the results based on the fire driven data. This shows that the impact of the fire effect on the wind plays an important role for the backward fire propagation algorithm.

Chapters 5 and 6 talk about research projects efficient for the fire simulation. In particular, in Chapter 5 we propose a new type of the Ensemble Kalman Filter (EnKF), which uses the Fast Fourier Transform (FFT) for covariance estimation from a very small ensemble with automatic tapering, avoiding the need to solve a sparse system with a tapered matrix. Chapter 6 describes the method of fuel fraction computation, which is realized by interpolation of the time of ignition of the cell, for a case when the cell is partially burning.
Parts of this thesis are based on the following papers. Section 2.1 is based on papers [17] and [21]. Section 2.4 and chapter 6 are based on [20]. Section 3.2 is based on [17]. Sections 3.3 - 3.5 are new. Chapter 4 is using parts of [16]. Chapter 5 is based on [23].
2. Wildfire Modeling

2.1 Coupled Models

Wildland fire is a complicated multiscale process. Nowadays, a wide range of wildland fire behavior properties can be tracked by coupling of a weather model with a 2D fire spread model \([5, 7, 6, 8]\). Wildfire behavior strongly depends on weather conditions, in particular, wind and moisture have a high impact on the rate of spread fire. On the other side, the fire influences the weather through the heat and vapor fluxes caused by burning hydrocarbons and evaporation of the fuel moisture. The heat from a fire can cause tornadic strength winds, and the wind and the moisture from the fire can affect the atmosphere layers located away from the fire.

The model considered here couples the atmospheric code WRF-ARW \([33]\) with a fire spread module SFIRE. SFIRE is based on modified Rothermel’s formula \([30]\), which approximates the fire spread rate in the normal direction to the fireline as a function of fuel properties, the component of the wind vector in normal direction close to the ground, and terrain slope:

\[
R = \min\{B_0, R_0 + \phi_W + \phi_S\},
\]

where \(B_0\) is the spread rate against the wind, \(R_0\) is the spread rate in the absence of wind, \(\phi_W = a(\vec{v} \cdot \vec{n})^b\) is the wind correction, and \(\phi_S = d\nabla z \cdot \vec{n}\) is the terrain correction. Further, \(\vec{v}\) is the wind vector, \(\nabla z\) is the terrain gradient vector, and \(\vec{n}\) is the normal vector to the fireline pointed away from the burning area.

Reduction of the fuel mass during burning is approximated as an exponential decay in time. At each time step, the fire model inputs the atmospheric winds and outputs heat fluxes into the atmosphere. The finest possible domain in WRF is coupled with the fire model. However, the fire effect (for instance smoke) is also visible in the parent domains.

For each node of the fire grid the state variables of the fire model are calculated. In particular, level set function \(\Phi\), time of ignition \(T_1\) and remaining fuel fraction \(F\) are
given by their values on the nodes of the fire model mesh. At a given simulation time \( t \), the fire area is represented by the level set function \( \Phi \) as the set of all points \((x, y)\) where \( \Phi (t, x, y) \leq 0 \). The ignition time \( T_i = T_i (x, y) \) has to satisfy the consistency conditions

\[
\begin{cases}
  T_i (x, y) \leq t, & \text{if } (x, y) \text{ is burning at time } t \text{ or has burnt already} \\
  T_i (x, y) > t, & \text{if } (x, y) \text{ was not set on fire yet ,}
\end{cases}
\]  

(2.1)

which in terms of the level set function can be described as

\[
\begin{cases}
  \Phi (t, x, y) \leq 0, & \text{if } (x, y) \text{ was set on fire before time } t \\
  \Phi (t, x, y) \leq 0 \text{ if } (x, y) \text{ was not ignited yet .}
\end{cases}
\]  

(2.2)

Each time step of the simulation, the advance of the level set function is computed by a Runge-Kutta scheme for the level set equation

\[
\frac{d\Phi}{dt} = -R|\nabla \Phi|,
\]  

(2.3)

where \( R = R(t, x, y) \) is the fire rate of spread and \(|\cdot|\) is the Euclidean norm. From the level-set representation of the fireline at the time \( t \), as \( \Phi (t, x, y) = 0 \), it follows that the normal direction is \(-\nabla \Phi/|\nabla \Phi|\), therefore

\[
R = R \left( u \cdot \frac{\nabla \Phi}{|\nabla \Phi|} \right),
\]  

(2.4)

where \( u \) is the wind field.

While the fuel is burning, the remaining mass fraction \( F = F(t, x, y) \) is approximated by exponential decay,

\[
F(t, x, y) = \begin{cases} 
  F_0 (x, y) \cdot \exp \left( -\frac{t-T_i (x, y)}{T_f (x, y)} \right), & t > T_i (x, y), \\
  1, & t \leq T_i (x, y),
\end{cases}
\]

(2.5)

where \( T_f \) is the fuel burn time, that represents the number of seconds that takes the fuel to burn down to \( 1/e \approx 0.3689 \) of the starting fuel fraction \( F = 1 \).
At each time step of the atmospheric model the winds are interpolated from the atmospheric model grid to a finer fire model grid. A numerical scheme for the level set equation (2.3) is then advanced to the next time step value, and the fuel, burned during the time step, is computed by taking an integral of (2.5) in each fire model cell. The average of the heat fluxes is calculated over the fire cells that make up one atmosphere model cell.

WRF can be run with several nested refined domains, which can run different physical models. SFIRE uses the WRF infrastructure for parallel computing and for data management. The coupled code performs faster than real time and requires at least several hundred cores, with the fire model resolution of few meters and horizontal atmospheric resolution on the order of 100m, for a large real fire [14]. Another WRF feature is the ability to export and import state, in order to implement data assimilation (input of additional data while the model is running), which is essential for fire behavior prediction from all the newest available data. Fuel data and topography can be obtained from government databases in the United States, from satellite images and GIS elsewhere [14], etc. The model is available from the Open Wildland Fire Modeling environment at openwfm.org, along with utilities for data preparation, visualization, and diagnostics.

2.2 Propagation Methods

The materials in this section are partially reproduced from [34].

Starting in the 1950s, considerable effort was expended exploring the effects of the mass bombings that occurred during World War II, and the collateral incendiary effects of nuclear weapons. This research effort was closely related to that of forest fires and has resulted in a boom in wildland fire behavior research.

Modeling the behavior of wildfires has became a very popular topic over the past few decades due to advances in computational power. Different propagation models have been developed and can be divided into three main categories: Physical,
Empirical and Simulation Models. All of the models have a mathematical form and are in fact mathematical models.

In this thesis, we are mainly focused on constructing a mathematical model that helps to improve the data assimilation method. Data assimilation needs to be performed whenever new data containing an updated fire perimeter arrives and is being used in a coupled fire-atmosphere model. The simulation model generally implements a pre-existing fire spread model (can be physical or empirical) in such a way as to simulate the spread of fire across a landscape.

Mathematically analogous models are usually based on mathematical concepts that have been applied to wildland fire spread. However, because these models are not delivered from any understanding of wildland fire behavior that lets this models be applied to other fields, such as epidemiology, flood studies, etc.

In general, there are two approaches for the fire representation that are implemented in simulation models: raster and vector implementation. Raster implementation deals with a fire as a group of independent cells that grow or decrease in number as the fire propagates. Vector implementation represents fire perimeter as a set of linked points in a shape of a closed curve. There are different propagation methods, whose purpose is to implement a certain extension algorithm to the fire model, such as minimal travel time, level set method, method of tracers, etc. In general, each propagation method is linked with one of the fire representation methods.

In recent years, our team was using a fire propagation method based on the level set function. In the level set function studies important results were achieved by Mallet [18] (uses high order ENO-WENO method), Mandel [19](second order upwinding) and Rochoux [20] (Flux limiter and Hamilton-Jacobi equation). Our method deals with the non-homogeneous rate of spread, which makes it more stable than the level set method.

In my model, I am applying the minimum time travel method to a raster based
fire simulation based on the Rothermel spread model. A similar approach to our method has Johnston in [13]. However, in our method we do not approximate the rate of spread, assuming that the fire shape is close to ellipse, but calculate rate of spread based on Rothermel’s formula. Unlike [13], where all the possible states of the cell and all possible resulting events are kept in memory, in our method, the state of the cell is continuously being updated to its optimal value, while all the not satisfying states are being filtered out in the earlier stage. This makes our method faster and more memory efficient.

Other methods that have a similar idea to our method is the Fast Marching Method and the method described by Finney in [11]. However, the part that makes our model different is that it continuously updates the time of ignition of the fire area using all the varying variables, such as wind, moisture and slope.

2.3 Ignition Problem and Atmospheric Balance

The current WRF-Fire model starts the fire from a given ignition point at a given time. However, usually the first data arrives when the fire has already been running for some time and the location of the ignition point is unknown and only current fire perimeter is given. If the whole fire area is set on fire simultaneously, then the atmosphere state will be incorrect and the coupled atmosphere-fire model will crash. This problem resulted in a need for the fire model to be able to start the fire from a given fire perimeter at a given time instead. Since the fuel balance and the state of the atmosphere depend on the history of the fire, the purpose of this work is to create an approximate artificial history of the fire based on the given fire perimeter and time, the fuel map, and the state of the atmosphere during the period before the perimeter time. The history is encoded as the fire arrival time at the nodes of the fire model mesh. We then use the artificial fire arrival time instead of the fire spread model to burn the fuel and generate the heat release to the atmosphere up until the fire reaches given fire perimeter. Replaying the artificial fire history enables gradual
fuel burn, instead of igniting the whole inside of the fire perimeter at once, and thus allows the fire-induced atmospheric circulation to develop. At the perimeter time, the coupled atmosphere-fire model takes over.

In [17], the fire arrival times inside a given perimeter were approximated based on the distance from a known ignition point to the perimeter, while use of the reinitialization equation was proposed in [21]. Our current approach consists of reversing the direction of time in a fire spread method and shrinking the fire to one or more ignition points. For this purpose, we have developed a new fire spread method, which is suitable for time reversal. The fire propagation methods, developed in this thesis, determine the ignition time at a node as the earliest time the fire can get to that node from the nodes that are already burning. Such methods are known as minimal travel or minimal fire arrival time [11]. Different algorithms were developed in order to suit possible demands of the user, such as implementation in parallel programming, minimization of the required amount of iterations and memory use, and use of the rate of spread as a time dependent variable. For the algorithms that deal with the homogeneous rate of spread, it was proven that the values of fire arrival times they produce are optimal. It was also shown that starting from arbitrary initial state the algorithms have finite convergence and the amount of iterations was estimated. A list of nodes on the boundary of the already burning region is maintained similarly as in the fast marching method [31]. However, the fast marching method cannot be used, since it depends on the current wind speed driving the rate of spread in the simulation, while in fire modeling the fire travel time from one node to the next changes dynamically. To build the artificial fire history, we reverse the direction of the time and proceed from the perimeter to the inside of the domain.

Replaying the fire history then establishes a reasonable fuel balance and outputs heat fluxes into the atmospheric model, which allows the atmospheric circulation to develop properly. Then the coupled atmosphere-fire model continues further fire
2.4 WRF-SFIRE Software

The background information about the WRF-SFIRE software, described in this section, is mostly reconstructed from papers [19] and [20].

The coupled WRF and SFIRE code [19] combines the Weather Research and Forecasting Model (WRF) with a semi-empirical fire spread model. It is intended to be faster than real time in order to deliver a forecast. The original source of the code was the NCAR’s CAWFE code [7, 5, 6, 8], which consists of the Clark-Hall mesoscale atmospheric model, coupled with a tracer-based fire spread model. The main advantage of the WRF model is that it is parallel supported community code routinely used for real runs [9], unlike the Clark-Hall, which is not supported, and is difficult to modify or use for real cases requiring real meteorological data, topography, and fuel maps. The model was started as WRF-Fire by [28], who proposed a combination of WRF with the tracer-based model from CAWFE. Later instead of using the existing tracer-based CAWFE code, the fire module SFIRE was developed based on the level set method [27]. A more complete description is available at [http://www.openwfm.org/wiki/OpenWFM_development_notes](http://www.openwfm.org/wiki/OpenWFM_development_notes).

The representation of the fire region by the level set function is considered to be more flexible than the representation of the burning region in CAWFE by four tracers in each cell of the fire mesh, which is one of the reasons why the fire propagation scheme was replaced. In particular, the level set function can be maintained more easily than tracers for the purpose of data assimilation.

SFIRE is a public domain software and it is distributed as WRF-Fire in the WRF source code at [http://wrf-model.org](http://wrf-model.org). The released version is updated periodically and supported by NCAR. The current version of SFIRE is available and supported at [http://openwfm.org](http://openwfm.org).

The atmospheric model operates on a 3-D grid of the Earth surface, and uses a
Figure 2.1: One $2 \times 2$ tile with the lowest layer of the atmospheric grid and the fire mesh on the surface shown. Wind vector components $u$, $v$, $w$ are located at the midpoints of the sides of the atmospheric grid cells. Reproduced from [20].

sequence of horizontally nested grids, called domains. Only the inmost atmospheric domain is coupled with the fire model. Scalar variables in the atmospheric model are located at the centers of the grid cells, while the wind vector components are given at the midpoints of the cell faces. The variables that the fire model operates on are represented by their values at the centers of the cells of the refined fire mesh (Fig. 2.1).

To fuel specifications consist of 13 [2] categories, which are predefined vectors of values of the fuel properties. The user can modify these values or specify completely new, custom fuel categories. The fire spread model operates with the average values of fuel properties.

Parallel computing, which is essential for fast execution, imposes a significant impact on user programming technique. Essentially WRF parallel infrastructure [20] divides the domain horizontally into rectangular regions, called tiles, in a way that different tiles are assigned to different processor cores, which execute in parallel. WRF code runs on a single tile, using values from strips around the tile boundary in neighboring tiles, if necessary. A disadvantage of the parallel computing structure is
that it limits the class of numerical methods that can be implemented. In particular, high-order methods, which need to update values at a node using values from distant nodes are no longer very practical, both because of the complexity of programming and because of the sharply increased communication cost. This is why numerical methods of the lowest possible order are usually more preferred.

WRF model can run in either "ideal" or "real" mode. The ideal run is mostly used for testing purposes. Particularly it is used to check the performance of the model when different fire-related capabilities are added. A WRF real run is used for forecasting and analysis of natural events.

A more complete description is available in [20].
3. Propagation on a Mesh

3.1 Existing Methods

One commonly used level set function is the signed distance from the given closed curve $\Gamma$,

$$L(x, y) = \pm \text{dist}((x, y), \Gamma),$$

(3.1)

where the sign is taken to be negative inside the region limited by $\Gamma$ and positive outside $\Gamma$, and dist stands for the Euclidean distance. Surprisingly, such function cannot be defined consistently once the problem is discretized. Consider a level set function $L$ that is given by its values on the corners of grid cells, interpolated linearly along the grid lines, and $\Gamma$ given by its intersection with the grid lines (Fig. 3.1). Then, the ratio of the values of $L$ at two neighboring mesh corners on the opposite sides of $\Gamma$ is fixed by the requirement that $L$ is linear between the two corners. In particular, it is not possible in general to define $L$ as the signed distance (3.1). For example, in Fig. 3.1, the ratio $L(X)/L(Y)$ is fixed and $L(X)$ does not depend on $Z$, while $L(Y)$ does.

One possibility is simply define the values of $L$ next to $\Gamma$ by the signed distance, and forget about the exact representation of $\Gamma$ as $L = 0$. Instead, in [25], we have proposed to find the values of $L$ next to $\Gamma$ by least squares. Denoting by $u$ the vector of the values $L$ next to $\Gamma$, it is easy to see that $u$ satisfies a homogeneous system of linear equations of the form $Bu = 0$ with at most two nonzeros per rows, and each row corresponding to an edge on the mesh that is intersected by $\Gamma$, as the edge $XY$. We can then find a suitable $u$ minimizing $\|u - d\|^2$ subject to $Bu = 0$, where $d$ are the signed distances (3.1). Once the values of $L$ near $\Gamma$ are found, one can extend $L$ to the whole domain as the distance function by the Fast Marching Method (FMM) [31], or by a simpler and less accurate approximate method suggested in [25].

A better method can be obtained by taking the spread rate into account. The
Figure 3.1: A level set function linear on the line segments connecting the nodes of the fire mesh cannot be defined at the nodes $X$ and $Y$ consistently as the signed distance (3.1) from the interface $\Gamma$. The distance of the point $X$ from $\Gamma$ does not depend on the location of the point $Z$, while the distance of $Y$ does; yet the values of the level set function at $X$ and $Y$ are linear along the segment $XY$ and so fixed by the ratio of their distances from $W$. Reproduced from [21].

level set function $L$ is a solution of the Hamilton-Jacobi equation

$$R |\nabla L| = 1, \quad L = 0 \text{ on } \Gamma.$$ 

which can be found by solving the reinitialization equation [27, Eq. (7.4)]

$$\frac{\partial L}{\partial t} = \pm (1 - R |\nabla L|)$$  \hspace{1cm} (3.2) 

where the sign is taken positive outside of $\Gamma$ and negative inside. Equation (3.2) is solved by upwinding formulas moving away from $\Gamma$ and starting from the values of $L$ on the other side of $\Gamma$. Alternating the solution process between the outside and the inside of $\Gamma$, the values of $L$ on the two sides of $\Gamma$ “balance out and a steady-state signed distance function is obtained” [27, p. 66].

The situation here is more complicated, because the spread rate $R$ depends on
the level set function $L$ following (2.4). Hence, we freeze $L$ inside $R$ and use successive approximations of the form

$$\frac{\partial L_{k+1}}{\partial t} = \pm \left( 1 - R \left( \frac{u \cdot \nabla L_k}{|\nabla L_k|} \right) \frac{\nabla L_{k+1}}{\nabla L_k} \right).$$

$L > 0$ outside of $\mathcal{P}$, $L < 0$ inside of $\mathcal{P}$, $L = 0$ on $\mathcal{P}$. \hfill (3.3)

Among the other methods that were introduced in the past few decade we should note Rehm at [29], where the second ordered method and flux limiting schemes are being used. A big variety of the level set methods and fast marching methods used for the interface advancement are described by Sethian [32].

### 3.2 Distance Based Method

This section is based on [17]. The purpose of the distance based algorithm, described later in this section, is to create artificial values of the the time of ignition on the fire model mesh, given ignition point $(x_{\text{ign}}, y_{\text{ign}})$, ignition time $T_{\text{ign}}$, fire perimeter $\Gamma$, and the time when the fire reached the perimeter $T_{\text{per}}$, assuming that the fire perimeter is convex, or at least star-shaped with respect to the ignition point. The fire perimeter is given as a set of points $(x_k, y_k)$ in the fire model domain, $k = 1, \ldots, n$ which form a closed curve consisting of line segments $[(x_k, y_k), (x_{k+1}, y_{k+1})]$ between each two successive points. We take $(x_1, y_1) = (x_{n+1}, y_{n+1})$ so that the starting and the ending point are identical. The coordinates of the point of ignition and of the points defining of the fire perimeter do not need to coincide with mesh points of the grid.

The method consists of linear interpolation of the ignition time between $T_{\text{ign}}$ at the ignition point and $T_{\text{per}}$ on the perimeter, along straight lines connecting the ignition point with points on the perimeter. The ignition time is also extrapolated beyond the perimeter in the same manner to provide a suitable level set function, as discussed in the previous section. Given a mesh point with coordinates $(x, y)$, the algorithm to determine the ignition time $T_i(x, y)$ consists of the following steps.
1. Find the intersection \((x_b, y_b)\) of the fire perimeter and the half-line starting at the ignition point and passing through the point \((x, y)\) (Fig. 3.2). For this purpose, we use the function

\[
F(x, y, x_b, y_b) = (y_b - y_{\text{ign}})(x - x_{\text{ign}}) - (x_b - x_{\text{ign}})(y - y_{\text{ign}}),
\]

which is zero if point \((x_b, y_b)\) lies on the line defined by \((x, y)\) and \((x_{\text{ign}}, y_{\text{ign}})\), and it is positive in one half-plane and negative in the other. We then find segment \([ (x_k, y_k), (x_{k+1}, y_{k+1}) ] \) such that \(F(x, y, x_k, y_k)F(x, y, x_{k+1}, y_{k+1}) < 0\) that is, the points \((x_k, y_k)\) and \((x_{k+1}, y_{k+1})\) lie on opposite sides of the line passing through \((x, y)\) and \((x_{\text{ign}}, y_{\text{ign}})\). Since the line intersects the fire perimeter at two points, one on each side of the ignition point, we choose correct segment as follows:

- If \((x, y)\) is inside \(\Gamma\) that is, closer to the ignition point than to the intersection, then the desired segment is the one that lies on the same side from the ignition point as the point \((x, y)\);

- If \((x, y)\) is outside of \(\Gamma\), then the needed segment lies on the same side from the ignition point as \((x, y)\).

2. Calculate the time of ignition of the mesh point, based on the ratio of the distances of the mesh point and the perimeter point to the ignition point,

\[
T_i(x, y) = T_{\text{ign}} + \frac{\| (x, y) - (x_{\text{ign}}, y_{\text{ign}}) \|}{\| (x_b, y_b) - (x_{\text{ign}}, y_{\text{ign}}) \|} (T_{\text{per}} - T_{\text{ign}}).
\]
3.2.1 Computational Results

We have tested this algorithm on an ideal example to measure the difference in the atmospheric winds between a simulation propagated naturally from a point and another one advanced artificially. In this example, the topography was flat except for a small hill roughly 500 m in diameter and 100 m high in the center of a domain of size $2.4 \text{ km} \times 2.4 \text{ km}$. The atmospheric and fire grid resolutions used were 60 m and 6 m respectively, with a 0.25 s time step. The background winds were approximately 9.5 m/s blowing towards southwest at the atmospheric layer 6.1 m above the surface. The first simulation was ignited from a point in the northeast corner of the domain 2 seconds from the start, and the fire perimeter was recorded after 40 minutes. This perimeter and ignition location were used to generate an artificial history for the first 40 minutes, which was replayed in the second simulation. Therefore, the fire perimeters in both simulations are identical at 40 minutes. Both simulations were then allowed to advance another 28 minutes, using the standard coupled model. The
outputs were then collected for analysis.

Any differences in the simulations after this time are a result of the error of the artificial fire propagation. In Fig. 3.2.1 we show 3D renderings of the simulation. The streamlines, that indicate the wind direction near the surface, show the updraft created as a result of the heat output due to the impact of the hill and of the fire. In Fig. 4.4.1 the fire is affecting the atmosphere despite being propagated artificially. A semi-transparent volume rendering of VAPOR (Visualization and Analysis Platform for Ocean, Atmosphere, and Solar Researchers) was added to simulate the smoke release. In Fig. 3.2.1 the differences in the wind between the two simulations at 68 minutes and the fire perimeter are shown. Fig. 3.2.1 shows the difference of the wind from the direct fire propagation minus the wind from the artificial propagation. Fig. 3.2.1 shows the relative error in the wind speed defined as the norm of the difference from Fig. 3.2.1 divided by the wind speed from the direct simulation. This shows that the maximum error at the end of this 68 minute simulation is less than 2.5%. On the figure below red and green contours represent the burning area and the area that was not set on fire yet respectively.
Figure 3.3: The difference in the U and V wind components, computed at 6.1m, of the direct simulation minus the artificial propagation at 68 minutes. Reproduced from [17].

3.2.2 Conclusion

In this preliminary investigation, the ignition times in the fire area are calculated based on the distance from the ignition point to the perimeter, assuming that the perimeter is convex or star-shaped. Simulation results for an ideal example show that the fire can continue in a natural way from the perimeter. Possible extensions include algorithms for more general perimeters and running the fire model backwards in time from the perimeter to create a more realistic history.
Figure 3.4: The relative error in the horizontal wind speed at 6.1m above the ground at 68 minutes. Reproduced from [17].

Figure 3.5: The direct fire simulation at 68 minutes. Reproduced from [17].
3.3 Forward Method

3.3.1 Simple Forward Propagation Method

For the propagation methods, described in the following chapters, the data is taken out of wrfout files, however, different sources can be used as an input data. The fire travel times $\Delta t_{AB}$ are computed from the rates of spread in the 8 directions to neighbors, stored in arrays $F_{ROS11},...,F_{ROS33}$. An extension of the algorithm, to be described later, will take into account that the rates of spread change with time, and so the fire travel times are actually functions of time. The structure of the algorithms allows a point of the mesh to have an arbitrary amount of neighbors, which makes them applicable for different meshes and connectivity. We want to find the time of ignition as the minimal travel time along the neighbors from the given fire perimeter. The goal of the algorithm is to create an array $t$ of the fire arrival times at the points in the mesh $\mathcal{M}$, from the given fire perimeter and rate of spread. The rate of spread gives the travel time ($\Delta t_{AB} > 0$) between the neighbors A and B. At each point $B$ of the mesh, $t(B)$ is the minimal fire travel distance from the neighbors. The array of the minimal travel time at every node satisfies:

$$t(B) = \min_{\forall A \in O(B)} (t(A) + \Delta t_{AB}),$$

where $O(B)$ is a set of all neighbors of the point $B$.

The area of the fire $\mathcal{F}$ is given. The boundary of $\mathcal{F}$, called a perimeter of the fire $\mathcal{P}$, consists of all the points in $\mathcal{F}$, which have at least one neighbor in $\mathcal{M} \setminus \mathcal{F}$. The values of time of ignition $t$ are given on $\mathcal{P}$

Putting $\forall A \notin O(B), \Delta t_{AB} = \infty$, we can replace (3.4) with

$$t(B) = \min_A (t(A) + \Delta t_{AB}).$$

(3.5)
Definition 3.3.1 The mesh outside of the fire perimeter defines a directed graph \( \mathcal{G}(\mathcal{N}, \mathcal{E}) \), where \( \mathcal{N} = (\mathcal{M} \setminus \mathcal{F}) \cup \mathcal{P} \), and \( \mathcal{E} \) contains the edge \( AB \) if \( \Delta t_{AB} < \infty \), where \( A \) and \( B \) are two neighboring points, \( A, B \in \mathcal{N} \) and \( \mathcal{E} \) has no looping edges (\( \Delta t_{AA} = 0 \)).

Therefore, we are looking for the minimal arrival time in the sense of the following definition.

Definition 3.3.2 For any point \( A \), the minimal arrival time \( \lambda(A) \) is defined by

\[
\lambda(A) = \min \left\{ t_0(A_0) + \sum_{i=1}^{n} \Delta t_{A_{i-1}A_i} | A_0 \in \mathcal{P}, \quad A_1, \ldots, A_{n-1} \in \mathcal{N} \setminus \mathcal{P}, \quad A_n = A, \quad 0 \leq n \leq m \right\}
\]

where \( m \) is the number of points contained in the set \( \mathcal{M} \setminus \mathcal{F} \).

Definition 3.3.3 We call point \( A \in \mathcal{N} \) reachable from \( \mathcal{P} \) if there exists a path along the edges of the graph \( \mathcal{G} \) from one of the vertices in \( \mathcal{P} \) to the point \( A \), and call it not reachable from \( \mathcal{P} \) otherwise.

Remark 3.3.4 If point \( A \) is reachable from the \( \mathcal{P} \), then there exists at least one neighbor \( B \) of \( A \), such that \( \Delta t_{BA} < \infty \). If point \( A \) is not reachable from the \( \mathcal{P} \), then one of two cases is possible

1. \( \forall B \in \mathcal{O}(A) : \Delta t_{BA} = \infty \).

2. \( A \in \mathcal{C} \), where \( \mathcal{C} \in \mathcal{F} \) is a closed curve, such that \( \forall C \in \mathcal{C} \) and \( \forall B \in \mathcal{F} \setminus \mathcal{C} : \Delta t_{BC} = \infty \).

Definition 3.3.5 Let \( \mathcal{V} \) denote a set of all vectors in \( \mathbb{R}^m \cup \{\infty\} \), where \( m \) is the number of points contained in the set \( \mathcal{N} \). Let us denote by \( t_k(B) \) the entry of the state \( t_k \) that corresponds to the point \( B \), where \( t_k \in \mathcal{V} \) is the state of the algorithm that was computed during the \( k \)-th iteration.
Definition 3.3.6 We will denote by $\mathcal{W} \in \mathcal{V}$ a set of vectors, s.t. for any $u \in \mathcal{W}$, $u(B) = \infty$ for any point $B$ that is not reachable from $\mathcal{P}$.

In the following algorithm, during each iteration, the time of ignition of all the points on the mesh is being updated from its neighbors using formula (3.5). Such construction of the algorithm is very convenient for parallel computing, since most of the computations can be performed simultaneously.

Algorithm 3.3.7 (Forward Propagation of the Fire) Given are the perimeter of the fire area $\mathcal{P}$, the area outside of the fire region $\mathcal{N}$, spread rate $\Delta t_{AB}$ for all $AB \in \mathcal{E}$, and $t_0(B) \in \mathbb{R}$ for all $B \in \mathcal{P}$. Initially $t_0(B) \in (-\infty, +\infty)$ for all $B \in \mathcal{N} \setminus \mathcal{P}$.

For $k=1,2,\ldots$

For all points $B \in \mathcal{N} \setminus \mathcal{P}$

$$t_k(B) = \min_{A \in \mathcal{O}(B)} \{t_{k-1}(A) + \Delta t_{AB}\}$$  

(3.6)

end

end

Lemma 3.3.8 Algorithm (3.3.7) defines a mapping $\alpha : \mathcal{W} \to \mathcal{W}$ by $\alpha(t_k) = t_{k+1}$.

Proof: We need to show that for $t_k \in \mathcal{W}, t_{k+1} = \alpha(t_k) \in \mathcal{W}$. Using the definition of $\mathcal{W}$ (3.3.6), it follows from remark 3.3.4 and (3.6) that $t_{k+1} \in \mathcal{W}$.

We will show that the mapping is monotonous in the partial ordering on $\mathcal{W}$

$$u \leq v \iff \forall B \in \mathcal{N} : u(B) \leq v(B)$$

(3.7)

for any $u, v \in \mathcal{W}$.

Lemma 3.3.9 For any $u, v \in \mathcal{W}$: if $u \leq v$, then $\alpha(u) \leq \alpha(v)$. 

24
Proof: Let \( u = u_k \leq v_k = v \), then by definition for all \( A \in \mathcal{N} \) \( u_k(A) \leq v_k(A) \). Since \( \alpha(u_k) = u_{k+1} \) and \( \alpha(v_k) = v_{k+1} \), then from (3.6), for all \( B \in \mathcal{N} \setminus \mathcal{P} \), \( u_{k+1}(B) \leq v_{k+1}(B) \). In addition, for all \( B \in \mathcal{P} \), \( u_{k+1}(B) = v_{k+1}(B) \) and thus \( \alpha(u) \leq \alpha(v) \).

Lemma 3.3.10 The mapping \( \alpha \) is sequentially continuous on \( \mathbb{W} \).

Proof: Suppose that for any \( A \in \mathcal{N} \), \( t_k \in \mathbb{W} \)

\[
\lim_{k \to \infty} t_k(A) = t(A),
\]

where \( t(A) \in (-\infty, +\infty] \). Let us denote \( \alpha(t_k) = u_k \), then it follows from (3.3.8) that for any \( B \in \mathcal{N} \setminus \mathcal{P} \)

\[
u_k(B) = \min_{A \in \mathcal{O}(B)} \{ t_k(A) + \Delta t_{AB} \},
\]

where \( \mathcal{O}(B) \) is a finite set. Then

\[
\lim_{k \to \infty} t_k(A) = t(A) \implies \lim_{k \to \infty} (t_k(A) + \Delta t_{AB}) = t(A) + \Delta t_{AB}.
\]

Since the minimum is taken over the finite set, then

\[
\lim_{k \to \infty} \min_{A \in \mathcal{O}(B)} \{ t_k(A) + \Delta t_{AB} \} = \min_{A \in \mathcal{O}(B)} \{ t(A) + \Delta t_{AB} \} \implies \\
\lim_{k \to \infty} u_k(B) = u(B)
\]

The next lemma shows, in particular that the fixed point of the mapping \( \alpha \) on \( \mathbb{W} \) (if it exists) is unique.

Lemma 3.3.11 If \( t = \alpha(t) \), \( t \in \mathbb{W} \), then for any \( A \) from \( \mathcal{N} \), \( t(A) \) is the minimal arrival time at the point \( A \).

Proof: For every point \( A \in \mathcal{N} \), denote \( L(A) \) the smallest length \( n \) of the path for which the minimum is achieved in the definition (3.3.2) of the arrival time of \( A \). The proof will proceed by induction on \( n \). Let \( n = 0 \), then all the points \( A \) for which
\( L(A) = 0 \) are located in \( \mathcal{P} \) and so it follows from def. 3.3.2 that \( \lambda(A) = t_0(A) \). For the case when \( n = 1 \), let \( A \in \mathcal{N} \), s.t. \( L(A) = 1 \), then

\[
\lambda(A) = \min_{A_0} \left\{ t(A_0) + \sum_{i=1}^{1} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_1 = A \right\} = \min_{A_0} \{ t(A_0) + \Delta t_{A_0A_1}, A_0 \in \mathcal{P}, A_1 = A \} = t(A).
\]

Suppose that the statement holds for all points \( A \), such that \( L(A) = k \) that is,

\[
t(A) = \lambda(A) = \min_{A_0, \ldots, A_k} \left\{ t(A_0) + \sum_{i=1}^{k} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_k = A \right\},
\]

where \( A_k \) can be only taken from the set of neighbors of \( A \), such that \( L(A_k) = k \), hence

\[
\lambda(A) = \min_{A_0, \ldots, A_k} \left\{ t(A_0) + \sum_{i=1}^{k} \Delta t_{A_{i-1}A_i} + \Delta t_{A_kA_{k+1}}, A_0 \in \mathcal{P}, A_{k+1} = A, L(A_k) = k \right\} = \min_{A_k A \in \mathcal{E}} \{ \lambda(A_k) + \Delta t_{A_kA} \} = \min_{A_k A \in \mathcal{E}} \{ t(A_k) + \Delta t_{A_kA} \} = t(A).
\]

In the theorem below we want to prove that mapping \( \alpha \) has a unique fixed point. In order to do that, let us first define the sequences \( u_k \) and \( w_k \) and describe their properties.

**Definition 3.3.12** Define the sequence \( w_k \) by

\[
w_0(B) = \begin{cases} 
    t_0(B), & \forall B \in \mathcal{P} \\
    \min \{ t_0(A) \mid A \in \mathcal{N} \setminus \mathcal{P} \}, & \forall B \in \mathcal{N} \setminus \mathcal{P} \text{ reachable from } \mathcal{P} \\
    \infty, & \forall B \in \mathcal{N} \setminus \mathcal{P} \text{ not reachable from } \mathcal{P}
\end{cases}
\]

and \( w_i = \alpha(w_{i-1}) \).
Definition 3.3.13 Define the sequence $u_k$ as

$$u_0(B) = \begin{cases} t_0(B), & \forall B \in \mathcal{P} \\ \infty, & \forall B \in \mathcal{N} \setminus \mathcal{P} \end{cases}$$

(3.9)

and $u_i = \alpha (u_{i-1})$

Lemma 3.3.14 The sequence $w_k$ is monotonically increasing.

Proof: From (3.6), for any $B \in \mathcal{N} \setminus \mathcal{P}$,

$$w_1(B) \geq w_0(B) = \min \{ t_0(A) \mid A \in \mathcal{N} \setminus \mathcal{P} \}$$

and thus $w_1 \geq w_0$. From the monotonicity of $\alpha$ (Lemma 3.3.9) we get by induction that for any $k$, $w_k \geq w_{k-1}$ and therefore sequence $w_k$ is monotonically increasing. ■

Lemma 3.3.15 The sequence $u_k$ is monotonically decreasing.

Proof: From (3.6), for any $B \in \mathcal{N} \setminus \mathcal{P}$,

$$u_1(B) \leq u_0(B) = \infty$$

and therefore $u_1 \leq u_0$. By induction, It follows from the monotonicity of $\alpha$ (Lemma 3.3.9) that for any $k$, $u_k \leq u_{k-1}$, and thus sequence $u_k$ is monotonically decreasing. ■

Theorem 3.3.16 For any $t_0 \in \mathbb{W}$, the sequence $\{t_k\}$, defined by $t_k = \alpha (t_{k-1})$, converges to the unique fixed point of the mapping $\alpha$ on $\mathbb{W}$.

Proof: First we show that the sequence $t_k$ converges to a fixed point of $\alpha$ on $\mathbb{W}$, by bounding the iterations $t_i$ by lower and upper states $w_i$ and $u_i$, defined by (3.3.12) and (3.3.13), respectively, and which will converge monotonously to fixed points $w$ and $u$ as $i \to \infty$. The uniqueness of the fixed point will conclude the proof.

Sequences $w_0$ and $u_0$ bound $t_0$ from below and above

$$w_0 \leq t_0 \leq u_0.$$
By induction and from the Lemmas\ref{lem:3.3.14} and \ref{lem:3.3.15} we get that

\[ w_{i-1} \leq w_i \leq t_i \leq u_i \leq u_{i-1}, \quad (3.10) \]

for all \( i = 1, 2, \ldots \).

If the point \( B \in \mathcal{N} \setminus \mathcal{P} \) is not reachable from \( \mathcal{P} \) then for all \( i \),

\[ w_i(B) = t_i(B) = u_i(B) = \infty. \]

In the case when the point \( B \) is reachable from \( \mathcal{P} \), then, since the sequence \( \{u_i(B)\} \) is monotonically decreasing and bounded below by \( w_0(B) \), it has a limit \( u(B) \). Consequently,

\[ \lim_{i \to \infty} u_i = u, u \in \mathbb{W} \quad (3.11) \]

and, since \( u_{i+1} = \alpha(u_i) \) and the mapping \( \alpha \) is sequentially continuous (Lemma\ref{lem:3.3.10}), it follows that \( u = \alpha (u) \).

Since \( w_{i-1} \leq w_i \) for all \( i \geq 1 \) by (3.14), for any \( B \in \mathcal{N} \setminus \mathcal{P} \), the sequence \( \{w_i(B)\} \) is nondecreasing and is bounded above by \( u \) (by (3.10) and (3.11)), therefore \( \lim_{i \to \infty} w_i(B) \) exists. Defining \( w \) by this limit, we have

\[ \lim_{i \to \infty} w_i = w, w \in \mathbb{W}. \]

Again, from \( w_{i+1} = \alpha(w_i) \) and the sequential continuity of the mapping \( \alpha \) on \( \mathbb{W} \), we have \( w = \alpha (w) \).

It follows from Lemma\ref{lem:3.3.11} that the fixed point of \( \alpha \) is unique, and thus \( u = w \). Therefore, by the squeeze theorem, \( \lim_{i \to \infty} t_i = u \).

\begin{theorem}
For any \( t_0 \in \mathbb{W} \) the sequence \( \{t_k\} \), defined by \( t_k = \alpha(t_{k-1}) \), converges in a finite number of steps.
\end{theorem}

\begin{proof}
For any \( A \in \mathcal{N} \) and for any \( i \), \( t_i(A) \geq \min \{t_0(B) \mid B \in \mathcal{N}\} \). By definition of \( \mathbb{W} \), any state \( t_0 \leq u_0 \), where \( u_0 \) is defined in (3.9). From lemma\ref{lem:3.3.18}
\end{proof}
it follows that there exists \( n \leq m \), such that \( u_n(A) < \infty \) for all \( A \) that are reachable from \( \mathcal{P} \). It follows from (3.10) that \( t_i \leq u_n, \forall i \geq n \). Therefore for any \( A \in \mathcal{N} \) and for all \( i \geq n \)

\[
  u_n(A) \geq t_i(A) \geq \min \{ t_0(B) \mid B \in \mathcal{N} \},
\]

(3.12)

where \( n \leq m \). It follows that for any point \( A \) reachable from \( \mathcal{P} \), the value \( t_k(A) \) is bounded below and above by constants and can only take a finite number of values.

Therefore for every point \( A \), reachable from \( \mathcal{P} \), the value \( t_i(A) \) reaches its minimal arrival time \( \lambda(A) \) in the finite number of steps \( i \). Since for all the points that are unreachable from \( \mathcal{P} \), their time of ignition stays equal to \( \infty \) that completes the proof.

In the case when the state \( t_0 \) is initialized so that \( t_k \) is monotonically decreasing with \( k \), the number of steps of Algorithm [3.3.7] is bounded by the number of nodes. This will be the case, in particular, if \( t_0(A) = \infty \) for all \( A \in \mathcal{N} \setminus \mathcal{P} \).

Lemma 3.3.18 Suppose that \( t_0 \in \mathbb{W}, t_k = \alpha(t_{k-1}) \) for all \( k \geq 1 \), and \( t_0 \geq t_1 \). Then, for any point \( A \in \mathcal{N} \), which can be reached from \( \mathcal{P} \) by a path with \( k \) edges,

\[
  t_k(A) = \min \left\{ t_0(A_0) + \sum_{i=1}^{n} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_n = A, n \leq k \right\}
\]

(3.13)

In particular, \( t_k \) equals to the unique fixed point of \( \alpha \) on \( \mathbb{W} \) for some \( k \leq m \).

Proof: In order to prove the theorem \( t_k \) needs to be monotonically decreasing with \( k \), which is satisfied since \( t_0 \geq t_1 \). We will show (3.13) by induction. First, for \( k = 1 \).

For any \( A \in \mathcal{N} \), which has a neighbor from \( \mathcal{P} \):

\[
  t_1(A) = \min_{A_0 \in \mathcal{O}(A)} \{ t_0(A_0) + \Delta t_{A_0A} \}^{t_0 \geq t_1} \min_{A_0 \in \mathcal{P}} \{ t_0(A_0) + \Delta t_{A_0A} \}.
\]

Suppose that the statement holds for \( k \) and now we need to prove it for \( k + 1 \). By the definition of the mapping \( \alpha \) (3.3.8):

\[
  t_{k+1}(A) = \min_{B \in \mathcal{O}(A)} \{ t_k(B) + \Delta t_{BA} \}.
\]
Substituting (3.13) for $t_k(B)$ we get

$$t_{k+1}(A) = \min_{B \in \mathcal{N} \setminus \mathcal{P}} \left\{ \begin{array}{l} t_0(A_0) + \sum_{i=1}^{n} \Delta t_{A_i-1A_i}, A_n = B, \\ A_0 \in \mathcal{P}, A_1\ldots, A_{n-1} \in \mathcal{N} \setminus \mathcal{P}, n \leq k \end{array} \right\} + \Delta t_{BA}$$

$$t_0 \geq t_1 = \min \left\{ \begin{array}{l} t_0(A_0) + \sum_{i=1}^{n+1} \Delta t_{A_i-1A_i}, \\ A_0 \in \mathcal{P}, A_1\ldots, A_n \in \mathcal{N} \setminus \mathcal{P}, A_{n+1} = A, n + 1 \leq k + 1 \end{array} \right\}.$$

### 3.3.2 Fast Forward Propagation Algorithm

The main feature of the Fast Forward Propagation Algorithm is that it only uses the points, which time of ignition was changed recently to update the values of time of ignition of their neighbors. This method is efficient in terms of speed, saving memory and decreasing the amount of iterations, however the construction of the algorithm makes it hard to implement for a parallel computing.

**Algorithm 3.3.19 (Fast Forward Propagation of the Fire)**

Given are fire area $\mathcal{F} \subset \mathcal{M}$, its perimeter $\mathcal{P} \subset \mathcal{F}$, and the ignition times $t_0(B) = T$ for all $B \in \mathcal{P}$.

1. Initialize the auxiliary set $\mathcal{D}_0 = \mathcal{P}$. For all points $A \notin \mathcal{P}$, set $t_0(A) = \infty$.

2. For $k = 1, 2, \ldots$

3. Initialize the set of points that were updated in this cycle, $\mathcal{D}_k = \emptyset$.

4. For all $B$, $f(B) := t_{k-1}(B)$.

5. For all $A \in \mathcal{D}_{k-1}$, for all neighbors $B$ of $A$,

   if $t_{k-1}(A) + \Delta t_{AB} < f(B)$ then

   $$f(B) := t_{k-1}(A) + \Delta t_{AB},$$

   $$\mathcal{D}_k = \mathcal{D}_k \cup \{B\}.$$
6. For all $B$, $t_k(B) = f(B)$.

7. If $(D_k \neq \emptyset)$
   
   Repeat from step 2
   
   else $(D_k = \emptyset)$ that is, $t$ has not changed in step 3
   
   \[ \text{EXIT} \]

**Lemma 3.3.20** The algorithm defines a mapping $\gamma$ on $\mathbb{W}$, by $\gamma(t_k) = t_{k+1}$ is monotonically decreasing and its range is in $\mathbb{W}$.

**Proof:** As it follows from the equations (3.14) and (3.15), for each entry $B$ of the state $t_k$, the value of $t_{k+1}(B)$ can only decrease or stay unchanged, which proofs the monotonic decrease of $\gamma$.

In order to show that the range of $\gamma$ is in $\mathbb{W}$, we need to prove that for all $B$ not reachable from $\mathcal{P}$, $t_k(B) = t_{k+1}(B) = \infty$, which follows from the remark 3.3.4 and (3.14, 3.15). \[ \blacksquare \]

**Theorem 3.3.21** The sequence $t_k$ defined by Algorithm 3.3.19 converges.

**Proof:** Since $\gamma$ is a monotonically decreasing mapping, it follows that for any $A \in \mathcal{N}$, $t_k(A)$ is monotonically decreasing with $k$. Since $t_k(A)$ is also bounded below by 0, the sequence converges. \[ \blacksquare \]

**Definition 3.3.22** For every iteration $k$ of the Algorithm 3.3.19 the set $\mathcal{U}_k$ is defined in such way:

1. $\mathcal{U}_0 = \mathcal{P}$;
2. If $(D_k \neq \emptyset)$ : Find a $B_k$, s.t.

   \[ t_k(B_k) = \min \{ t(B) \mid B \in \mathcal{N} \setminus \mathcal{U}_{k-1} \} \]  \hspace{1cm} (3.17)

   \[ \mathcal{U}_k = \mathcal{U}_{k-1} \cup \{ B_k \} ; \]
3. If \((D_k = \emptyset)\): \(U_k = N\).

**Definition 3.3.23** Point \(A\) is called **permanent at step** \(l\) if for any \(i > l\):

\[t_i(A) = t_i(A)\]

**Theorem 3.3.24** Algorithm 3.3.19 is finite.

**Proof:** In the following Lemma (3.3.25) we will show that \(U_l\) is a set that only consists of permanent points at the step \(l\). According to the step \(3\) of the definition 3.3.22, the set of permanent points increases at least by one point at each step. As such, the algorithm will be finite since the size of the mesh is finite.

**Lemma 3.3.25** \(\forall n \forall k > n\) for any \(B \in U_n : B \notin D_k\)

**Proof:** The proof is by induction on \(n\):

1. For \(n = 0\), initially \(U_0 = \mathcal{P}\). Since the time of ignition of the points on the perimeter is set and can not be updated during the algorithm, thus no point from \(\mathcal{P}\) can be in \(D_k\) for any \(k > 0\).

2. Suppose that the statement holds for all the cycles up to \(n - 1\). From Def. 3.3.22 after the end of the cycle \(n\),

\[B_n \in \arg \min \{t_n(A) \mid A \in \mathcal{N} \setminus U_{n-1}\}\]

and \(U_n = U_{n-1} \cup \{B_n\}\).

By the induction assumption, \(\mathcal{P}, B_1, \ldots, B_{n-1} \notin D_k\) for all \(k > n - 1\), and so, they can not be used to update the value \(t\) at the point \(B_n\) at any cycle \(k > n\).

Let \(A \in \mathcal{N} \setminus U_n = \mathcal{N} \setminus \{\mathcal{P}, B_1, \ldots, B_n\}\). We want to show that \(t_i(A) \geq t_n(B_n)\) for all \(i > n\), and, thus, \(A\) cannot be used to update point \(B_n\) in any cycle \(i > n\) either, since it will not satisfy the inequality (3.14).

(a) If \(A \notin D_i\) for all \(i > n\), then \(t_i(A) = t_n(A) \geq t_n(B_n)\) by (3.18).
(b) Assume that \( A \in D_i \) for some \( i > n \), and so using formula \( (3.15) \),
\( A \) (here \( A = A_i \)) was updated from some point \( A_{i-1} \in D_{i-1} \), which was
previously updated from some \( A_{i-2} \in D_{i-2} \), e.t.c., until some \( A_{n+1} \in D_{n+1} \)
was updated from some \( A_n \in D_n \). Then,

\[ t_i (A_i) \geq t_{i-1} (A_{i-1}) \geq \cdots \geq t_n (A_n) \geq t_n (B_n), \]

because \( t_n (B_n) \) was the minimal from \( (3.18) \). Hence, \( t_i (A) \) could not be used to update \( t \) at \( B_n \). Since \( A \) was arbitrary chosen from \( N \setminus U_n \), therefore, \( B_n \) was never updated in a cycle \( k > n \), that is, \( B_n \notin D_k \)
for all \( k > n \).

\[ \square \]

**Theorem 3.3.26** Algorithm \( 3.3.19 \) generates values of \( t \) which satisfy \( (3.5) \) for all
points in \( N \).

**Proof:** Lets assume that Algorithm \( 3.3.19 \) is finished at step \( i \), but there is a
point \( B \) and \( A \in \mathcal{O}(B) \), such that

\[ t_i (A) + \Delta t_{AB} < t_i (B) \quad (3.19) \]

Since the Algorithm is finished it follows from step 7 of Algorithm \( 3.3.19 \) and step 3
of the definition \( 3.3.22 \) that \( D_i = \emptyset \) and \( U_i = \mathcal{F} \).

Since there is a strong inequality in \( (3.19) \), then \( t_i (A) < \infty \) and thus
\( \exists k = \max \{ j : A \in D_j, j < i \} \).

By step 5 of the Algorithm and since \( \gamma \) is a monotonically decreasing mapping

\[ t_i (B) \leq t_{k+1} (B) \leq t_k (A) + \Delta t_{AB} = t_i (A) + \Delta t_{AB}, \quad (3.20) \]

which is a contrary to \( (3.19) \).

\[ \square \]
Theorem 3.3.27 Suppose that $t_0$ on $\mathcal{P}$ and $t_0(A) = \infty$ for all the points $A \in \mathcal{N} \setminus \mathcal{P}$. Then Algorithm 3.3.7 and Algorithm 3.3.19 produce the same sequence of states.

Proof: Define sequence $v_k$ by Algorithm 3.3.7, $v_k = \alpha(v_{k-1})$ and sequence $u_k$ by Algorithm 3.3.19, $u_k = \gamma(u_{k-1})$, so that $u_0 = v_0 = t_0$. In order to prove the theorem, we need to show that $v_k = u_k$ for any $k$. The proof will proceed by induction on $k$.

1. For $k = 0$ the proof follows from the definition: $u_0 = v_0$.

2. Suppose that $v_k = u_k$, that is, $\forall B \in \mathcal{N} : v_k(B) = u_k(B)$.

3. To finish the proof we need to show that $v_{k+1} = u_{k+1}$.

Assume that there exists a point $B$, such that $v_{k+1}(B) \neq u_{k+1}(B)$. According to (3.6) in Algorithm 3.3.7

$$v_{k+1}(B) = \min_{A \in \mathcal{O}(B)} \{v_k(A) + \Delta t_{AB}\} = v_k(A') + \Delta t_{A'B}, \quad (3.21)$$

for some $A' \in \mathcal{O}$. In Algorithm 3.3.19 the value of $u_{k+1}(B)$ can only be updated to its final value in (3.15) from one of the neighbors $A''$ of the point $B$, and so

$$u_{k+1}(B) = u_k(A'') + \Delta t_{A''B} = v_k(A'') + \Delta t_{A''B} \geq \min_{A \in \mathcal{O}(B)} \{v_k(A) + \Delta t_{AB}\} = v_k(A') + \Delta t_{A'B} = v_{k+1}(B)$$

Since by assumption $v_{k+1}(B) \neq u_{k+1}(B)$, then

$$u_{k+1}(B) > v_{k+1}(B). \quad (3.22)$$

1. Assume there exists an index $i \leq k$, such that $A' \in \mathcal{D}_i$. Let $j$ be the largest such index. That is, $A' \in \mathcal{D}_j$, where $j \leq k$, and $A' \notin \mathcal{D}_i$ for any $i, j < i \leq k$.

Since $j$ is the last step of the Algorithm 3.3.19 such that the value of $u$ at point $A'$ was updated, it follows that

$$u_j(A') = u_{j+1}(A') = ... = u_k(A'). \quad (3.23)$$
But then, since \( A' \in \mathcal{O}(B) \) and \( j \leq k \)
\[
\begin{align*}
u_{k+1}(B) & \leq u_{j+1}(B) \overset{(3.20)}{\leq} u_j(A') + \Delta t_{A'B} \overset{(3.14, 3.21)}{\leq} u_k(A') + \Delta t_{A'B} = v_k(A') + \Delta t_{A'B} = v_{k+1}(B),
\end{align*}
\]
which contradicts (3.22)

2. Otherwise, assume that

\( A' \notin \mathcal{D}_i \) for all \( i < k + 1 \). (3.24)

Since \( \mathcal{D}_i \) is the set of points that were updated during the time step \( i \), it follows that the value of \( u_i(A') \) was never changed for any \( i < k + 1 \). Therefore,

\[
u_k(A') = ... = u_0(A'),
\]
(3.25)

It follows from the induction assumption that

\[
v_k(A') = u_k(A') = u_0(A')
\]
(3.26)

If \( A' \in \mathcal{P} \), then, by step 1 of the Algorithm 3.3.19, \( A' \in \mathcal{D}_0 \), which contradicts (3.24).

If \( A' \notin \mathcal{P} \), then \( u_0(A') = \infty \), then

\[
\begin{align*}
u_{k+1}(B) & \overset{(3.22)}{>} v_{k+1}(B) \overset{(3.21)}{=} v_k(A') + \Delta t_{A'B} = \\overset{(3.20)}{=} u_k(A') + \Delta t_{A'B} = \overset{(3.26)}{=} u_0(A') + \Delta t_{A'B} = \infty,
\end{align*}
\]
(3.27)

which is a contradiction.

Therefore, for any \( A \) in \( \mathcal{N} \setminus \mathcal{P} \) it follows that \( v_{k+1}(A) = u_{k+1}(A) \) and that completes the proof.

\[\blacksquare\]

3.4 Backward Method

3.4.1 Simple Backward Propagation Algorithm
The goal of the backward propagation algorithm is to reconstruct an array \( t \) of the fire arrival times from its values on the perimeter \( \mathcal{P} \) of the fire area \( \mathcal{F} \), given the rate of spread of the fire at each point of \( \mathcal{F} \). For this purpose assume that \( t \) is obtained from forward propagation, starting from the ignition point that has reached its fixed point, \( t = \alpha(t) \).

Recall equation (3.5)

\[
 t(B) = \min_A \left( t(A) + \Delta t_{AB} \right).
\]

That is, the fire spreads from point \( A \) to point \( B \) during the forward propagation. It follows that

\[
 \forall C : t(B) \leq t(C) + \Delta t_{CB}
\]  

which means that the fire would arrive at the same time or later if it propagated from any other point, besides \( A \). From (3.29) we get

\[
 \forall C : t(B) - \Delta t_{CB} \leq t(C).
\]

Since \( \forall A \notin \mathcal{O}(B), \Delta t_{AB} = \infty \) we get

\[
 t(A) \geq \max_{B \in \mathcal{F}} (t(B) - \Delta t_{AB}) = \max_{B \in \mathcal{O}(A)} (t(B) - \Delta t_{AB}).
\]  

Remark 3.4.1 If \( t(A) < \max_{B \in \mathcal{F}} (t(A) - \Delta t_{AB}) \), then the fire would get to one of the neighbors of the point \( B \) too early, which would lead to the incorrect approximation of the fire arrival times.

Motivated by (3.31) we define \( t_b(B) \) as the minimum time at which the fire can arrive to the point \( B \), so that it still gets to its neighbors in time.

\[
 t_b(B) = \max_{A \in \mathcal{O}(B)} (t_b(A) - \Delta t_{AB}), \text{ where } \forall B \in \mathcal{P}, t_b(B) = t(B). \tag{3.32}
\]

Then the smallest possible fire arrival time at point \( B \), such that the fire will arrive from point \( B \) to the perimeter \( \mathcal{P} \) in time, is given in the following definition.
Definition 3.4.2 For any point $A$, the minimal arrival time $\mu(A)$ is defined by

$$\mu(A) = \max \{ t_0(A_0) - \sum_{i=1}^{n} \Delta t_{A_{i-1}A_i} | A_0 \in \mathcal{P}, A_1, \ldots, A_{n-1} \in \mathcal{N} \setminus \mathcal{P}, A_n = A, 0 \leq n \leq m_b \} ,$$

where $m_b$ is the number of points contained in the set $\mathcal{F}$.

This choice of $t(B)$ guarantees that the fire is not going to propagate to the neighbors of $B$ too early.

Definition 3.4.3 Let $\mathbb{V}_b$ denote a set of all vectors in $\mathbb{R}^m \cup \{-\infty\}$, where $m$ is the number of points contained in the set $\mathcal{F}$. Let us denote by $t_k(B)$ the entry of the state $t_k$ that corresponds to the point $B$, where $t_k \in \mathbb{V}_b$ is the state of the algorithm that was computed during the $k$-th iteration.

Definition 3.4.4 We will denote by $\mathbb{W}_b \in \mathbb{V}$ a set of vectors, s.t. for any $u \in \mathbb{W}_b$, $u(B) = -\infty$ for any point $B$ that is not reachable from $\mathcal{P}$.

In the following algorithm, during each iteration, the time of ignition of all the points on the mesh is being updated from its neighbors using formula (3.32). Such construction of the algorithm is very convenient for parallel computing, since most of the computations can be performed simultaneously.

Algorithm 3.4.5 (Backward Propagation of the Fire) Given are the perimeter $\mathcal{P}$ of the fire area $\mathcal{F} \subset \mathcal{M}$, spread rate $\Delta t_{AB}$ for all $AB \in \mathcal{E}$, and $t_0(B) \in \mathbb{R}$ for all $B \in \mathcal{P}$. Initially $t_0(B) = -\infty$ for all $B \in \mathcal{F} \setminus \mathcal{P}$.

For $k=1,2,\ldots$

For all points $B \in \mathcal{F} \setminus \mathcal{P}$

$$t_k(B) = \max_{A \in \mathcal{O}(B)} \{ t_{k-1}(A) - \Delta t_{AB} \}$$  \hspace{1cm} (3.33)

end

end
Lemma 3.4.6 Algorithm (3.4.5) defines a mapping \( \alpha_b : \mathbb{W}_b \to \mathbb{W}_b \) by \( \alpha_b(t_k) = t_{k+1} \).

**Proof:** We need to show that for \( t_k \in \mathbb{W}_b, t_{k+1} = \alpha_b(t_k) \in \mathbb{W}_b \). Using the definition of \( \mathbb{W}_b \) (3.4.4), it follows from remark 3.3.4 and (3.33) that \( t_{k+1} \in \mathbb{W}_b \) \( \blacksquare \)

We will show that the mapping is monotonous in the partial ordering on \( \mathbb{W}_b \)

\[ u \leq v \iff \forall B \in \mathcal{F} : u(B) \leq v(B) \] (3.34)

for any \( u, v \in \mathbb{W}_b \).

**Lemma 3.4.7** For any \( u, v \in \mathbb{W}_b \): if \( u \leq v \), then \( \alpha(u) \leq \alpha(v) \).

**Proof:** Let \( u = u_k \leq v_k = v \), then by definition for all \( A \in \mathcal{F} \) \( u_k(A) \leq v_k(A) \). Since \( \alpha_b(u_k) = u_{k+1} \) and \( \alpha_b(v_k) = v_{k+1} \), then from (3.33), for all \( B \in \mathcal{F} \setminus \mathcal{P} \), \( u_{k+1}(B) \leq v_{k+1}(B) \). In addition, for all \( B \in \mathcal{P} \), \( u_{k+1}(B) = v_{k+1}(B) \) and thus \( \alpha(u) \leq \alpha(v) \). \( \blacksquare \)

**Lemma 3.4.8** The mapping \( \alpha_b \) is sequentially continuous on \( \mathbb{W}_b \).

**Proof:** Suppose that for any \( A \in \mathcal{F}, t_k \in \mathbb{W}_b \)

\[ \lim_{k \to \infty} t_k(A) = t(A), \]

where \( t(A) \in [-\infty, +\infty) \). Let us denote \( \alpha_b(t_k) = u_k \), then it follows from (3.4.6) that for any \( B \in \mathcal{F} \setminus \mathcal{P} \)

\[ u_k(B) = \max_{A \in \mathcal{O}(B)} \{ t_k(A) - \Delta t_{AB} \}, \]

where \( \mathcal{O}(B) \) is a finite set. Then

\[ \lim_{k \to \infty} t_k(A) = t(A) \implies \lim_{k \to \infty} (t_k(A) - \Delta t_{AB}) = t(A) - \Delta t_{AB}. \]

Since the maximum is taken over the finite set, then

\[ \lim_{k \to \infty} \max_{A \in \mathcal{O}(B)} \{ t_k(A) - \Delta t_{AB} \} = \max_{A \in \mathcal{O}(B)} \{ t(A) + \Delta t_{AB} \} \implies \]

\[ \lim_{k \to \infty} u_k(B) = u(B) \]
Figure 3.6: (a) The situation when both forward and backward propagations give the same results. (b) Since there are no outgoing arrows from the point $C$, the value of the fire arrival time at this point will differ for forward and backward propagations.

We want to explore the relation between Algorithms 3.3.7 and 3.4.5. For this purpose, let us define the relation $\phi$ first.

**Definition 3.4.9** Assume that $(A, B) \in \phi$, then

$$t_f(B) = t_f(A) + \Delta t_{AB}, \quad (3.35)$$

where $t_f$ is a sequence that is being produced after applying Algorithm 3.3.7 to the initial sequence $t_0$.

In theorems 3.4.10 and 3.4.11 we assume that Algorithm 3.3.7 starts from the given perimeter $P_1$, which can be a single ignition point, and reaches its fixed point $\alpha(t_f) = t_f$. Pick an arbitrary fire area $F \in \mathcal{M}$ so that $P_1 \in P_2$, where $P_2$ is a contour of $F$, and use $P_2$ as a perimeter for Algorithm 3.4.5 meaning that for any $A \in P_2$, $t_b(A) = t_f(A)$. Assume that Algorithm 3.4.5 has reached its fixed point $\alpha_b(t_b) = t_b$. 

39
Theorem 3.4.10 (Algorithm comparison) For any point \( A \in \mathcal{F} \), if there exists a path \( \{B_1, \ldots, B_n\} \notin \mathcal{P}_1 \), such that

\[
(B_0, B_1) \in \phi, (B_1, B_2) \in \phi, \ldots, (B_{n-1}, B_n) \in \phi, B_0 = A, B_n \in \mathcal{P}_2,
\]

then \( t_b(A) = t_f(A) \).

**Proof:** By assumption we are given that \( t_f(B_n) = t_b(B_n) \). From (3.35) it follows that

\[
t_f(B_n) = t_f(B_{n-1}) + \Delta t_{B_{n-1}B_n} \implies t_f(B_{n-1}) = t_f(B_n) - \Delta t_{B_{n-1}B_n}, \tag{3.36}
\]

then from (3.31)

\[
t_f(B_{n-1}) \geq \max_{A \in \mathcal{O}(B_n)} (t(A) - \Delta t_{AB_n}). \tag{3.37}
\]

It follows from (3.36) and (3.37) that

\[
t_f(B_{n-1}) = t_f(B_n) - \Delta t_{B_{n-1}B_n} = \max_{A \in \mathcal{O}(B_n)} (t(A) - \Delta t_{AB_n}) \tag{3.38}
\]

and thus

\[
t_f(B_{n-1}) = t_b(B_{n-1}). \tag{3.39}
\]

Using the same logic it follows by induction on \( i \) that \( t_f(B_i) = t_b(B_i), \forall i = 0, \ldots, n \). Therefore \( t_f(A) = t_b(A) \).

Theorem 3.4.11 (General Algorithm Comparison) For any \( A \in \mathcal{F} \), if there exists a point \( B \), such that

\[
t_f(B) = t_f(A) + \Delta t_{AB}, \tag{3.40}
\]

then

\[
t_b = t_f.
\]
Proof: By definition equation 3.40 means that \((A, B) \in \phi\). Since \(F\) can have only limited amount of points then there is a path \(B_1, \ldots, B_n\)

\[(A, B_1) \in \phi, (B_1, B_2) \in \phi, \ldots, (B_{n-1}, B_n) \in \phi, B_n \in \mathcal{P}_2,\]

And thus by theorem 3.4.10 \(t_b(A) = t_f(A)\) for any \(A \in F\).

The next lemma shows, that the fixed point of the mapping \(\alpha_b\) on \(\mathbb{W}_b\) (if it exists) is unique.

Lemma 3.4.12 If \(t = \alpha_b(t), t \in \mathbb{W}_b\), then for any \(A\) from \(F\), \(t(A)\) is the maximal arrival time at the point \(A\).

Proof: For every point \(A \in F\), denote \(L(A)\) the smallest length \(n\) of the path for which the maximum is achieved in the definition 3.4.2 of the arrival time of \(A\). The proof will proceed by induction on \(n\). Let \(n = 0\), then all the points \(A\) for which \(L(A) = 0\) are located in \(\mathcal{P}\) and so it follows from def. 3.4.2 that \(\mu(A) = t_0(A)\). For the case when \(n = 1\), let \(A \in \mathcal{N}\), s.t. \(L(A) = 1\), then

\[
\mu(A) = \max_{A_0} \left\{ t(A_0) - \sum_{i=1}^1 \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_1 = A \right\} = \max_{A_0} \left\{ t(A_0) - \Delta t_{A_0A_1}, A_0 \in \mathcal{P}, A_1 = A \right\} = t(A).
\]

Suppose that the statement holds for all points \(A\), such that \(L(A) = k\) that is,

\[
t(A) = \mu(A) = \max_{A_0, \ldots, A_{k-1}} \left\{ t(A_0) - \sum_{i=1}^k \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_k = A \right\}
\]

For \(n = k + 1\) let \(A \in \mathcal{N}\), such that \(L(A) = k + 1\),

\[
\mu(A) = \max_{A_0, \ldots, A_k} \left\{ t(A_0) - \sum_{i=1}^{k+1} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_{k+1} = A \right\},
\]

where \(A_k\) can be only taken from the set of neighbors of \(A\), such that \(L(A_k) = k\), hence

\[
\mu(A) = \max_{A_0, \ldots, A_k} \left\{ t(A_0) - \sum_{i=1}^k \Delta t_{A_{i-1}A_i} - \Delta t_{A_kA_{k+1}}, A_0 \in \mathcal{P}, A_{k+1} = A, L(A_k) = k \right\}.
\]
\[ \mu(A) = \max_{A_k \in E} \{ \lambda(A_k) - \Delta t_{A_kA} \} = \max_{A_k \in E} \{ t(A_k) - \Delta t_{A_kA} \} = t(A). \]

In the theorem below we want to prove that mapping \( \alpha_b \) has a unique fixed point.

**Theorem 3.4.13** For any \( t_0 \in \mathbb{W}_b \), the sequence \( \{ t_k \} \), defined by \( t_k = \alpha_b(t_{k-1}) \), converges to the unique fixed point of the mapping \( \alpha \) on \( \mathbb{W}_b \).

**Proof:** Let us recall that

\[
t_0(B) = \begin{cases} 
-\infty \leq t_0(B) < \infty, & \forall B \in \mathcal{P} \\
-\infty, & \forall B \in \mathcal{F} \setminus \mathcal{P}
\end{cases}
\tag{3.41}
\]

and

\[ t_i = \alpha_b(t_{i-1}). \]

From (3.32), for any \( B \in \mathcal{F} \),

\[ t_1(B) \geq t_0(B) \]

and thus \( t_1 \geq t_0 \).

From the monotonicity of \( \alpha_b \) (Lemma 3.4.7), we have by induction that for all \( i = 1, 2, \ldots, t_i \geq t_{i-1} \).

If the point \( B \in \mathcal{F} \setminus \mathcal{P} \) is not reachable from \( \mathcal{P} \) then for all \( i \), then

\[ t_i(B) = -\infty. \]

In case when the point \( B \) is reachable from \( \mathcal{P} \), then, since sequence \( \{ t_i(B) \} \) is monotonically increasing and bounded above by \( \max_{A \in \mathcal{P}} \{ t_0(A) \} \), it has a limit \( t(B) \). Consequently,

\[ \lim_{i \to \infty} t_i = t \]

and, since \( t_{i+1} = \alpha_b(t_i) \) and the mapping \( \alpha_b \) is sequentially continuous, it follows that \( t = \alpha_b(t) \). It follows from Lemma (3.4.12) that the fixed point of \( \alpha_b \) is unique, which completes the proof.
In particular, the theorem below shows that for any \( t_0 \in \mathcal{W}_b \), such that \( \forall A \in \mathcal{F} \setminus \mathcal{P} \), \( t_0(A) = -\infty \), the sequence \( \{ t_k \} \), defined by \( t_k = \alpha_b(t_{k-1}) \), converges in a finite number of steps \( k \), where \( k \leq m_b \).

**Theorem 3.4.14** Suppose that \( t_0 \in \mathcal{W}_b \), \( t_k = \alpha_b(t_{k-1}) \) for all \( k \geq 1 \), and for all \( A \in \mathcal{F} \setminus \mathcal{P} \), \( t_0(A) = -\infty \). Then, for any point \( A \in \mathcal{F} \), which can be reached from \( P \) by a path with \( k \) edges, \( t_k(A) \), equals to the unique fixed point of \( \alpha_b \) on \( \mathcal{W}_b \) for some \( k \leq m_b \).

**Proof:** We will show (3.42) by induction.

Let \( k = 1 \), then for any \( A \in \mathcal{F} \) that can be reached from \( P \) in one step

\[
t_1(A) = \max_{A_0 \in \mathcal{O}(A)} \{ t_0(A_0) \} = \max_{A_0 \in \mathcal{P}} \{ t_0(A_0) \}.
\]

Suppose (3.42) holds for \( k \) and so we need prove it for \( k + 1 \). By definition of \( \alpha_b \),

\[
t_{k+1}(A) = \max_{B \in \mathcal{O}(A)} \{ t_k(B) - \Delta t_{BA} \}.
\]

Substituting (3.42) for \( t_k(B) \) we get

\[
t_{k+1}(A) = \max_{B \in \mathcal{F} \setminus \mathcal{P}} \left\{ \max_{A \in \mathcal{O}(A)} \left\{ t_0(A_0) - \sum_{i=1}^{n} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_1, \ldots, A_{n-1} \in \mathcal{F} \setminus \mathcal{P}, A_n = B, n \leq k \right\} \right\} - \Delta t_{BA}
\]

\[
= \max_{A \in \mathcal{P}, A_1, \ldots, A_{n+1} \in \mathcal{F} \setminus \mathcal{P}, A_{n+1} = A, n+1 \leq k+1} \left\{ t_0(A_0) - \sum_{i=1}^{n+1} \Delta t_{A_{i-1}A_i}, A_0 \in \mathcal{P}, A_1, \ldots, A_n \in \mathcal{F} \setminus \mathcal{P} \right\}.
\]

\[\square\]

### 3.4.2 Fast Backward Propagation Algorithm
The main feature of the Fast Backward Propagation Algorithm is that it only uses the points, which time of ignition was changed recently to update the values of time of ignition of their neighbors. This method is efficient in terms of speed, saving memory and decreasing the amount of iterations, however the construction of the algorithm makes it hard to implement for a parallel computing.

Algorithm 3.4.15 (Fast Backward Propagation of the Fire) Given are fire area $\mathcal{F} \subset \mathcal{M}$, its perimeter $\mathcal{P} \subset \mathcal{F}$, and the ignition times $t_0(B) = T$ for all $B \in \mathcal{P}$.

1. Initialize the auxiliary set $D_0 = \mathcal{P}$. For all points $A \notin \mathcal{P}$, set $t_0(A) = -\infty$.

2. For $k = 1, 2, ...$

3. Initialize the set of points that were updated in this cycle, $D_k = \emptyset$.

4. For all $B$, $f_b(B) := t_{k-1}(B)$.

5. For all $A \in D_{k-1}$,

   for all neighbors $B$ of $A$,

   
   
   
   if $t_{k-1}(A) - \Delta t_{AB} > f_b(B)$ then
   
   
   
   $f_b(B) := t_{k-1}(A) - \Delta t_{AB}$,  \hfill (3.43)

   

   

   $D_k = D_k \cup \{B\}$.  \hfill (3.44)

6. For all $B$, $t_k(B) = g_b(B)$.

7. If ($D_k \neq \emptyset$)

   Repeat from step 2

   else ($D_k = \emptyset$) that is, $t$ has not changed in step 3

   EXIT
Lemma 3.4.16 The algorithm defines a mapping $\gamma_b$ on $\mathbb{W}_b$, by $\gamma_b(t_k) = t_{k+1}$ is monotonically increasing and its range is in $\mathbb{W}_b$.

Proof: As it follows from the equations (3.43) and (3.44), for each entry $B$ of the state $t_k$, the value of $t_{k+1}(B)$ can only increase or stay unchanged, which proofs the monotonic increase of $\gamma_b$.

In order to show that the range of $\gamma_b$ is in $\mathbb{W}_b$, we need to prove that for all $B$ not reachable from $\mathcal{P}$, $t_k(B) = t_{k+1}(B) = -\infty$, which follows from the remark 3.3.4 and (3.43 3.44).

Theorem 3.4.17 The sequence $t_k$ defined by Algorithm 3.4.15 converges.

Proof: Since $\gamma_b$ is a monotonically increasing mapping, it follows that for any $A \in \mathcal{F}$, $t_k(A)$ is monotonically increasing with $k$. Since $t_k(A)$ is also bounded above by $\max_{B \in \mathcal{P}} \{t_0(B)\}$, the sequence converges.

Definition 3.4.18 For every iteration $k$ of the Algorithm 3.4.15 the set $U_{b,k}$ is defined in such way:

1. $U_{b,0} = \mathcal{P}$;
2. If $(\mathcal{D}_k \neq \emptyset)$ : Find a $B_k$, s.t. 
   $$ t_k(B_k) = \max \{t(B) \mid B \in \mathcal{N} \setminus U_{b,k-1}\} $$  (3.46) 
   $$ U_{b,k} = U_{b,k-1} \cup \{B_k\}; $$
3. If $(\mathcal{D}_k = \emptyset)$: $U_{b,k} = \mathcal{N}$.

Theorem 3.4.19 Algorithm 3.4.15 is finite.

Proof: In the following Lemma (3.4.20) we will show that $U_l$ is a set that only consists of permanent points (3.3.23) at the step $l$. According to the step 3 of the definition 3.4.18, the set of permanent points increases at least by one point at each step. As such, the algorithm will be finite since the size of the mesh is finite.
Lemma 3.4.20 \( \forall n \\forall k > n \) for any \( B \in U_{b,n} : B \notin D_k \)

Proof: The proof is by induction on \( n \):

1. For \( n = 0 \), initially \( U_{b,0} = P \). Since the time of ignition of the points on the perimeter is set and can not be updated during the algorithm, thus no point \( B \) from \( P \) can be in \( D_k \) for any \( k > 0 \).

2. Suppose that the statement holds for all the cycles up to \( n - 1 \). From definition \text{3.4.18}, after the end of cycle \( n \),

\[
B_n \in \arg\max \{ t_n(A) \mid A \in F \setminus U_{b,n-1} \}, \tag{3.47}
\]

and \( U_{b,n} = U_{b,n-1} \cup \{ B_n \} \).

By the induction assumption, \( P, B_1, \ldots, B_{n-1} \notin D_k \) for all \( k > n - 1 \), and so, they can not be used to update the value \( t \) at the point \( B_n \) at any cycle \( k > n \).

Let \( A \in F \setminus U_{b,n} = F \setminus \{ P, B_1, \ldots, B_n \} \). We want to show that \( t_i(A) \leq t_n(B_n) \) for all \( i > n \), and, thus, \( A \) can not be used to update point \( B_n \) in any cycle \( i > n \) either, since it will not satisfy the inequality \text{(3.43)}.

(a) If \( A \notin D_i \) for all \( i > n \) that is, the point \( A \) was never updated, and

\[
t_i(A) = t_n(A) \leq t_n(B_n) \text{ by (3.47),}
\]

(b) Assume that \( A \in D_i \) for some \( i > n \), and so using formula \text{(3.44)},

\( A \) (here \( A = A_i \)) was previously updated from some point \( A_{i-1} \in D_{i-1} \),

which was updated from some \( A_{i-2} \in D_{i-2} \), e.t.c., until some \( A_{n+1} \in D_{n+1} \) was updated from some \( A_n \in D_n \). Then,

\[
t_i(A_i) \leq t_{i-1}(A_{i-1}) \leq \cdots \leq t_n(A_n) \leq t_n(B_n),
\]

because \( t_n(B_n) \) was the maximal from \text{(3.47)}.

Hence, \( t_i(A) \) could not be used to update \( t \) at \( B_n \). Since \( A \) was arbitrary from \( F \setminus U_{b,n} \), therefore, \( B_n \) was never updated in a cycle \( k > n \) that is, \( B_n \notin D_k \) for all \( k > n \).
Theorem 3.4.21 Algorithm \[3.4.15\] generates values of \(t\) which satisfy (3.32) for all points in \(\mathcal{N}\).

**Proof:** Lets assume that Algorithm \[3.4.15\] is finished at step \(i\), but there is a point \(B\) and \(A \in \mathcal{O}(B)\), such that

\[
t_i(A) - \Delta t_{AB} > t_i(B)
\]  \hspace{1cm} (3.48)

Since the Algorithm is finished it follows from step 4 of Algorithm \[3.4.15\] and step 3 of the definition \[3.4.18\] that \(\mathcal{D}_i = \emptyset\) and \(\mathcal{U}_i = \mathcal{F}\).

Since there is a strong inequality in (3.48), then \(t_i(A) > -\infty\) and thus \(\exists j = \min \{k : A \in D_k, k < i\}\).

By step 5 of the algorithm \[3.4.15\] and since \(\gamma_b\) is a monotonically increasing mapping

\[
t_i(B) \geq t_{j+1}(B) \geq t_j(A) - \Delta t_{AB} = t_i(A) - \Delta t_{AB},
\]  \hspace{1cm} (3.49)

which is a contrary to (3.48).

Theorem 3.4.22 Suppose that \(t_0\) on \(\mathcal{P}\) and \(t_0(A) = -\infty\) for all points \(A \in \mathcal{F}\setminus\mathcal{P}\). Then Algorithm \[3.4.5\] and Algorithm \[3.4.15\] produce the same sequence of states.

**Proof:** Define sequence \(v_k\) by Algorithm \[3.4.5\], \(v_k = \alpha_b(v_{k-1})\) and sequence \(u_k\) by Algorithm \[3.4.15\], \(u_k = \gamma_b(u_{k-1})\), so that \(u_0 = v_0 = t_0\). In order to prove the theorem, we need to show that \(v_k = u_k\) for any \(k\). The proof will proceed by induction on \(k\).

1. For \(k = 0\) the proof follows from the definition: \(u_0 = v_0\).
2. Suppose that \(v_k = u_k\), that is, \(\forall B \in \mathcal{F} : v_k(B) = u_k(B)\).
3. To finish the proof we need to show that \(v_{k+1} = u_{k+1}\).
Assume that there exists a point $B$, such that $v_{k+1}(B) \neq u_{k+1}(B)$. According to (3.33) in Algorithm 3.4.5,

$$v_{k+1}(B) = \max_{A \in \mathcal{O}(B)} \{ v_k(A) - \Delta t_{AB} \} = v_k(A') - \Delta t_{A'B},$$

(3.50)

for some $A' \in \mathcal{O}$. In Algorithm 3.4.15, the value of $u_{k+1}(B)$ can only be updated to its final value in (3.44) from one of the neighbors $A''$ of the point $B$, and so

$$u_{k+1}(B) = u_k(A'') - \Delta t_{A''B} = v_k(A'') - \Delta t_{A''B} \leq \max_{A \in \mathcal{O}(B)} \{ v_k(A) - \Delta t_{AB} \} = v_k(A') - \Delta t_{A'B} = v_{k+1}(B)$$

Since by assumption $v_{k+1}(B) \neq u_{k+1}(B)$, then

$$u_{k+1}(B) < v_{k+1}(B).$$

(3.51)

1. Assume there exists an index $i \leq k$, such that $A' \in \mathcal{D}_i$. Let $j$ be the largest such index. That is, $A' \in \mathcal{D}_j$, where $j \leq k$, and $A' \notin \mathcal{D}_i$ for any $i$, $j < i \leq k$. Since $j$ is the last step of the Algorithm 3.4.15 such that the value of $u$ at point $A'$ was updated, it follows that

$$u_j(A') = u_{j+1}(A') = \ldots = u_k(A').$$

(3.52)

But then, since $A' \in \mathcal{O}(B)$ and $j \leq k$

$$u_{k+1}(B) \geq u_{j+1}(B) \geq u_j(A') - \Delta t_{A'B} = \ldots = u_k(A') - \Delta t_{A'B} = v_k(A') - \Delta t_{A'B} = v_{k+1}(B),$$

(3.53)

which contradicts (3.51)

2. Otherwise, assume that

$$A' \notin \mathcal{D}_i$$

for all $i < k + 1$.

(3.53)
Since $D_i$ is the set of points that were updated during the time step $i$, it follows that the value of $u_i(A')$ was never changed for any $i < k + 1$. Therefore,

$$u_k(A') = ... = u_0(A'), \quad (3.54)$$

It follows from the induction assumption that

$$v_k(A') = u_k(A') = u_0(A') \quad (3.55)$$

If $A' \in P$, then, by step 1 of the Algorithm 3.4.15 $A' \in D_0$, which contradicts (3.53).

If $A' \not\in P$, then $u_0(A') = \infty$, then

$$u_{k+1}(B) \quad (3.51) v_{k+1}(B) \quad (3.50) v_k(A') - \Delta t_{A'B} =$$

$$= u_k(A') - \Delta t_{A'B} \quad (3.55) u_0(A') + \Delta t_{A'B} = -\infty, \quad (3.57)$$

which is a contradiction.

Therefore, for any $A$ in $F \setminus P$ it follows that $v_{k+1}(A) = u_{k+1}(A)$ and that completes the proof.  

\qed
3.5 Time Dependent Method

3.5.1 Backward Propagation

The advantage of the method described in this section is that it uses the rate of spread of the fire as a time dependent variable. This makes this method applicable in the real world problems, where such variables as wind and moisture are changing continuously.

In the figure 3.7, it is shown how the fire propagates until time step $t_k$. The fire has spread from point $B$ to point $C$ on the way to point $A$ and covered the distance $d$. The distance $d(A, B)$ that the fire covers on the way from point $B$ to point $A$ at the time $t$, given the rate of spread of the fire $ROS(A, B, t)$ can be described by the differential equation

$$\frac{\partial d(A, B)}{\partial t} = ROS(A, B, t) \tag{3.58}$$

In order to solve the differential equation, we are using Euler’s discretizing method. According to figure 3.7, equation (3.58) can be written in the integral form

$$\int_{t(A)}^{t(B)} ROS(A, B, \tau) d\tau = d(A, B). \tag{3.59}$$

In other words the problem that equation (3.59) is solving is to find how much time $t(B)$ it takes for the fire to cover the distance $d(A, B)$ between points $A$ and $B$ and ignite point $B$, given that point $A$ was ignited at time $t(A)$.

**Algorithm 3.5.1 (Time-dependent Backward Propagation of the Fire)** Given are fire area $\mathcal{F} \subset \mathcal{M}$, its perimeter $\mathcal{P} \subset \mathcal{F}$, and the ignition times $t_0(B) = T$ for all $B \in \mathcal{P}$. For any $A, B \in \mathcal{E}$ and time step $k$ the distance between points $A$ and $B$ $\text{Dist}(A, B)$ and the Rate of Spread $RoS_k(A, B) \text{ (m/sec)}$ is given, where the time step length is $TS$ seconds.
Figure 3.7: Fire line at step $t_k$, where the fire covered distance $d$ on the way from point $B$ to point $A$. $d' = ROS(t, C, \vec{BA})$.

1. Initialize the auxiliary set $D_0 = P$. For all points $A \notin P$, set $t_0(A) = -\infty$. For any points $A$ and $B$, s.t. $AB \in \mathcal{E}$, initialize the time that it takes for the fire to travel from point $A$ to $B$ as $\tau(A, B) = 0$.

2. For $k = 1, 2, ...$

3. Initialize the set of points that would be updated in the end of the current cycle, $D_k = \emptyset$ and the time that would be left for the fire to propagate from point $A$ to point $B$ during the current time step, $t_s(A, B) = TS$.

4. For all $B$, $u(B) := t_{k-1}(B)$.

5. For all $A \in D_{k-1}$ (here set $D_{k-1}$ might still get updated dynamically, where new points are added in the end of the set)

   for all $B \in \mathcal{O}(A)$,

   \[
   \text{If} \quad \frac{\text{dist}(A, B)}{\text{Ros}_i(A, B)} > t_s(A, B)
   \]
then (case when it would take more time steps for the fire to travel from point $A$ to point $B$)

$$\tau(A, B) = \tau(A, B) + ts(A, B),$$
$$Dist(A, B) = Dist(A, B) - RoS_i(A, B) * ts(A, B),$$
$$\mathcal{D}_k = \mathcal{D}_k \cup A,$$

else (case when the fire reaches point $B$ during the current time step)

$$\tau(A, B) = \tau(A, B) + \frac{Dist(A, B)}{RoS_i(A, B)}$$

If $u(B) < u(A) - \tau(A, B)$

$$u(B) := u(A) - \tau(A, B)$$

end

$$\forall C \in \mathcal{O}(B), \ ts(B, C) = ts(B, C) - \frac{dist(A, B)}{RoS_i(A, B)}.$$

$$\mathcal{D}_{k-1} = \mathcal{D}_{k-1} \cup B$$

end (end of step 5)

end (end of step 4)

$$\mathcal{D}_{k-1} = \mathcal{D}_{k-1} \setminus \{A\}$$

end (end of step 2)

6. If $(\mathcal{D}_{k-1} \neq \emptyset)$

Repeat from step 5

7. For all $B$, $t_k(B) = u(B)$.

8. If $(\mathcal{D}_k \neq \emptyset)$
Repeat from step 2.

else $(\mathcal{D}_k = \emptyset)$ that is, for any $A$, $t(A)$ can not change anymore.

**EXIT**

### 3.5.2 Forward Propagation

**Algorithm 3.5.2 (Time-dependent Backward Propagation of the Fire)** Given are fire area $\mathcal{F} \subset \mathcal{M}$, its perimeter $\mathcal{P} \subset \mathcal{F}$, and the ignition times $t_0(B) = T$ for all $B \in \mathcal{P}$. For any $A, B \in \mathcal{E}$ and time step $k$ the distance between points $A$ and $B$ $\text{Dist}(A, B)$ and the Rate of Spread $\text{RoS}_k(A, B)$ (m/sec) is given, where the time step length is $TS$ seconds.

1. Initialize the auxiliary set $\mathcal{D}_0 = \mathcal{P}$. For all points $A \notin \mathcal{P}$, set $t_0(A) = -\infty$. For any points $A$ and $B$, s.t. $AB \in \mathcal{E}$, initialize the time that it takes for the fire to travel from point $A$ to $B$ as $\tau(A, B) = 0$.

2. For $k = 1, 2, ...$

3. Initialize the set of points that would be updated in the end of the current cycle, $\mathcal{D}_k = \emptyset$ and the time that would be left for the fire to propagate from point $A$ to point $B$ during the current time step, $ts(A, B) = TS$.

4. For all $B$, $u(B) := t_{k-1}(B)$.

5. For all $A \in \mathcal{D}_{k-1}$ (here set $\mathcal{D}_{k-1}$ might still get updated dynamically, where new points are added in the end of the set)

   for all $B \in \mathcal{O}(A)$,

   If

   $$\frac{\text{dist}(A, B)}{\text{RoS}_k(A, B)} > ts(A, B)$$
then (case when it would take more time steps for the fire to travel from point A to point B)

\[ \tau(A, B) = \tau(A, B) + ts(A, B), \]
\[ Dist(A, B) = Dist(A, B) + RoS_i(A, B) \times ts(A, B), \]
\[ D_k = D_k \cup A, \]

else (case when the fire reaches point B during the current time step)

\[ \tau(A, B) = \tau(A, B) + \frac{Dist(A, B)}{RoS_i(A, B)}, \]
\[ If \ u(B) > u(A) + \tau(A, B) \]
\[ u(B) := u(A) + \tau(A, B) \]

end

\[ \forall C \in O(B), \ ts(B, C) = ts(B, C) - \frac{dist(A, B)}{RoS_i(A, B)}. \]
\[ D_{k-1} = D_{k-1} \cup B \]

end (end of step 5)

end (end of step 4)

\[ D_{k-1} = D_{k-1} \setminus \{A\} \]

end (end of step 2)

6. If \( (D_{k-1} \neq \emptyset) \)

Repeat from step 5

7. For all \( B, \ t_k(B) = u(B). \)

8. If \( (D_k \neq \emptyset) \)
Repeat from step 2.

else $(D_k = \emptyset)$ that is, for any $A$, $t(A)$ can not change anymore.

EXIT
4. Application to Fires: Perimeter Ignition

4.1 Initialization from a Fire Perimeter

The description of the WRF-SFIRE model is from [17], [18], the perimeter ignition is a further development of the ideas in [14] and some of the results were already published in [16].

In this chapter we use a scheme for creating an artificial fire history, that will be used in the fire replay scheme, and test it on a real case simulation of the two fires, Witch and Guejito, that have happened in Santa Ana in 2007. Given a fire perimeter, Algorithm 3.5.2 is used to approximate ignition times, represented as $t_{ign}(i, j)$, and which are located at the mesh points. The approximation is established by simulated propagation based on their fire propagation speed and ignition times of the neighboring mesh points.

A fire perimeter, represented as $\mathcal{P}$, is a closed curve composed of linear segments. It is given as a sequence of the coordinates at the endpoints of the segments. In practice, such geospatial data are often provided as a GIS shapefile, or encoded in a KML file, e.g., from GEOMAC.

The ignition times at locations outside of the given fire contour can be thought of as predictions of future ignition times in those locations as the fire continues to burn.

In the real case, the wind and the speed of propagation always change with time, and so they will be treated as variables instead of constants. For this reason, the Fast Marching method, which requires the propagation speed constant in time, cannot be used.

4.2 Creating Artificial Ignition Time History from the Current Fire Perimeter

A typical fire model starts a fire simulation from a known ignition point at a known ignition time. However, users are also interested in starting WRF-SFIRE from an existing fire, which presence has been detected and mapped. Under these circumstances, the ignition point and ignition time typically become known too late to be
relevant for real-time simulation and forecasting. Thus, we are interested in starting a fire simulation from a given fire perimeter at a given time, from now on, called the perimeter time. However, the fuel balance and the state of the atmosphere depend on the history of the fire, which is not known.

Our solution is to create an approximate artificial history of the fire based on the given fire perimeter and the perimeter time, the fuel map, and the state of the atmosphere during the period before the perimeter time. The history is encoded as the fire arrival time at the nodes of the fire model mesh. We then use the artificial fire arrival time instead of the fire spread model to burn the fuel and generate the heat release to the atmosphere. Replaying the artificial fire history enables gradual fuel burn, instead of igniting the whole inside of the fire perimeter at once, and thus allows the fire-induced atmospheric circulation to develop. At the perimeter time, the complete coupled atmosphere-fire model takes over.

4.3 Encoding and Replaying the Fire History

The state of the fire model consists of a level set function, \( \Phi \), given by its values on the nodes of the fire model mesh, and time of ignition \( T_i \). The level set function is interpolated linearly. At a given simulation time \( t \), the fire area is the set of all points \( (x, y) \) where \( \Phi(t, x, y) \leq 0 \). The level set function and the ignition time satisfy the consistency condition

\[
\Phi(t, x, y) \leq 0 \iff T_i(x, y) \leq t, \tag{4.1}
\]

as both of these inequalities express the condition that the location \( (x, y) \) is burning at the time \( t \). In every time step of the simulation, the level set function is advanced by one step of a Runge-Kutta scheme for the level set equation

\[
\frac{d\Phi}{dt} = -R \| \nabla \Phi \|,
\]

where \( R = R(t, x, y) \) is the fire rate of spread, which depends on the fuel, wind speed, and slope. The ignition time at nodes is then computed for all newly ignited nodes, and it satisfies the consistency condition \( (4.1) \).
The fire history is encoded as an array of ignition times $T_i(x, y)$, prescribed at all fire mesh nodes. To replay the fire in the period $0 \leq t \leq T_{\text{per}}$, the numerical scheme for advancing $\Phi$ and $T_i$ is suspended, and instead the level set function is set to
\[
\Phi(t, x, y) = T_i(x, y) - t.
\]
After the end of the replay is reached, the numerical scheme of the level set method is started from the level set function $\Phi$ at $t = T_{\text{per}}$.

For reasons of numerical accuracy and stability, the level set function needs to have approximately uniform slope. For example, a very good level set function, which has slope equal to one, is the signed distance from a given closed curve $\Gamma$,
\[
\Phi(x, y) = \pm \text{dist}
\]

where the sign is taken to be negative inside the region limited by $\Gamma$, and positive outside [27]. Thus, the ignition times $T_i$ need to be given on the whole domain and they need to be such that $T_i$ decreases with the distance from the given perimeter inside the fire region, and increases outside. The ignition times $T_i$ outside of the given fire perimeter are perhaps best thought of as what the ignition times might be in future as the fire keeps burning.

4.4 Results

4.4.1 Santa Ana fires

This subsection was already published as a part of [16]. These were two fires, Witch fire and then later Guejito fire, which merged quickly into one massive fire. The perimeter from the simulation on October 22, 2007, 1:00 PM PDT is in Fig. 4.2 and the artificial fire arrival time, created based on the precomputed fire progression, is shown in Fig. 4.3. The artificial fire arrival time graph has two minima, which correspond to the two ignition points and times. For the Witch fire, the error in the ignition point location was 3.28 km, which is 5.7% of the diameter of the given fire perimeter, and the ignition time was exactly the same. For the Guejito fire, the error
in the location of the ignition point was 0.04 km, which is 0.07% of the diameter of the perimeter, and the error in the ignition time was 0.26 h, which is 2.4% of the time from the ignition to the perimeter time. The RMSE of the artificial fire arrival time up to the perimeter time compared to the original simulation was 1,199 s. Scaling by the time 24 h 15 m = 81,100 s from the first ignition, October 21, 2007 12:15 PM PDT to the fire perimeter time October 22, 2007, 1:00 PM, gives the relative RMSE of the artificial fire arrival time only 1.5%. Figure 4.4 shows a comparison of the wind from the original simulation and from the spin-up using the artificial fire arrival time. We have then continued the simulation for additional 8 h to assess the effect of the perimeter ignition on further propagation of the fire (Fig. 4.5). Again, the original simulation, and the simulation started from the fire perimeter ignition, are quite close, demonstrating the utility of the present approach to simulating the progression of already developed fires detected as perimeters, rather than ignition points. The RMSE of the fire arrival time after continuing from the artificial fire arrival time for 8 h was 706 s that is, 0.25%. This is the error of the perimeter ignition, which was entirely caused by the slight change in the wind at the perimeter time due to the use of the artificial fire arrival time inside the given perimeter.
Figure 4.2: Perimeter of the 2007 Santa Ana fires simulation October 22, 2007 1PM PDT [16]. The fire consisted of two fires, Witch and Guejito, which started on October 21, 2007 12:15 PM PDT and 22 October 2007 1:00 AM PDT, respectively, and subsequently merged. The asterisks and circles from left to right indicate the real and artificially calculated ignition points of Guejito and Witch fires respectively. Reproduced from [16].
Figure 4.3: Artificial fire arrival time found by fire propagation back in time from the fire perimeter in [4.2]. The two peaks on the bottom, marked by arrows, are the two ignition locations and times, found automatically from the perimeter. The vertical axis and the false color are the time from the beginning of the simulation. Reproduced from [16].
Figure 4.4: (a) Horizontal wind at 6.1 m in the 2007 Santa Ana fires simulation on October 2007 1:00PM PDT. (b) The same wind as in (a), but with the artificial ignition time history from Fig. 4.3 until October 2007 1:00PM PDT. (c) The difference of (a) and (b). The RMSE is 1.1 $ms^{-1}$, which is 8.8% of the maximal wind speed 12.53 $ms^{-1}$. Reproduced from [16].
4.4.2 Ideal Case. Algorithm Driven by Pure Atmospheric Data with no Fire Involved

The simulation of the Santa Ana fires described in the previous section utilized previously simulated wind prior to the design of Extended Fast Backward algorithm (Algorithm 3.5.2). Such approach uses the rate of spread computed beforehand, which is based on the known ignition points and the fire-atmosphere feedback, which modifies the winds, which in turn affect the spread rate. However, in reality, the ignition point to start the simulation from is not known yet. In this section, we study how the algorithm performs using winds that have not been perturbed by the presence of fire.

The fire perimeter is obtained from a coupled atmosphere-fire simulation, and it is used to initialize the Extended Fast Backward algorithm, which goes back in time
from the perimeter. The rate of spread is computed from winds unperturbed by the fire, using an identical simulation where the fire was not ignited.

In Figure 4.6(a), the fire starts from two ignition lines originating at 1860m East, 1860m North, and the ignition continues in opposite directions. The first ignition point corresponds to \( i = 31, j = 31 \) on the atmospheric mesh, and \( i_f = 310, j_f = 310 \) on the fire grid. Figure 4.6(b) presents the fire perimeter, measured after 30 minutes of fire driven simulation. Figures 4.7(a) and 4.7(b) show the artificial time of ignition, created after applying the extended backtracking algorithm to the fire perimeter measured at 30 minutes of the no fire and fire driven simulation respectively, with the resulting ignition points located at \( (i_f = 312, j_f = 307) \) and \( (i_f = 315, j_f = 307) \). The time of ignition history of the original fire simulation with the ignition at \( (i_f = 310, j_f = 310) \) is shown in Figure 4.8(a). The difference between fire arrival times calculated using the no fire and original simulation is shown in Figure 4.8(b). Figure 4.9 shows the graphical comparison of the location of the real ignition point, the one found with perturbed winds, and the ignition point with winds without fire in relation to the contour of the perimeter.

Overall, the results of using the fire rate of spread data, not affected by the fire, in Extended Fast Backward propagation algorithm for the ideal case are close to the results based on the fire driven data.
Figure 4.6: (a) The ignition point of the fire driven simulation. (b) Perimeter of the fire, taken after the fire driven simulation was running for 30 minutes. The false color represents the amount of fuel burnt. Problem setup and visualization done by Adam Kochanski.

Figure 4.7: Artificial fire arrival time found by the backward fire propagation algorithm which originated from the fire perimeter in 4.6(b) and was using no fire driven rate of spread in (a) and fire driven rate of spread in (b). The vertical axis and the false color are the time from the beginning of the simulation.
Figure 4.8: Fire arrival time of the original simulation. (b) Difference between fire arrival times of the original simulation and the simulation with no fire affect.

Figure 4.9: Location of the real ignition point, the one found with perturbed winds, and the ignition point with winds without fire in relation to the contour of the perimeter.
5. Fast Fourier Transform Ensemble Kalman Filter with Application to a Coupled Atmosphere-Wildland Fire Model

In this chapter, based on [22], we propose a new type of the Ensemble Kalman Filter (EnKF), which uses the Fast Fourier Transform (FFT) for covariance estimation from a very small ensemble with automatic tapering, and for a fast computation of the analysis ensemble by convolution, avoiding the need to solve a sparse system with the tapered matrix. The method is combined with the morphing EnKF to enable the correction of position errors, in addition to amplitude errors, and demonstrated on WRF-Fire, the Weather Research Forecasting (WRF) model coupled with a fire spread model implemented by the level set method. My personal contribution was visualization of the computational results and participating in the discussions of the theoretical base of the chapter.

Data assimilation is a statistical technique used to modify the state of a running model in response to data, based on sequential Bayesian estimation [15]. The EnKF [10] accesses the model only as a black box, which makes it suitable for a wide range of problems. However, the ensemble size needed can be large, and the amount of computation required in EnKF can be significant when the EnKF is modified to suppress spurious long-range correlations. We propose a new variant of EnKF that can overcome these difficulties by the use of FFT. The new method is applied to the morphing EnKF [3]. My personal contribution was visualization of the computational results and participating in the discussions of the theoretical base of the chapter.

5.1 Data Assimilation

We first provide a background information about data assimilation.

The EnKF advances in time an ensemble of simulations $u_1, \ldots, u_N$, which approximates the probability distribution of the model state $u$. The simulations are then advanced in time until analysis time, when new data $d$ arrives. It is assumed that the data error is known, $d - Hu \sim N(0, R)$ given $u$, where $H$ is observation operator and
$R$ is the *data error covariance matrix*. The ensemble, now called *forecast ensemble*, is combined with the data to give the *analysis ensemble* by the EnKF formulas \[4\]

$$u^a_k = u_k + C_N H^T \left( HC_N H^T + R \right)^{-1} \left( d + e_k - H u^f_k \right),$$  \hspace{1em} (5.1)

where $C_N$ is an estimate of the covariance of the model state, and $e_k$ is random perturbation $e_k \sim N(0, R)$. The analysis ensemble $u^a_1, \ldots, u^a_N$ is then used as the initial condition for a new simulation, advanced to a new analysis time, and the *analysis cycle* repeats.

In the standard EnKF \[4\], $C_N$ is the sample covariance computed from the ensemble. It can be proved, under certain assumptions, that the ensemble converges for large $N$ to a sample from the Kalman filtering distribution and $C_N$ converges to the true covariance \[24\].

### 5.1.1 FFT EnKF and Covariance Estimation by FFT

Starting from this section, the chapter contains mostly new material.

We are interested in obtaining a reasonable approximation of the covariance matrix from a very small sample. Therefore, we take advantage of the fact that the simulation state $u$ is a block vector, where the blocks are values of the modelled physical fields on grids of points in a spatial domain $\Omega$, which are (discrete versions of) smooth random functions, i.e., realizations of random fields. But for a small sample, the ensemble covariance is a matrix of low rank with large off-diagonal elements even at a great distance. Therefore, localization techniques are used, such as tapering, which consists of multiplication of the terms of the sample covariance by a fixed function to force the drop-off of the covariance away from the diagonal, resulting in a more accurate approximation of covariance for small samples \[12\]. However, solving a system with the resulting approximate covariance matrix is expensive, because efficient dense linear algebra, which relies on the representation of the sample covariance matrix as the product of two rectangular matrices \[19\], can no longer be used.
For simplicity, we explain the FFT EnKF in the 1D case. Higher-dimensional cases work exactly the same. Consider first the case when the model state consists of one block only.

The basic operation in the EnKF (5.1) is the multiplication $v = C_N u$ of a vector $u$ by an approximate covariance matrix $C_N$. Denote by $u(x_i)$ the entry of vector $u$, corresponding to node $x_i$ and suppose for the moment that the random field is stationary that is, its covariance matrix satisfies $C(x_i, x_j) = c(x_i - x_j)$ for some covariance function $c$. Then $v$ is the convolution

$$v(x_i) = \sum_j C(x_i, x_j) u(x_j) = \sum_j u(x_j) c(x_i - x_j).$$

The discrete Fourier transform changes convolution to multiplication entry by entry. Hence, if the random field is stationary, multiplication by its covariance matrix becomes multiplication by a diagonal matrix in the frequency domain.

The proposed method of covariance estimation consists of computing the sample covariance of the ensemble in the frequency domain, and neglecting all of its off-diagonal terms. This is justified by the assumption that the covariance depends mainly on distance.

1. Given ensemble $[u_k]$, apply a FFT operator $F$ to each member to obtain the member $\hat{u}_k = Fu_k$ in the frequency domain.

2. Compute the approximate forecast covariance matrix $\hat{C}_N$ in the frequency domain as the diagonal matrix with the diagonal entries $\hat{c}_i$ equal to the diagonal entries of the sample covariance of the ensemble $[\hat{u}_k]$,

$$\hat{c}_i = \frac{1}{N-1} \sum_{k=1}^N |\hat{u}_{ik} - \bar{u}_i|^2, \quad \bar{u}_i = \frac{1}{N} \sum_{k=1}^N \hat{u}_{ik}. \quad (5.2)$$

Multiplication by the approximate covariance matrix $C_N$ then becomes in the frequency domain

$$u = C_N v \iff \hat{u} = Fu, \quad \hat{v} = \hat{c} \cdot \hat{u}, \quad v = F^{-1} \hat{v},$$
where $\bullet$ is entry-by-entry multiplication, $(\hat{c} \bullet \hat{u})_i = \hat{c}_i \hat{u}_i$. In the important case $H = I$ and $R = rI$ (the whole state is observed and the data errors are uncorrelated and the same at every point), considered here, the EnKF (5.1) in the frequency domain reduces to entry-by-entry operations,

$$\hat{u}^a_k = \hat{u}_k + \hat{c} \bullet (\hat{c} + r)^{-1} \bullet \left( \hat{d} + \hat{e}_k - \hat{u}^f_k \right). \quad (5.3)$$

In an application, the state has multiple variables. The state vector, its covariance, and the observation matrix then have the block form

$$u = \begin{bmatrix} u^{(1)} \\ \vdots \\ u^{(n)} \end{bmatrix}, \quad C = \begin{bmatrix} C^{(11)} & \cdots & C^{(1M)} \\ \vdots & \ddots & \vdots \\ C^{(M1)} & \cdots & C^{(MM)} \end{bmatrix}, \quad H = \begin{bmatrix} H^{(1)} & \cdots & H^{(M)} \end{bmatrix}. \quad (5.4)$$

Assume that the first variable is observed, then $H^{(1)} = I$, $H^{(2)} = 0, \ldots, H^{(M)} = 0$. The EnKF (5.1) then simplifies to

$$u^{(j),a}_k = u^{(j)}_k + C_N^{(j1)} \left( C_N^{(11)} + R \right)^{-1} \left( d + e_k - u^{(1)}_k \right), \quad j = 1, \ldots, M, \quad (5.5)$$

which becomes in the frequency domain

$$\hat{u}^{(j),a}_k = \hat{u}_k + \hat{c}^{(j1)} \bullet (\hat{c}^{(11)} + r)^{-1} \bullet \left( \hat{d} + \hat{e}_k - \hat{u}^f_k \right), \quad (5.6)$$

where the spectral cross-covariance between field $j$ and field 1 is approximated from

$$\hat{c}^{(j1)}_i = \frac{1}{N-1} \sum_{k=1}^{N} \left( \frac{\hat{c}^{(j)}_i}{\hat{u}_{ik}} - \frac{\hat{c}^{(j1)}_i}{\hat{u}_i} \right) * \left( \frac{\hat{c}^{(1)}_i}{\hat{u}_{ik}} - \frac{\hat{c}^{(1)}_i}{\hat{u}_i} \right), \quad \frac{\hat{c}^{(j)}_i}{\hat{u}_i} = \frac{1}{N} \sum_{k=1}^{N} \frac{\hat{c}^{(j)}_i}{\hat{u}_{ik}}. \quad (5.7)$$
5.1.2 Morphing EnKF

To treat position errors in addition to amplitude errors, FFT EnKF is combined with the morphing EnKF [3, 19]. The method uses an additional ensemble member \( u_{N+1} \), called the reference member. Given an initial state \( u \), the initial ensemble is given by \( u_{N+1} = u \) and

\[
    u_k^{(i)} = \left( u_{N+1}^{(i)} + r_k^{(i)} \right) \circ (I + T_k), \quad k = 1, \ldots, N, \quad (5.8)
\]

where \( r_k^{(i)} \) are random smooth functions on \( \Omega \), \( T_k \) are random smooth mappings \( T_k : \Omega \rightarrow \Omega \), and \( \circ \) denotes composition. Thus, the initial ensemble has both amplitude and position variability, and the position change is the same for all blocks. Random smooth functions and mappings are generated by FFT as a Fourier series with random coefficients that decay quickly with frequency.

The data \( d \) is an observation of \( u^{(1)} \), and it is expected that it differs from the model in amplitude as well as in the position of significant features, such as firelines. The first blocks of \( u_1, \ldots, u_N \) and \( d \) are then registered against the first block of the reference member \( u_{N+1} \). We find registration mappings \( T_k : \Omega \rightarrow \Omega, \ k = 0, \ldots, N \) such that

\[
    u_k^{(1)} \approx u_{N+1}^{(1)} \circ (I + T_k), \quad T_k \approx 0, \quad \nabla T_k \approx 0, \quad k = 0, \ldots, N,
\]

where \( d = u_0^{(1)} \). Define the registration residuals \( r_k^{(j)} = u_k^{(j)} \circ (I + T_k)^{-1} - u_{N+1}^{(j)}, \ k = 0, \ldots, N \). The morphing transform maps each ensemble member \( u_k \) into the extended statevector

\[
    u_k \mapsto \tilde{u}_k = M_{u_{N+1}}(u_k) = \left( T_k, r_k^{(1)}, \ldots, r_k^{(M)} \right). \quad (5.9)
\]

Similarly, the data becomes the extended data vector \( d \mapsto \tilde{d} = \left( T_0, r_0^{(1)} \right) \). The FFT EnKF method (5.6) is applied to the transformed ensemble \( \tilde{u}_1, \ldots, \tilde{u}_N \) with the
observation operator given by \((T, r^{(1)}, \ldots, r^{(M)}) \mapsto (T, r^{(1)})\). The cross-covariances between \(x\) and \(y\) components of \(T\) and \(r^{(1)}\) are neglected, so the covariance \(C^{(11)}\) in (5.5) consists of three diagonal matrices, and (5.6) applies. The new transformed reference member is obtained as

\[
\tilde{u}_{N+1}^a = \frac{1}{N} \sum_{k=1}^{N} \tilde{u}_k^a
\]  

(5.10)

and the analysis ensemble \(\tilde{u}_1, \ldots, \tilde{u}_{N+1}\) by the inverse morphing transform

\[
u_k^{a,(i)} = M_{a_{N+1}}^{-1}(\tilde{u}_k^a) = \left(u_{N+1}^{(i)} + r_k^{a,(i)}\right) \circ (I + T_k^a), \quad k = 1, \ldots, N + 1.
\]  

(5.11)

5.2 Computational Results

We have used the optimization method from Ref. [3] for registration. We have used the real sine FFT, which forces zero change on the boundary. We have used a standard ideal problem distributed with WRF-Fire. The model has a 420 × 420 fire mesh and a 42 × 42 × 41 atmospheric mesh. The fuel was uniform on the whole domain. The model was initialized with the wind blowing diagonally across the grid, and two line ignitions and one circle ignition occur within the first 4 seconds of simulation time. After one minute of simulation time, when the fire was established and one of the line fires has merged with the circular fire, the simulation was stopped and an initial ensemble was generated by random smooth perturbation both in position and in amplitude. Artificial data was created by a similar perturbation. The forecast was taken the same as the initial ensemble. The described data assimilation algorithm was then applied with 5 members, with the results shown in Figure 5.1 for the morphing EnKF and Figure 5.2 for the morphing FFT EnKF. We see that the EnKF was not able to approach that data at all with such a small ensemble, while the FFT EnKF delivered an ensemble around the correct data shape.

5.3 Conclusion

We have shown that the morphing FFT EnKF is capable of data assimilation in a wildfire simulation, which exhibits sharp boundaries and coherent features. We have
shown that the FFT EnKF can deliver acceptable results with a very small ensemble (5 members), unlike the standard EnKF, which is known to work with morphing for this application, but only with a much larger ensemble[3].
Figure 5.1: The morphing Kalman Filter (EnKF) with 5 ensemble members, applied to the ground heat flux from the WRF-Fire model. The ensemble size is not sufficient, the correct analysis is not even approximately in the span of the forecast, and the EnKF cannot reach it.
Figure 5.2: The morphing FFT EnKF with 5 ensemble members, applied to the ground heat flux from the WRF-Fire model. The analysis ensemble moved towards the data. Reproduced from [22].
6. Computation of the Fuel Fraction

This chapter is based on the research done in [20]. My personal contribution to the chapter was developing the code that implements the calculation of the fuel fraction and embedding the code into the WRF model code. The section below summarizes the background material.

Initially, fire simulation area is partitioned into a grid, where each cell starts with fuel fraction $F = 1$. In this chapter we are particularly investigating the case when the cell is partially burning. Once the fuel is ignited at a time $t_i$, the fuel fraction decreases exponentially,

$$F(t) = \exp \left( -\frac{(t - t_i)}{T_f} \right), \quad t > t_i,$$

(6.1)

where $t$ is the time, $t_i$ is the ignition time, $F_0$ is the initial amount of fuel, and $T_f$ is the number of seconds for the fuel to burn down to $1/e \approx 0.3689$ of the original quantity. The fuel weight $w$ is given by the user in the input data as one of the coefficients in the fuel categories. The default values are taken from the CAWFE code, which, according to [6, p. 55], were chosen to approximate the mass-loss curve from the BURNUP algorithm [1]. In this chapter, the speed of burning is taken to be independent of the wind speed and the fuel moisture.

The average sensible heat flux density released in time interval $(t, t + \Delta t)$ is computed as

$$\phi_h = \frac{F(t) - F(t + \Delta t)}{\Delta t} \frac{1}{1 + M_f} w_h, \quad (W m^{-2})$$

(6.2)

and the average latent heat (i.e., moisture) flux density is given by

$$\phi_q = \frac{F(t) - F(t + \Delta t)}{\Delta t} \frac{M_f + M_f}{1 + M_f} L w_h, \quad (W m^{-2})$$

(6.3)

where $M_f$ is the estimated mass ratio of the water output from the combustion to the dry fuel, and $L$ is the specific latent heat of condensation of water at $0 \, ^\circ C$, used for nominal conversion of moisture to heat. This computation is from CAWFE.
It should be noted that there is significant uncertainty in the data as well as in the approximations made above, and many factors that influence the spread rate are not accounted for.

6.1 Computation of the Fuel Fraction.

This section contains new material.

The fuel fraction is approximated over each fire mesh cell $C$ by integrating \((6.1)\) over the fire region. Hence, the fuel fraction remaining in cell $C$ at time $t$ is given by

$$F = 1 - \frac{1}{\text{area}(C)} \iint_{x \in C, \psi(x) \leq 0} 1 - \exp \left( -\frac{t - t_i(x)}{T_f(x)} \right) \, dx.$$  \hspace{1cm} (6.4)

Once the fuel fraction is known, the heat fluxes are computed from \((6.2)\) and \((6.3)\). This scheme has the advantage that the total heat released in the atmosphere over time is exact, regardless of approximations in the computation of the integral \((6.4)\). Our objective in the numerical evaluation of \((6.4)\) is a method that is second order accurate when the whole cell is on fire, exact when no part of the cell $C$ is on fire, and provides a transition between these two cases.

While the fuel burn time $T_f$ can be interpolated as constant over the whole cell, the level set function $\psi$ and the ignition time $t_i$ must be interpolated more accurately to allow a submesh representation of the burning area and a gradual release of the heat as the fireline moves over the cell. We will also need the fuel fraction computed over each mesh cell, because the heat fluxes in the mesh cells are summed up to give the heat flux in an atmospheric cell. Our solution is to split each cell into 4 subcells $C_j$, interpolate to the corners of the subcells, and add the integrals,

$$\iint_{x \in C, \psi(x) \leq 0} 1 - \exp \left( -\frac{t - t_i(x)}{T_f(x)} \right) \, dx = \sum_{j=1}^{4} \iint_{x \in C_j, \psi(x) \leq 0} 1 - \exp \left( -\frac{t - t_i(x)}{T_f(x)} \right) \, dx,$$  \hspace{1cm} (6.5)

cf., Fig. 6.1 Here $T_f$ is constant on each subcell $C_j$, given by its value at the fire grid nodes.
When the whole cell $C$ is on fire (that is, $\psi \leq 0$ on all four vertices of $C$), $t_i$ is interpolated also linearly to the vertices of the subcells $C_j$. However, the case when the fireline crosses the cell $C$ requires a special treatment of the ignition time $t_i$; $t_i(x)$ has meaningful value only when the node $x$ is on fire, $\psi(x) \leq 0$. Also $\psi(x) = 0$ and $t_i(x) = t$ on the fireline. Thus, approximating both $\psi$ and $t_i$ in the fire region by linear functions suggests interpolating from the relation

$$t_i - t = c\psi,$$  \hfill (6.6)

for some $c$. We interpolate on the grid lines between two nodes first. If both nodes are on fire, we interpolate $t_i$ bilinearly as before. However, when one cell center is on fire and one not, say $\psi(a_1) > 0$, $\psi(a_2) < 0$, we find the proportionality constant $c$ in (6.6) from $t_i(a_2) = c\psi(a_2)$, and set $t_i(b) = c\psi(b)$ at the midpoint $b = (a_1 + a_2)/2$. In the case of interpolation to the node $c = (a_1 + a_2 + a_3 + a_4)/4$ between nodes $a_1, a_2, a_3, a_4$, we find the proportionality constant $c$ by solving the least squares problem

$$\sum_{j=1}^{4} |t_i(a_j) - t - c\psi(a_j)|^2 \to \min$$

and set again $t_i(c) = c\psi(c)$. 

78
Figure 6.1: Division of fire mesh cells into subcells for fuel fraction computation. The level set function $\psi$ and the ignition time $t_i$ are given at the centers $a_1, \ldots, a_4$ of the cells of the fire grid. The integral \((6.5)\) over the cell $C$ with the center $a_3$ is computed as the sum of integrals over the subcells $C_1, \ldots, C_4$. While the values of $\psi$ and $t_i$ are known at $a_3 = x_3$, they need to be interpolated to the remaining corners $x_1, x_2, x_4$ of the subcell $C_1$ from their values at the points $a_1, \ldots, a_4$. Reproduced from [20].
In order to compute the integral over a subcell $C_j$, we first need to estimate the fraction of the subcell that is burning, by

$$\frac{\text{area} \{ \mathbf{x} \in C_j : \psi (\mathbf{x}) \leq 0 \}}{\text{area}(C_j)} \approx \beta = \frac{1}{2} \left( 1 - \frac{\sum_{k=1}^{4} \psi (\mathbf{x}_k)}{\sum_{k=1}^{4} \mid \psi (\mathbf{x}_k) \mid} \right), \quad (6.7)$$

where $\mathbf{x}_k$ are the corners of the subcell $C_j$.

Next, replace $t_i(\mathbf{x}_k)$ by $t$ when $\psi (\mathbf{x}_k) > 0$ (i.e., the node $\mathbf{x}_k$ is not on fire), and compute the approximate fraction of the fuel burned as

$$\frac{1}{\text{area} (C)} \iint_{\mathbf{x} \in C \atop \psi (\mathbf{x},t) \leq 0} 1 - \exp \left( -\frac{t - t_i(\mathbf{x})}{T_{fi}(\mathbf{x})} \right) d\mathbf{x} \approx \beta \left( 1 - \exp \left( -\frac{1}{4} \sum_{k=1}^{4} \frac{t_i(\mathbf{x}_k) - t}{T_{fi}} \right) \right) \quad (6.8)$$

6.2 Conclusion

In this chapter we implement the method of computing the fuel fraction of the cell that is partially burning. This fuel fraction calculation $6.8$ is accurate asymptotically when the fuel burns slowly and the approximation $\beta$ of the burning area is exact. The estimation $6.7$ of the area of the subcell that is burning, needed for $6.8$, is exact when no part of the subcell $C_j$, is on fire that is, all $\psi (\mathbf{x}_k) \geq 0$ and at least one $\psi (\mathbf{x}_k) > 0$; the whole $C_j$ is on fire that is, all $\psi (\mathbf{x}_k) \leq 0$ and at least one $\psi (\mathbf{x}_k) < 0$; the values $\psi (\mathbf{x}_k)$ define a linear function and the fireline crosses the subcell diagonally or when it is aligned with one of the coordinate directions.
REFERENCES


