NTNU
Norwegian University of Science and Technology

A mathematical model for calculating river hydrographs using high resolution digital elevation models

Anders Opskar Voldsund

Master of Science in Physics and Mathematics

Submission date:	June 2017
Supervisor:	Knut Andreas Lie, IMF
Co-supervisor:	André Brodtkorb, Sintef
	Odd Andersen, Sintef

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Prediction of floods is important to prevent damage to human lives, buildings or infrastructure, and we have developed a simulation model to address this. The model is a so-called distributed rainfall-runoff model, which accounts for spatial variations within the watershed; it is based on a distributed version of the time-area method (Clark, 1945). Travel times are calculated by assuming that the flow can be modelled as creeping flow, and the resulting velocity field is used to solve the so-called time-of-flight/Eikonal equation. Both topographical and heterogeneous properties can be accounted for by the model, which ideally can be used for both gauged and ungauged watersheds after calibration. An automatic watershed delineation algorithm has been implemented to delineate the river's watershed using the D8 Algorithm (O'Callaghan and Mark, 1984). To test our model, we create artificial rainfalls, and calculate the hydrograph response at the watershed outlet. The model works well given the simplifications, and is a framework which can be expanded upon and made more complex.

Samandrag

Gode flaumvarslingar er viktig for å forhindre skader på menneskeliv, bygningar og infrastruktur, og vi har utvikla ein simuleringsmodell som adresserer dette. Modellen er ein såkalla distribuert nedbør/avløpsmodell som tek høgde for romlege endringar i nedbørsområdet, og den baserer seg på ein distribuert versjon av tid-areal metoden (Clark, 1945). Reisetider er rekna ut ved å anta at flyten kan modellerast som 'creeping flow', og det resulterande hastigheitsfeltet blir brukt til å løyse den såkalla 'time-of-flight'/Eikonal-likninga. Både topografiske og heterogene eigenskapar kan bli tatt høgde for av modellen, som ideelt sett kan bli nytta i nedbørsområder med og utan målestasjonar, etter at modellen er kalibrert. Ei algoritme for å automatisk rekne ut nedbørsområder har blitt implementert ved bruk av D8-algoritma (O'Callaghan and Mark, 1984). For å teste modellen har vi konstruert kunstige nedbørsbyger, og rekna ut den resulterande vassføringa i elveutløpet. Modellen fungerer bra, gitt forenklingane, og er eit rammeverk som kan bli tilført kompleksitet om ynskja.

Preface

The thesis before you has been written to fulfill the requirements for the Master of Science in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU), Trondheim. The research was conducted for the Computational Geosciences group at SINTEF, from January to June 2017.

I would like to thank my supervisors, Knut-Andreas Lie, Odd Andersen, and André Rigland Brodtkorb, for their neverending patience and willingness to help. Their unique perspectives made the thesis more complete, and their guidance and support have been greatly appreciated.

During my time here at SINTEF, I have had numerous discussions with colleagues, all of whom have been happy to offer insight and suggestions. It cannot be emphasized enough how much I have enjoyed my time here, and I would be hard pressed to find a more knowledgeable, friendly, and welcoming group of people. The fascinating and entertaining lunch discussions, as well as the intense chess matches, will be sorely missed. Lastly, I want to thank my family and friends for always being there for me, and keeping me motivated until the very end.

Anders Opskar Voldsund

Oslo, June 7, 2017

Contents

1 Introduction

Norway is a country with a varied topography and climate, praised for its majestic nature. The weather at the west coast and in the north can be both unpredictable and intense. During the last hundred years there have been several floods with severe consequences, in particular east in Norway. Frequent storms and floods call for accurate weather forecasts, and the ability to predict these events. The advent of supercomputers and remote satellite data have improved this tremendously.

Norwegian climate scientists estimate an increase in precipitation in the range of 5 % to 30 % for Norway by year 2100[1]. The number of extreme precipitation events is also expected to rise. A potential consequence of this is more frequent floods, which threatens human lives and cause damage to infrastructure and buildings. Floods are hard to predict, in particular if they are caused by a local rainfall. A perfect example of this is the convective summer rainfalls in Norway, which bring large amounts of water in a short time. Cities are particularly vulnerable because of the many impermeable surfaces, and infrastructure incapable of handling large water masses in a short time.

In order to investigate flood risk for a river, we need to delineate its watershed, i.e., locate the region it collects its water from. One common method is the New Hampshire Method [1], which relies on a map and a pen, but this is obsolete. The increase in computational power and availability of elevation data, have paved the way for automatic algorithms; an essential component of these is the choice of flow direction algorithm. The most popular one is perhaps the D8 (deterministic eight) Algorithm [26], which was used by Jenson and Domingue with excellent results [16], even though D8 has some limitations [10]. Many other algorithms have also been developed, like the Global Search Algorithm (GD8) [28], and the Aspect-Driven Kinematic Routing Algorithm [19]; both of these, and also the D8 algorithm, are single flow direction (SFD) algorithms. This is advantageous for watershed delineation, as no cell belongs to different watersheds simultaneously.

We build upon the implementations and findings in [39] in this Master's thesis, and have expanded the algorithms and software implementations with new functionality. It uses the D8 Algorithm, and largely follows the steps of Jenson and Domingue [16] to make the landscape depressionless. Unlike Jenson and Domingue, we refrain from setting flow directions in flat areas, as part of an optimization. After the landscape is made depressionless, we construct a cell connectivity matrix, which tells us the upslope and downslope neighbors of each cell. This gives us the flexibility to delineate the watershed of any location. The connectivity matrix also

[1]According to a summary of NorClim's findings in the article 'Mer regn i framtiden' at forskning.no. NorClim was funded by the Research Council of Norway, and was a climate research project that lasted from 2007 to 2010. Findings of the study were used in The Fifth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC) [27].

allows us to calculate accumulated flow in the landscape.

A so-called rainfall-runoff model can be used to predict a river's discharge based on the forecasted rainfall in the delineated watershed. These models have traditionally been used by engineers for flood forecasting and design flood estimations [23]. Two of the most common model types are the lumped rainfall-runoff models, and the distributed rainfall-runoff models [24]. The lumped model considers the watershed as a single homogeneous unit [18], in which all inputs, outputs and parameter values are averaged over the entire watershed [24]. This is fundamentally different from the distributed type, which accounts for spatial variations within the watershed.

Both lumped models and distributed models need to be calibrated before they can yield reliable runoff estimates [37]. Lumped models usually have far fewer (4-20) model parameters compared to distributed models (10 to 1000's), which makes lumped models more attractive. In reality, distributed models might have fewer than this, because of the spatial correlation for parameters since cells close to each other typically have the same soil type. Some people prefer distributed models because of the stronger link between physical properties and model parameters. In theory, the distributed models should outperform the lumped ones, but in practice this is not always the case [18], and the model should be selected based on the availability of data and its intended application. A drawback about most rainfall-runoff models is their inability to provide good runoff estimates for ungauged watersheds. When no historical input-output data is available for calibration, other methods must be applied, such as regionalization and regional calibration [20].

Distributed rainfall-runoff models are expected to increase in popularity, as some of its downsides have been remedied by the advent of supercomputers, and increased access to remote satellite data [12]. In this thesis we will implement a distributed model based on a version of the time-area method [7]. This has traditionally been a lumped model, but if we allow non-uniform precipitation and account for spatial variations within the watershed [34], it can become a distributed one. Usually the time-area method divides the watershed into smaller areas, but one statement says: 'it is conceivable that these areas are cells in a raster data set' [5]; this is exactly what we will do in our implementation, in which we consider cells of size 10 by 10 meters.

An integral part of our rainfall-runoff model is the estimation of speeds in the watershed. The combination of the segmental approach and Manning's Equation can be used to approximate the speeds, in a method which takes some terrain effects into account [5]. We have chosen another approach to estimate our velocity field. By assuming that the flow can be modeled as a creeping flow, we can use a simplified version of Darcy's Law for single-phase flow [21, p. 16] in porous media. Both the topography and the local heterogeneous properties can be included in the velocity field. Based on this we calculate the so-called time-of-flight [21, p. 129] equation to obtain the travel times to the watershed outlet. This allows us to create a hydrograph to visualize the runoff in the river.

In this thesis we have developed a framework that can account for both topographical and heterogeneous properties in the watersheds. Calibrating the model is outside the scope of this thesis, but our vision is that once the model has been calibrated, it does not have to be re-calibrated for every new watershed it is applied to. Hence, it should also work for ungauged watersheds. Because the heterogeneous properties are a part of the calibration process, these have not been included, which means the travel time is only dependent on topography.

Finally, we test our model using synthetic precipitation forecasts where we vary the rainfall's shape, speed, direction, and intensity. A hydrograph for the river is then created based on the runoff from the watershed. If the river has a maximum discharge level before it floods, the hydrograph can tell us if the level will be surpassed in the course of the rainfall (and the time after). In our model we do not account for storage in the terrain, and let all precipitation become runoff.

Outline

In Chapter 2 we start with some hydrological concepts that are necessary to have a basic understanding of before we go into details about our automatic watershed delineation algorithm, and the rainfall-runoff model. This will include information on the hydrologic cycle, what a watershed is and how we can obtain it, more about different rainfall-runoff models and how they work, and the river hydrograph. In Chapter 3, we outline our algorithm for automatic delineation of watersheds from digital elevation models, and use examples to explain the process. Some of the sections in Chapters 2 and 3 are based on the work in the specialization project [39] at NTNU, but have been modified and expanded. Chapter 4 outlines the algorithm to estimate travel times for the delineated watershed, which in turn are used in Chapter 5 to estimate the hydrograph for the river. In Chapter 6 we test our different algorithms on a high resolution digital elevation model. Appendices A and B are based on the specialization project, and have been revised. All implementations can be found at Github [38].

2 Important concepts in hydrology

In this chapter we will define some concepts that are useful to know about if you are new to hydrology. We mentioned in Chapter 1 that we want to estimate the discharge for a forecasted rainfall in a watershed, and how a rainfall-runoff model can accomplish this. Simply put, the rainfall-runoff model calculates the runoff that is generated from a rainfall, and the resulting discharge is illustrated in a hydrograph. If a river can only handle a certain discharge before it floods, the hydrograph will tell us whether this threshold will be surpassed. In this chapter, we will talk about more about these concepts, and we will start with the watershed.

2.1 The watershed

There are many synonyms for watersheds in use, and you may have encountered the words catchment, river basin or drainage basin before. These words all mean the same, but in this work we will use the term watershed. So what is a watershed? Rumynin [33] states that a watershed is a topographic region in which all water drains to a common outlet. This means that if a bucket of water is poured out at any location within this region, the water will flow to the same outlet. Margulis [23] phrases it slightly different: A watershed is the set of all upstream points that will ultimately route water to a defined outlet point. Technically, the two definitions are the same, as a set of points does not have to be discrete, but the wording of Margulis is easily applicable to our discrete setting as we view the watershed as discrete data points in our algorithms.

It is most common to define watersheds from outlets placed in large rivers, because large rivers potentially indicate large watersheds. If we want to investigate flooding, it is most sensible to study a location in the river that is particularly vulnerable. This can for example be a point in the river where the river is narrower or the river banks are lower.

Every watershed has a river network that leads the water to the outlet, and it has the structure of a tree. The watershed outlet is located in the biggest branch, where the river has the largest flow. It carries as much water as it does because it is the result of many smaller rivers, streams, creeks and rivulets. When we talk about a river that merges with a larger river, we call it a *tributary river*, and the tributary river's watershed is a sub-watershed of the largest river's watershed.

We will now take a look at how increased computational power and availability of elevation data have changed how we obtain watersheds in the last few decades. In Chapter 3 we will also outline an algorithm to obtain our own watersheds.

The process of determining the watershed of a river outlet in the landscape is called delineating a watershed. One common method for delineating a watershed

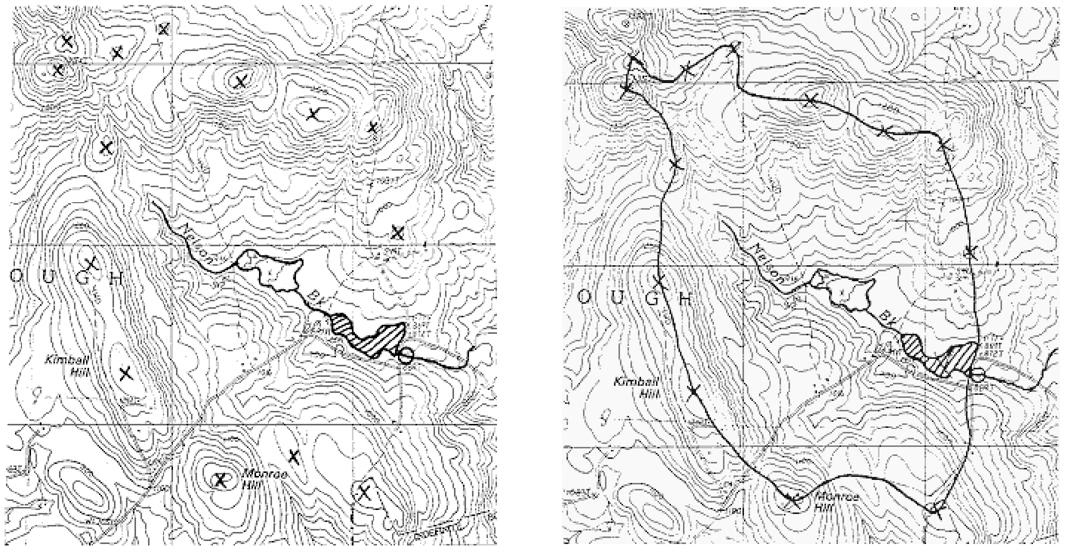


Figure 2.1: The New Hampshire method. The figures, as well as the method, are taken from Ammann and Stone [1]. In the left figure the two first steps are shown. The two last are shown in the right figure. Step 1: Choose and mark the outlet with a circle. Step 2: Mark all high points around the river and its tributaries with an x. Step 3: Start drawing a line from the circle to connect each x. Make sure the line is perpendicular to the contour curves. Step 4: Continue all the way around, and connect with the circle from the other side. The area within the line will be the watershed to the river outlet.

is the New Hampshire method. It was outlined by Ammann and Stone [1] and we explain the process in Example 1.

Example 1. Ammann and Stone's method has four steps, and the first two are illustrated in the left figure of Figure 2.1. The first step is to mark the desired outlet point in the river. This is indicated with a circle in the figure. Next, all high points along both sides of the upslope river and its tributaries are marked with an x. In steps three and four, each x is connected by forming a line that is starting and ending in the circle. It is important that the line is perpendicular to the contour lines. The resulting delineation is shown in the right figure.

What seems like a simple process can get very cumbersome, and often a great effort is required to obtain a correct delineation. However, the difficulty will vary for different topographies; an alpine landscape with large differences between the highs and lows is easier to delineate compared to a landscape with rolling hills and small variations in elevations.

With today's computer power and availability of *digital elevation models*[1] (DEM), delineation of watersheds is a perfect example of something that can be automated

[1]A *digital surface model* (DSM) is a representation of the surface which includes the elevations

with excellent results. Research concerning automatic delineation started already in the eighties, and many new algorithms emerged in the nineties. One of the algorithms is the one developed by Jenson and Domingue in 1988 [16]. They outline all the steps in a precise manner, some of which will be implemented in our algorithm. In the evaluation of their algorithm, two rivers in New York were considered: Susquehanna River and Genegantslet Creek. Jenson and Domingue reported a 97% spatial agreement between their delineation and the manual one for the Susquehanna River when using a planimeter[2]. After doing the same calculations for the Genegantslet Creek, a large section was found to not match the manual watershed delineation. Further examination concluded that there was an error in the manual delineation, a testimony to the success of automated methods. When this was fixed, the spatial agreement also turned out to be 97%.

As we recall from the beginning of this chapter, in order to obtain the hydrograph for a point in the landscape, we need the point's watershed. Once we know the watershed, we can create precipitation scenarios in the watershed, and see how this affects the hydrograph. We will get back to how we calculate the watershed with an automatic delineation algorithm in Chapter 3. In the next section we take a look at the hydrologic cycle and the water movement in the watershed.

2.2 The hydrologic cycle

The hydrologic cycle is also called the water cycle, and it describes how the water continuously circulates between the atmosphere, the surface of the Earth, and the sub-surface of the Earth. A simple example of the cycle is the journey a water drop undergoes from it falls as precipitation until it returns to the atmosphere. The time this takes, and the places it visits, depends on climate and terrain. If it trickles down into the groundwater or flows to the ocean, it might take hundreds, maybe thousands of years before it evaporates and becomes precipitation again. Figure 2.2 shows a simplified illustration of the hydrologic cycle.

A watershed can be treated as a closed system in which water is conserved. This is reflected in the water balance equation,

$$P = Q + E + S. \tag{2.1}$$

The left hand side of equation (2.1) represents the input to the system, which is only the precipitation P. The output from the system can be represented as the runoff Q at the watershed outlet, and the water that evaporates from the soil, lakes, plants etc., i.e., evapotranspiration E. Finally, S represents the water that is stored within the watershed.

of buildings and vegetation. To obtain the DEM, the non-ground points are filtered out, making it bare-surface. The remaining elevations are represented as a regularly-spaced raster grid where the heights are referenced to a common vertical datum.

[2]According to Encyclopedia Britannica, a planimeter is an instrument that can be used to measure an area bounded by an irregular curve — in this case the outline of a watershed.

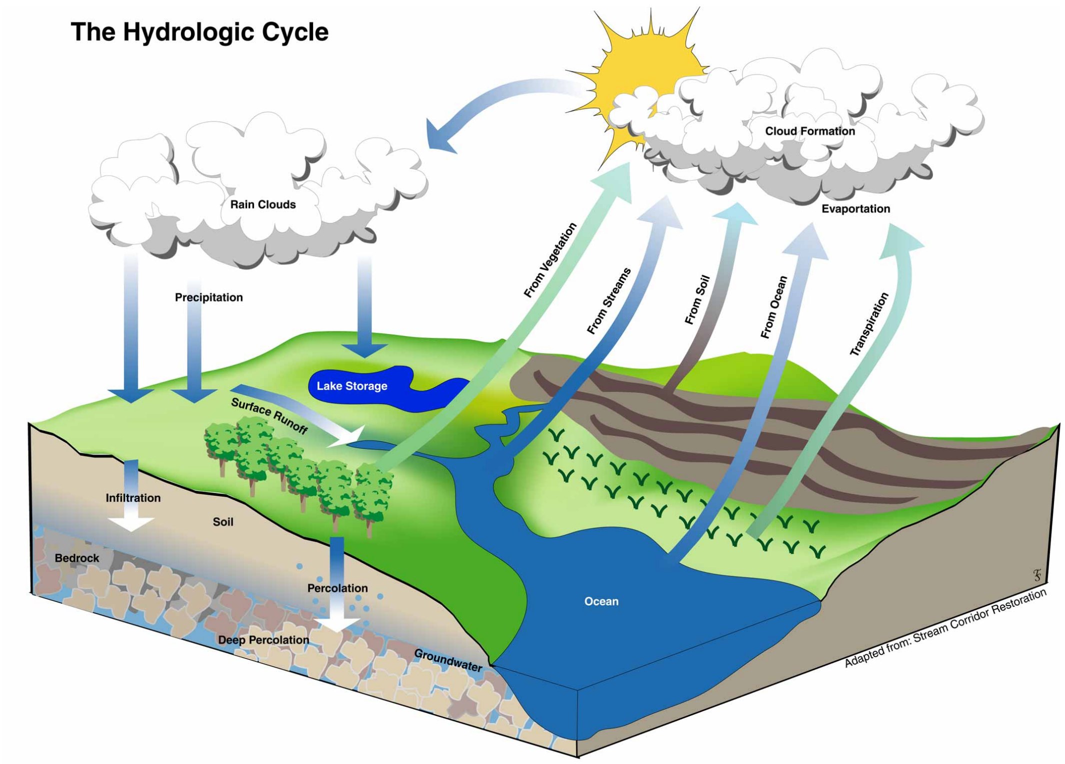


Figure 2.2: Illustration of the hydrologic cycle. The image is taken from 'Stream Corridor Restoration: Principles, Processes, Practices' [13, p. 2-3].

We will take a closer look at the movements of water that happens on, or below the surface of the landscape. In Figure 2.2 only the flow that happens on the surface — the *surface runoff* — is explicitly shown, but there are two other types of runoff as well; these are called *subsurface runoff* and *groundwater runoff*. The runoff types are often referred to as overland flow, interflow and baseflow, respectively. Collectively, they are referred to as *runoff*, which Rumynin [33] describes as the flow of a water layer over the surface and through the pores of soils and sediments that is coming out of the watershed.

The runoff types are quite different, and we will briefly explain the differences. Surface runoff is the water that flows at the surface of the landscape, which is driven by gravitational forces. Unlike surface runoff, both subsurface runoff and groundwater runoff take place below the ground surface. To explain the difference between them, the *vadose zone* is key. The vadose zone is the unsaturated zone where the pores in the soil are filled with both water and air. Water that passes the vadose zone and enters the groundwater system eventually becomes groundwater runoff. If water is blocked from reaching the groundwater by an impermeable or semi-impermeable layer, it ends up as subsurface runoff, which is a lateral flow at shallow depths in the vadose zone.

Characteristics of the watershed are important to predict how much water will flow at the surface. Precipitation will infiltrate the ground until it is either saturated or cannot infiltrate precipitation fast enough. Both types will initiate surface runoff. The first type is called *saturation excess runoff*, and the second is called *infiltration excess runoff*. One way of thinking about saturation excess runoff is that the ground is full of water from below; there is no more storage space. Infiltration excess runoff, on the other hand, happens because the water front is not moving fast enough inside the vadose zone. Different soil properties affect which runoff type will occur. The risk of a flood increases if either the soil is saturated and the water level in the vegetation is at the maximum, or if soil and vegetation cannot absorb water fast enough.

2.3 River hydrographs

A hydrograph shows the discharge for a point in the river over time, and is usually measured in cubic meters per second, which we will also do in this work. The river with the largest discharge in the world is the Amazon River, with a discharge of 209 000 cubic meters per second[3]. The Amazon River is rather exceptional as the next river on the list, the Congo River, only has a discharge of 41 200 cubic meters per second. The watersheds of these two rivers are vastly larger than the watersheds we will look at, so our river discharges will naturally be much smaller.

After a heavy rainfall we can expect that the discharge will increase at the watershed outlet. This alone is not necessarily enough to cause a flood, as the discharge from a river fluctuates constantly. However, the moment the *river stage* gets close to the *flood stage*, it is cause for concern. The river stage is the water level of a river with respect to a chosen reference height, and the river will not flood until the river stage hits the flood stage, which is when the water starts to flood the riverbanks. One example that illustrates river stage is shown in Figure 2.3. The illustration shows both the hydrograph and the river stage of the Mississippi River during the flood in 1993. It is important to keep in mind that river stage and discharge are not linearly related, as the discharge depends on the cross-sectional shape of the river [25]. Thus discharge alone is not enough to determine if a river will flood.

2.4 Rainfall-runoff models

To create a hydrograph for a given rainstorm, we need to predict the runoff after a rainfall, something *rainfall-runoff models* are designed to do. Traditionally these models have been used by engineers for flood forecasting and design flood estimations[4]

[3]According to Wikipedia's list of rivers by discharge: `https://en.wikipedia.org/w/index.php?title=List_of_rivers_by_discharge&oldid=767427793`

[4]A design flood is the hypothetical maximum flood that a structure is designed to withstand, for example a bridge.

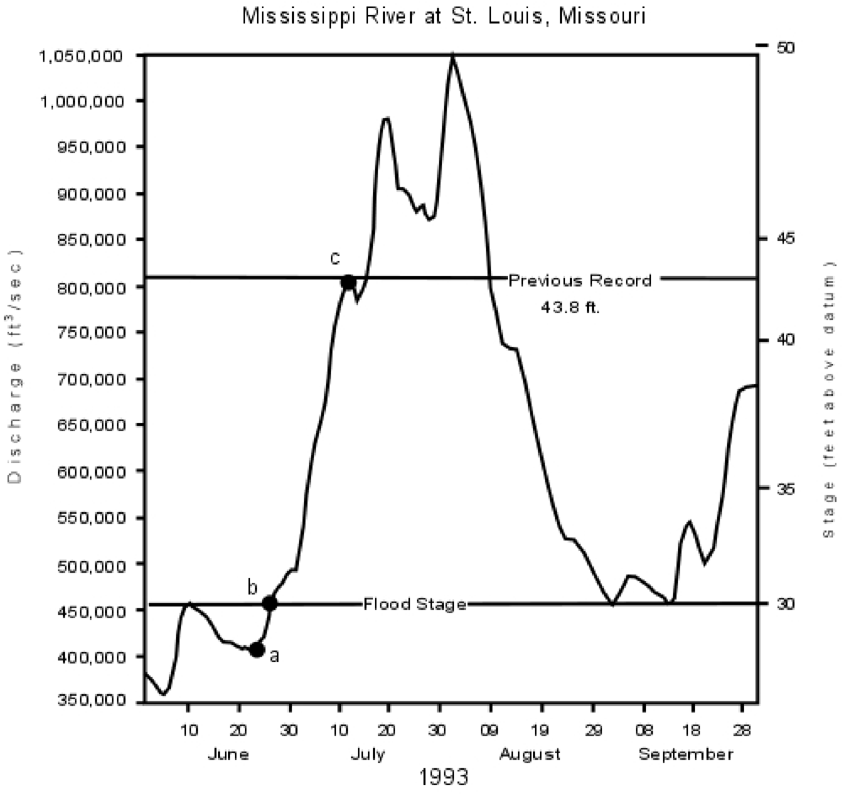


Figure 2.3: Hydrograph showing the river stage and discharge of the Mississippi River during the flood in 1993. The image is taken from Nelson [25].

[23]. The models are commonly classified into how they describe physical processes in the watershed (conceptually or physically), and how they describe watershed processes spatially (so-called lumped or distributed) [31]. In this work we will only scratch the surface of rainfall-runoff modelling, but two of the rainfall-runoff model types will be briefly explained: *lumped conceptual models* and *distributed physical models*. We will also discuss the state-of-the-art rainfall-runoff models, and the one that we will implement in this work.

One thing most rainfall-runoff models have in common, is that they must be calibrated before they can yield reliable runoff estimates [37]. To calibrate the models, historical input-output data is required for the watershed. It varies greatly how many parameters a model can have, and Australia's guidelines for rainfall-runoff modelling [37] gives an indication of how many are needed for each type: So-called *lumped* models are reported to have between 4 and 20 parameters, while *distributed* models can have from 10 to 1000's. In reality, distributed models might have fewer than this, because of the spatial correlation for parameters since cells close to each other typically have the same soil type. The lower number of parameters is one of the advantages lumped models have [20], which in turn makes them both easier and less time-consuming to calibrate. This is much of the reason why lumped models are so popular [18]. Fewer parameters also makes it possible to use automated calibration, which due to long run times has not been possible for distributed models [37] before. The advent of high-performing computers has made this much more attainable.

2.4.1 Lumped and distributed rainfall-runoff models

A lumped rainfall-runoff model treats a watershed as a single homogeneous unit [18], which means that all inputs, outputs and parameter values are averaged over the complete watershed [24]. Despite this, lumped models often yield good runoff estimates after they have been calibrated [18]. We already mentioned some of the upsides to the model parameters of lumped models, especially the relatively low number of them, but there is one drawback: most parameters are not necessarily connected to any measurable physical characteristics of the watershed. According to Australia's guidelines there are some exceptions to this, but overall the stronger link between physical properties and model parameters is why some prefer distributed models. Two popular lumped models are the Sacramento Model [4] and the Stanford Watershed Model [8].

Distributed models are different from lumped models as they account for spatial variations within a watershed. They also have the advantage that modelling parameters represent physical properties in the landscape. In a distributed model the watershed is discretized into small units, where each unit shares the same properties. Usually a regular grid or a triangular irregular network (TIN) [23] is used for this. These units can be quite large, with examples of 250 x 250 m [9] and 1 x 1 km [24]. In recent years the resolution of available DEMs has improved, so it is expected that much smaller units are used in the future. The higher resolution can potentially yield more accurate results, but will also increase the need for computational power. In our model to be presented later, we use cells that are ten by ten meters, but for some areas it is possible to get grids with a resolution down to one meter[5]. In theory, distributed rainfall-runoff models should provide better runoff estimates, but this is in general not the case in practice. Several studies have compared different models, where both lumped and distributed models were represented. In their studies, the distributed models did not perform significantly better than lumped models [18], nor did any single model outperform the other models in all cases [20]. Thus a model should be selected based on what it will be used for, and the availability of data. Two examples of popular distributed models are the Topmodel[6] [3] and the MIKE-SHE model [32]. An overview of lumped rainfall-runoff models and distributed rainfall-runoff models can be found in the article by Li et al. [20].

[5]'Høyde DTM 1 meter WMS' by Geonorge has a resolution of 1 meter.

[6]This is actually a semi-distributed model because it does not account for spatial variability of precipitation [20].

Something that has been a problem in the past, and still is today, is the prediction of runoff from ungauged watersheds. It is hard to calibrate rainfall-runoff models when the required historical data do not exist. As a consequence of this, lumped models are often preferred over distributed models for ungauged watersheds, as they require less data [18]. Two approaches have been developed to remedy the difficulties, called *regionalization* and *regional calibration*. In regionalization, the model is calibrated against a gauged watershed, and afterwards the model parameters are transferred to the ungauged watershed. In the second method, the parameters are calibrated simultaneously against multiple watersheds from a wide region [20]. The International Association of Hydrological Sciences[7] launched the so-called 'Predictions in Ungauged Basins (PUB)' initiative. It lasted from 2003 to 2012 [14], and was an effort to improve runoff predictions from ungauged watersheds. Despite IAHS's advances and the two methods we mentioned, the problem of predicting runoff from ungauged watersheds still persists today.

An ideal distributed rainfall-runoff model can capture the hydrological processes in each unit, and also calculate each unit's discharge contribution to the hydrograph. To capture the mechanism of runoff production in the physical model is hard to accomplish, and a large number of cells lead to difficulties in terms of computational cost [11]. This concern is nothing new. In the earlier days both computer power and distributed hydrologial data were limited [24]. This has been somewhat remedied in the last few decades. Weather data has become more easily accessible[8] [12], and computer power has increased enormously.

In a distributed rainfall-runoff model, many hydrological processes are interconnected [20] to explain the movement of water within the watershed. In the water balance equation (2.1), we saw that the input to the watershed, the precipitation, could be divided into three categories: runoff from the watershed, evapotranspiration, and storage in the ground, and in lakes and rivers. The models try to account for the amount of water that ends up in each category and how it moves around. There are a plethora of processes in the models, and some examples are: evaporation, soil moisture dynamics, subsurface processes, runoff generation mechanisms, snow melting and accumulation, and routing in lakes and rivers [20, 23]. Distributed rainfall-runoff models account for spatial variations within the watershed, including information about climate, terrain, soil type and vegetation. This information can improve the hydrologic predictions that are made for the watershed [20].

Garbrecht et al. [12] discuss remote sensing[9] and its potential use in distributed rainfall-runoff models. The authors also discuss how data be acquired for precipitation, land use, vegetation indices, surface temperature, soil moisture, stream networks and snow. This is data that is useful in distributed rainfall-runoff mod-

[7] IAHS's website can be found at: iahs.info

[8] In 'Hvorfor er norske værdata gratis?' from yr.no they talk about how they made Norwegian weather data free to the public.

[9] The acquisition of data by the use of satellite or sensors on airplanes. Examples include data about soil type, vegetation, rivers etc.

els, and the increase in available remote sensing data can potentially improve the hydrological predictions.

Beven [2] discusses current state of the art of rainfall-runoff modelling and argues that more powerful computers will continue to improve the advantages of distributed models, but he also questions whether the increased complexity will lead to better hydrological predictions. In the last two decades, rainfall-runoff models using neural networks have also been presented [40, 15], and it will be exciting to see where rainfall-runoff modelling is heading in the next decades.

2.4.2 Time-area method

We will base our rainfall-runoff model on the time-area method [7]. Ponze [30] refers to it as a lumped model, but according to Saghafian et al. [34] it can become a distributed one if we allow non-uniform precipitation and account for spatial variations within the watershed. Something similar is also done in Muzik [24].

The regular time-area method divides the watershed into sections of approximately equal travel time to the watershed outlet. Borders between these sections are called isochrones and indicate cells with equal travel time. It is possible to make a time-area histogram, which shows the size of the areas within each time interval. This is in turn paired with a storm hyetograph[10] to estimate the runoff hydrograph.

Travel times in the time-area method are estimated from e.g., the segmental velocity approach [5]. The flow paths are broken up into different segments where each has a defined flow; examples of this include overland flow and channel flow. To find the travel time, the length of the segment is divided by the associated velocity. The sum of the travel times from a cell to the outlet will then be the cell's travel time. The Manning equation is used to calculate velocities.

2.4.3 Our rainfall-runoff model

In this thesis, our goal is to implement a distributed rainfall-runoff model that utilizes the steadily increasing access to publically available weather and landscape data. It will be based on the time-area method, and we will use elevation data with high enough resolution to accurately represent the heterogeneous landscape.

To approximate the velocity field in the watershed, we assume the flow can be modeled as creeping flow. This way we can use Darcy's Law for a single-phase fluid [21, p. 117], which allow us to account for the watershed's heterogeneous properties. This includes the topography, as well as all other parameters we might want to include, such as vegetation, soil, land use etc. Data for this can be set on a cell level, which allows us to capture the heterogeneity of the terrain. To compute the travel times in the watershed, we solve the so-called *time-of-flight* equation. Unlike the time-area method, which divides the watershed into areas of approximately equal travel time, we will use each cell's travel time as basis for the hydrograph.

[10] Wikipedia's definition: 'A hyetograph is a graphical representation of the distribution of rainfall over time'. https://en.wikipedia.org/w/index.php?title=Hyetograph&oldid=649439813.

The rainfall-runoff models we described earlier in this chapter need to be calibrated for every new watershed if we want a good approximation of the surface runoff. For our model, we have an ambition that it should provide good runoff estimates after calibrating physical parameters once, given that all necessary data (e.g., from remote sensing) is available. This will make it work for gauged and ungauged watersheds alike. Due to the limited time we have in this project, we will only include the topography to estimate the travel times, but ideally heterogeneous properties can be added.

2.5 How can we predict floods?

A variety of information is needed to predict floods. The U.S. Geological Survey[11] lists four points that are important for flood prediction. The first two concern monitoring of the amount of rainfall and the rate of change in the river. Both happen on a real-time basis, but require preinstalled equipment. The third point is knowledge about what type of storm it is, such as duration and intensity. This is knowledge that the weather forecasting service can provide. The last point is about the landscape. Examples include the size of the watershed, soil-moisture conditions and the vegetation.

In our experiments, we will consider a precipitation scenario on a watershed, and based on the time required to reach the watershed outlet, calculate a hydrograph. The hydrograph is then investigated to see how the discharge at the outlet is affected by the precipitation. This can be used to predict a flood. As we mentioned in the previous section, it is not possible to say this for sure without a cross sectional-shape of the river. However, in this work we will focus on the hydrograph.

[11] Information about how floods are predicted is presented in the frequently asked questions in the USGS Floods and Droughts.

3 Delineate watershed

In Section 2.1 we gave a brief introduction to what a watershed is, and briefly discussed how the art of watershed delineation has been transformed with the advent of computers. Later, in Section 2.4, we talked about the watershed's importance in rainfall-runoff modelling. In this chapter we will outline and implement an algorithm that delineates the watershed of a location in the landscape. In Appendix A.1 we talk more about the data set that we will use. The delineated watershed we obtain in this chapter will be used as input for our rainfall-runoff model.

Our algorithm can be divided into roughly four steps. In the first step, we make the landscape *depressionless*. A depression is an area that is completely surrounded by higher elevations, which hinder flow routing and cause problems for the delineation process [36]. When the landscape is made depressionless, elevations in the DEM data are increased so that the depressions are removed. In the resulting landscape, every cell has at least one monotonically decreasing path of cells that leads to the domain boundary.

In the second step we divide the newly acquired landscape into watersheds based on the flow directions in the landscape. Before we made the landscape depressionless there were many flat areas that represented water accumulations like lakes, dams, and ponds in the DEM. During the process in step one, the elevations of these areas will increase or remain the same. As an optimization, we do not define flow directions in flat areas, which we will come back to later. Because of this lack of flow directions, the flat areas effectively trap the water (because they represent lakes, dams, and ponds), which is why we refer to them as *traps*. Each trap in the landscape has an upslope area it gathers water from, which is its watershed. We can calculate where the trap will spill over into another watershed, which the spill pairs will tell us.

The information we acquire in step two is enough to create a connectivity matrix between cells in the landscape. The matrix tells us the upslope and downslope neighbors of each cell. We can use this connectivity matrix to calculate the flow accumulation for the entire landscape. The flow accumulation plot shows locations that have a large upslope area, and hence have a potential to gather a lot of water. This should come close to a plot of rivers and lakes in the landscape. The plot will also be used to select a location for our rainfall-runoff model, i.e., the location we will create a hydrograph for.

In the fourth step, we compute the watershed of the outlet based on the connectivity matrix and the outlet cell. Once we have the connectivity matrix for the landscape, we can delineate the watershed of any location in the landscape. The four steps are combined into Algorithm 1, and we will walk through the algorithm step-by-step in this chapter. Because many of the variable and function names are self-explanatory, they will not always be explicitly stated.

In the next chapters we will calculate travel times for each cell in the outlet's delineated watershed, and construct precipitation scenarios to obtain hydrographs that measure the flow at the outlet.

Algorithm 1 CalculateWatershedOfOutlet takes a DEM E as input, and delineates the watershed of a location in the landscape. The location is chosen based on the flow accumulation in the landscape.

```
function CalculateWatershedOfOutlet($E$)
    $E \leftarrow$ FillSingleCellDepressions($E$)
    $flow\_directions \leftarrow$ CalcFlowDirections($E$)
    $flow\_destinations \leftarrow$ CalcFlowDestinations($flow\_directions$)
    $local\_watersheds,\ local\_minima \leftarrow$
      GetLocalWatersheds($flow\_destinations$)
    $combined\_minima \leftarrow$ CombineAdjacentMinima($local\_minima$)
    $watersheds \leftarrow$ CombineWatersheds($local\_watersheds,\ combined\_minima$)
    $watersheds,\ spillPairs \leftarrow$
      CombineWatershedsSpillingIntoEachOther($watersheds,\ E$)
    $dE \leftarrow$ RemoveDepressions($watersheds,\ spillPairs,\ E$)

    $watersheds,\ spill\_pairs,\ flow\_directions \leftarrow$ RecalculateWatersheds($dE$)
    $spill\_heights,\ traps \leftarrow$ GetTraps($dE,\ watersheds,\ spill\_pairs$)

    $conn\_mat \leftarrow$ CreateConnectivityMatrix($flow\_directions,$
      $traps,\ spill\_pairs$)
    $accumulated\_flow \leftarrow$ CalculateFlowAccumulation($conn\_mat$)
    $outlet \leftarrow$ DecideOutletBasedOnFlowAccumulation($flow\_acc$)
    $watershed\_of\_outlet \leftarrow$ ComputeWatershedOfOutlet($outlet,\ conn\_mat,$
  $traps$)
    return $ws$
end function
```

3.1 Fill single-cell depressions

One of the major obstacles in flow modeling is the presence of *depressions*. A depression is an area that is completely surrounded by higher elevations, which makes it impossible for flow to continue. Jenson and Domingue [16] dub it the nemesis of determining hydrologic flow directions. O'Callaghan and Mark [26] attempted to remedy the problem by smoothing the DEM data, which could fully remove shallow depressions. Jenson and Domingue [16] went even further and created a depressionless DEM in a process in which removed depressions are turned into flat areas. Afterwards, the flow directions in the flat areas are defined in an algorithm that uses the outflow points in the flat areas as starting points. In each

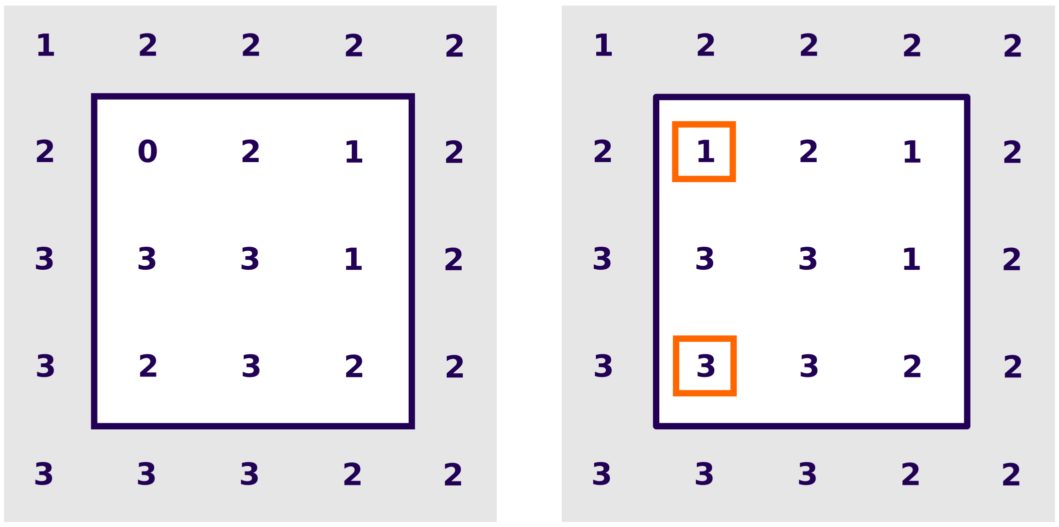


Figure 3.1: The process where single-cell depressions are filled. The left figure shows the elevations of the landscape before single-cell depressions are filled. We only consider interior cells (white background) as potential candidates for filling. In the right figure, all cells with increased elevation have an orange border. Notice that the elevation of each single-cell depression has been raised to the lowest neighbor elevation.

iteration, cells in the flats are assigned a flow direction to a neighbor that has already been assigned a flow direction, given that it is does not point back to the cell in question. This process will eventually assign flow directions to all cells in the domain.

In Algorithm 1, we begin with the same first step as Jenson and Domingue's algorithm [16] does — by filling *single-cell depressions*. A single-cell depression is a cell where all eight neighbors in the grid are at a higher elevation than the center cell. Thus, the center cell has no downslope flow direction. By raising the elevations of the single-cell depressions to the heights of their respective lowest neighbors, the number of depressions in the landscape are reduced. Because the number of local minima is reduced, this is a simple adjustment to reduce the complexity of our computations. An example of the filling is shown in Figure 3.1, where the left figure shows the elevations of the cells. In the right figure, the orange borders indicate the single-cell depressions that have been filled; these have now acquired the heights of their respective lowest neighbors. The two cells that have a value of 1 next to each other is also a depression, but because it is not a single-cell depression, it is not removed. As for the cells at the domain boundary, there is not enough information to determine if they are single-cell depressions. Hence, we only consider to raise the cells on the white background. After all single-cell depressions have been filled, we update the elevations in E.

3.2 Compute flow paths

When we model water flow for a raster model, we need to decide on an algorithm that determines the flow directions in the grid. Several algorithms exist, and one of the key distinctions between them is whether or not they allow flow from one cell to multiple neighbor cells at the time. The algorithms that do not allow this are called single flow direction (SFD) algorithms, while those that do are called multiple flow direction algorithms (MFD). The SFD algorithms often perform poorly on flat or convex surfaces [29], but are satisfactory on concave surfaces. Perhaps the most common algorithm is the D8 Algorithm (deterministic eight directions) [26], which is an SFD algorithm. The algorithm determines flow direction based on the neighbor cell it has the steepest descent to. If there are multiple cells with the same gradient, different variations can be implemented, such as selecting the first candidate in the clockwise direction, choosing the middle one if there are three adjacent directions, etc. In this thesis, we will implement the D8 Algorithm, and we will not go into detail about any other (we discuss several in 'Automatic Delineation and Analysis of Watersheds' [39]). In Appendix B, we will mention some of the drawbacks of this algorithm and also have a small discussion on D8 versus D4 (only flow in cardinal directions).

One of the assumptions we make when we delineate a watershed is that the landscape is impermeable and vegetationless. This implies that only surface runoff affects the delineation and that subsurface flow is ignored.

In the D8 Algorithm, the flow direction for an arbitrary cell is towards the neighbor cell with the largest positive $\Delta z/x$, where Δz is the elevation difference between the cell and the neighbor, and x the distance between them in the 2D-grid. Because our DEM has a resolution of 10 meters, x_i will be 10 m for cardinal neighbors, and a factor of $\sqrt{2}$ larger for diagonal neighbors. If we number the cell's neighbors in a clockwise manner from one to eight, we can encode the flow direction as

$$d = \begin{cases} 2^{i^*-1}, & \text{if } \Delta z_{i^*} > 0 \\ -1, & \text{if } \Delta z_{i^*} \leq 0 \end{cases} \tag{3.1}$$

where i^* is

$$i^* = \underset{i \in \{1,2,\dots,8\}}{\arg\max} \frac{\Delta z_i}{x_i}. \tag{3.2}$$

Here i^* is the index of the chosen neighbor, and the resulting flow direction d is encoded the same way as the one Jenson and Domingue uses [16]. For an arbitrary cell c, the possible encodings are shown in Figure 3.2. If none of the cell's neighbors are lower in elevation, we set d to -1. These cells will be referred to as local minima cells in Algorithm 1. Later in the algorithm, we will combine adjacent local minima.

In the following we will not calculate flow directions for boundary cells, which means that these cells are only used in the calculations of the interior cells' (cells with eight neighbors) flow directions. This is because we do not have enough information to determine the boundary cells' flow directions. One upside of this, is that we avoid special cases where the number of neighbors vary, which makes the implementation easier.

When we run the algorithm, each cell in the grid is assigned a flow direction that is either one of the cardinal directions (N, S, W, E), or one of the diagonal directions (NW, NE, SW, SE), with the exception of cells where none of its neighbors are at a lower elevation. In Example 2 we explain how the encoding can be useful for an MFD algorithm. For now we will only allow flow in one direction from a cell, so another possible encoding could be to simply use the index of the chosen neighbor.

Example 2. For an SFD algorithm, the flow direction encoding is 1 for a northeastern flow direction, 2 for an eastern flow direction and so on, but an encoding as a power of 2 also allows us to uniquely describe multiple flow directions at once. This can be useful for an MFD algorithm. One example of this is the simultaneous flow in the three directions N, SW and S. If we sum their individual encodings, we get a value of 152, which can only be achieved by these exact directions.

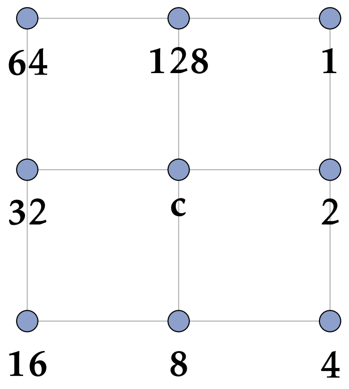
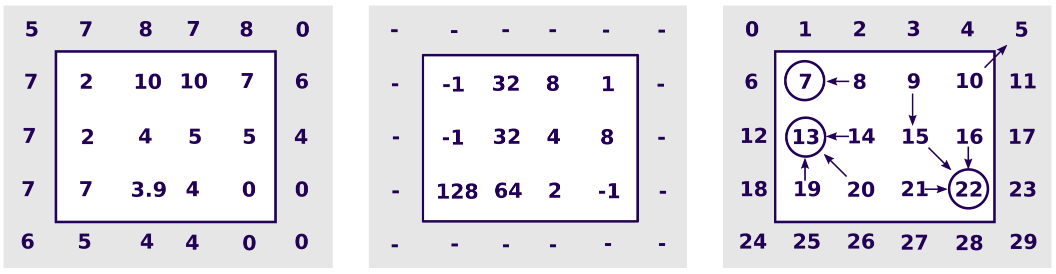


Figure 3.2: Flow directions encoding for an arbitrary cell x. If the flow is towards the northeast its flow direction is encoded as 1. East is 2, southeast is 4, etc. This is the same encoding used by Jenson and Domingue (1988) [16].

We will now look at an example grid where the elevations for the grid points are given in Figure 3.3(a). After we have used Equation (3.1) to calculate the flow directions for the interior cells, we obtain the result in Figure 3.3(b). In Example 3 we explain a little bit more about the result, and we will use the cell indices in Figure 3.3(c) as part of the explanation.

Example 3. In this example we will make some remarks about the resulting flow directions in Figure 3.3(b). Our flow direction algorithm assigns the flow directions according to Equation (3.1) — if a cell has a neighbor at a lower elevation, the cell is assigned one of the flow directions in Figure 3.2, but if it does not, its flow direction



(a) Landscape elevations. (b) Flow directions. (c) Overview using indices.

Figure 3.3: An example showing a landscape and the flow directions for the interior cells.

is set to -1. Note also that only the interior cells (white background) have been assigned flow directions.

In the landscape, three cells have been assigned a flow direction of -1: cells 7, 13 and 22. If we look at the elevations in Figure 3.3(a), we see that these local minima cells are part of flat areas, where none of their neighbors are at a lower elevation. In Figure 3.3(c) all local minima have been encircled.

To delineate watersheds, we want to answer the question: 'If a bucket of water is poured out in cell *c*, where will the water flow?' This question must be answered for all cells in the domain. The flow from cell *c* will stream from cell to cell until it hits a local minimum cell, which we refer to as cell *c*'s flow destination. In Example 4 we find the flow destination (and flow path) for one of the cells in Figure 3.3(c). Because we do not assign flow directions to boundary cells, we will also not allow flow to the boundary. This will not change the watershed delineation of the interior cells. In cases where the flow direction is towards the boundary, we will change the flow direction to -1. This has not been done yet to cell no. 10 in Figure 3.3(c).

Example 4. In this example we want to find the flow path and flow destination of a cell in the landscape shown in Figure 3.3(a). In Figure 3.3(c) we can see the resulting flow directions in the landscape. All flow paths will end up in one of the local minima cells, and for reasons we just discussed, we also consider cell no. 10 to be a local minimum. Because the local minima cells are part of pits or lakes, it is natural that the other cells are routed to them.

We want to study the flow path from cell no. 9 in Figure 3.3(c). The flow directions in Figure 3.3(b) tells us that water flows southwards from no. 9, which brings us to cell no. 15. From here the flow is in the southeastern direction, i.e., towards cell no. 22. Because its flow direction is encoded as -1, the flow cannot continue. This means that cell 22 is cell 9's flow destination.

To do this for all cells, we first alter the flow directions for the cells with flow to the boundary. Afterwards we transform the flow direction encodings to cell indices. For each cell, we route the flow to the next cell, and check if it is a local minimum.

If it is, we have found its flow destination, otherwise, we proceed. When we have calculated all flow destinations, we can combine the cells that have the same flow destination into a *local watershed*. In a local watershed every cell has a flow path to the same minimum. In Example 5 we show the local watersheds from the landscape in Figure 3.3(a).

Example 5. The left figure in Figure 3.4 shows the landscape from Figure 3.3(a) divided into local watersheds, of which there are four. We can see that all cells are connected to a minimum: cells 9, 15, 16 and 21 are connected to cell 22; cell 8 is connected to 7; cells 14, 19 and 20 are connected to cell no. 13, and cell no. 10 make out its own local watershed.

In Figure 3.3(a) we can see that the two adjacent minima 7 and 13 are actually at the same elevation, so water can flow between the two cells. Thus it makes sense to combine their local watersheds, which brings us to the next step — connecting adjacent minima cells, and their respective local watersheds.

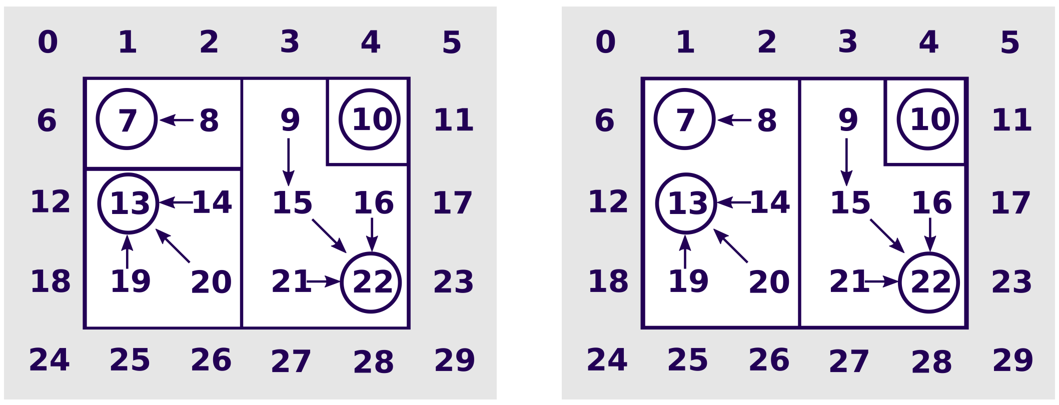


Figure 3.4: The left figure shows the local watersheds in a landscape. The blue borders separate the local watersheds in the interior of the domain (white background). The minimum cell in each local watershed is encircled and the flow direction of each cell is shown. In the right figure we combine the local watersheds of all minima that are adjacent to each other. This is because water can flow between the adjacent minima cells, and hence between their local watersheds.

3.3 Combine local watersheds

DEMs are usually pre-processed in different ways, and we can take advantage of this to locate the lakes. In our DEM, the grid points in each lake have been assigned the exact same elevation, and we will assume this only applies to lakes[1], so that

[1]It is not known whether e.g., parking lots and soccer fields have also been assigned elevations in this way.

flat areas are equivalent to being lakes. We know that all cells in a flat area will be classified as local minima cells (i.e., cells with a flow direction of -1), so if we combine all adjacent minima cells in the landscape, we will obtain the lakes. Now that a lake can be represented by a collection of minima cells, we can obtain the lake's watershed if we merge the local watersheds of the minima cells.

To combine local watersheds of adjacent minima, we need to know which minima are adjacent to each other. To do this, we look at each minimum and its eight neighbors. If the minimum has a neighbor that is also a minimum, we create a tuple with the indices of both minima. Each tuple represents that the adjacent minima cells are connected, and that water can flow between them. All the connections can be made into an undirected graph, where the vertices are the minima cells and the edges are the connections between them. To get all adjacent minima we calculate the connected components of the undirected graph. In Example 6 we show an example.

Example 6. Let a landscape contain five minima cells m_i where $i \in [1, 2, \ldots, 5]$. We first add all minima cells as vertices to our undirected graph, and afterwards we find the connections between them. For each minimum we check if it has any neighbor cells that are also minima cells, and if it has any, we add edges to represent that they are adjacent to each other. In our example we assume m_1 and m_2 are adjacent, so we add an edge between them. If m_3 is adjacent to m_2, we add another edge. This leaves the cells m_4 and m_5. Let us assume the two cells are only adjacent to each other, and not adjacent to any of the other three. Then a third and final edge is added. After all minima have been checked, we calculate the connected components from the undirected graph. In this case we get two connected components: $\{m_1, m_2, m_3\}$ and $\{m_4, m_5\}$. These are shown in Figure 3.5.

After we have combined adjacent minima cells, we combine their respective local watersheds to obtain the watershed of each combination. When we do this for the example in the left figure in Figure 3.4 we get three combinations of minima (three connected components), which means we get three watersheds. The obtained watersheds are shown in the right figure in Figure 3.4. The local watersheds of cells 7 and 13 have been merged as their minima were adjacent to each other, whereas the two others have been left as is because they did not have neighboring minima cells.

3.4 Compute spill pairs for all watersheds

For a large DEM we will have thousands of watersheds, and for each watershed we want to figure out where it will spill over if it is filled with water. Once we know the elevations of the spill points, we are close to having obtained a depressionless landscape.

The cell where it pours out of the watershed, and into another, is called the *spill point* (some articles may refer to it as the pour point). The pair with the spill point, and the cell that it spills to, is called the *spill pair*. Often there are multiple viable choices of spill pairs, and we will base our choice of spill pair on three factors:

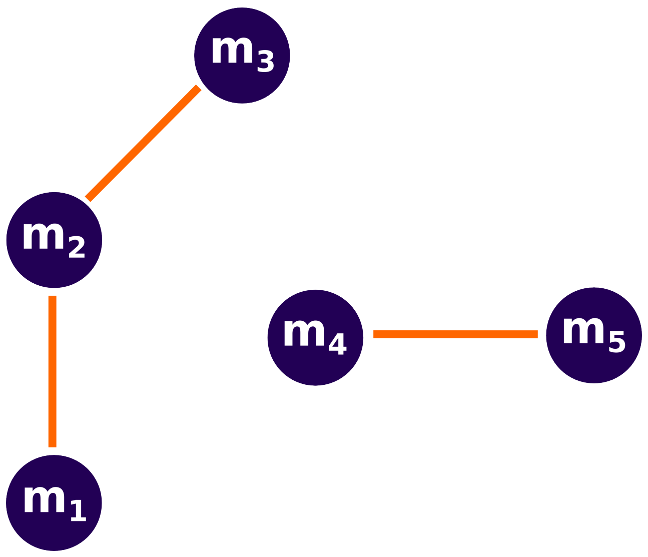


Figure 3.5: Let the vertices labeled m_1 through m_5 represent five minima cells in a landscape. For each minimum m_i we check if any of its neighbors are minima; if this is the case, we add an egde between the two vertices. Afterwards, if we calculate the connected components of this undirected graph, we will get the minima that are adjacent to each other. In this case we get the two connected components $\{m_1, m_2, m_3\}$ and $\{m_4, m_5\}$.

the elevation of the *boundary cells* in the watershed, the elevation of cells at the boundary of the surrounding watersheds, and the elevation of the domain boundary. A boundary cell is here referred to as a cell in the watershed where at least one of its eight neighbors are outside of the watershed. If there happens to be multiple suitable spill pairs, we choose the one with the steepest descent.

In Algorithm 2 we calculate the spill pairs for the watersheds. The first step is to calculate all *boundary pairs* for each watershed. A boundary pair is a pair of cells, where the first cell is located in the watershed in question, while the other cell is a neighbor cell in another watershed, or at the domain boundary. Out of all the boundary pairs, only one will be chosen as the watershed's spill pair. This procedure is basically the same as the one found in Jenson and Domingue [16] for determining pour points between watersheds.

Next, we turn our attention to the left figure in Figure 3.6, which shows a landscape with three different watersheds: one orange, one green and one blue. In Example 7 we will walk through the algorithm to obtain the spill pairs for the three watersheds.

Example 7. In this example we will walk through the steps in Algorithm 2 to obtain the spill pair for the blue watershed in the left figure in Figure 3.6. In the right figure we show the elevations of the landscape. We will also find the spill pairs for the green and orange watershed.

Algorithm 2 The CalculateSpillPairs algorithm calculates where a watershed will spill over, and where it spills to, if it is filled with water.

```
function CalculateSpillPairs($watersheds, E$)
    $spill\_pairs \leftarrow []$
    for each $ws$ in $watersheds$ do
        $boundary\_pairs \leftarrow$ CalculateBoundaryPairsOfWatershed($ws$)
        $max\_pair\_elevations \leftarrow$ GetMaxPairElevations($boundary\_pairs, E$)
        $spill\_height \leftarrow$ GetMinimumValue($max\_pair\_elevations$)
        $spill\_pairs \leftarrow$ PotentialSpillPairs($boundary\_pairs, spill\_height$)    ▷ Boundary pairs where their min of max equals spill height
        $selected\_spill\_pair \leftarrow$ SelectSteepestDescentSpillPair($spill\_pairs, E$)
        $spill\_pairs \leftarrow [spill\_pairs, selected\_spill\_pair]$
    end for
    return $spill\_pairs$
end function
```

The first step in the spill pair-algorithm is to calculate the boundary pairs for the watershed. One example of this, is the boundary pairs of cell no. 31, which have been drawn in the figure. After we have found all boundary pairs for the other cells in the blue watershed, we take the maximum elevation of each pair. One example of this is the pair (31, 23) which has a maximum elevation of eight. If we do this for all boundary pairs in the blue watershed, we get a list of maximum elevations; the spill height h_{spill} is obtained when we take the minimum of this list. The spill height of the blue watershed will be $h_{\text{spill}} = 3$, a value which is obtained in the two boundary pairs (31, 25) and (31, 32), so these pairs will be the output from the PotentialSpillPairs-function in Algorithm 2. We see from the right figure that if we fill the blue watershed with water to the spill point elevation, it will overflow into the orange watershed for both pairs. We could also have had a landscape where the pairs spilled into different watersheds, and we had to choose one of them.

In reality there will typically be a difference in elevation, so that the spill pair is unique. However, the data precision may not reflect this, which results in two points being assigned the same elevation. If we have multiple spill pairs in our algorithm, we choose the pair with the steepest descent away from the watershed. This is a convention, and we could have chosen any of them. Based on this the boundary pair (31, 25) is chosen as spill pair for the blue watershed.

Afterwards, we perform the same steps for the other watersheds. The orange watershed will have (26, 27) as its spill pair with a spill height $h_{\text{spill}} = 3$, while the green watershed has (8, 0) as spill pair and $h_{\text{spill}} = 4$ as spill height.

Occasionally, the spill pairs will create a cycle when water spills from watershed to watershed. If we change the elevation of cell 27 in Figure 3.6 from two to ten, this will create a cycle in our example. The spill pair of the orange watershed will change to (32, 31), which creates a situation where the orange and the green watershed spill

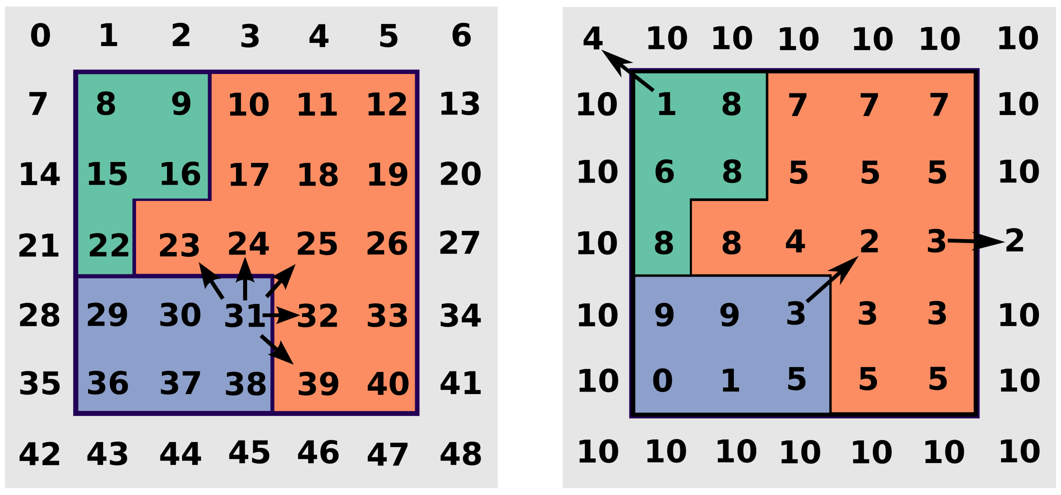


Figure 3.6: The left figure shows a landscape with three watersheds: a blue, an orange and a green. The boundary pairs for one of the cells in the blue watershed have also been drawn. In the right figure, the elevations in the domain, and the spill pairs (black arrows) of the watersheds are shown. The spill pair arrows shows which watersheds that will flow to which given that the domain is filled with water until the watersheds overflow.

into each other. In those special cases, the watersheds are combined, and a new spill pair is calculated. In this case, the new spill pair will be (23, 15) with a spill height of $h_{\text{spill}} = 8$. Two watersheds that are spilling into each other are in reality only two sub-watersheds of a larger watershed, which is why we merge them. If the cycles are not removed, there will be watersheds that potentially can accumulate infinite amounts of water. A cycle will also make it impossible to create a hierarchy of watersheds.

The process where we combine watersheds that spill into each other can potentially require many iterations. This especially holds true for a large grid. Figure 3.7 tries to explain why this is so. The example shows four different watersheds that are iteratively combined because they are spilling into each other, a process which requires four iterations. With small variations in elevations and thousands of watersheds, the number of iterations can quickly accumulate.

3.5 Identify traps and raise elevations

In a depressionless landscape there will always be a monotonically decreasing path to the boundary. This is not the case for our landscape yet, but after we found the spill pairs and spill heights for each watershed in Section 3.4, we are closer than ever.

We call the region below the spill height in the watershed for a *trap*, because

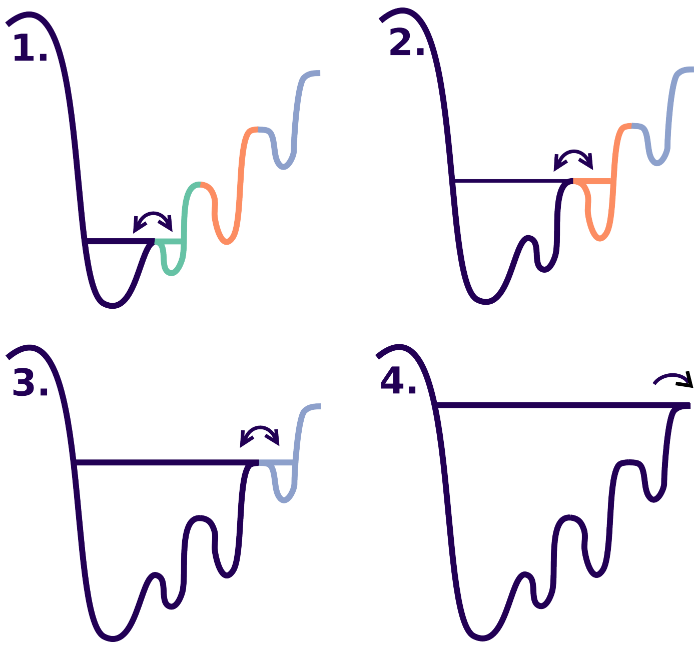


Figure 3.7: Illustration of the process where watersheds that spill into each other are iteratively added. In the process, we imagine that we fill all watersheds in the landscape up to the brim, and then add a little more water. If two watersheds spill into each other, we merge the two watersheds. After each merge, the watersheds are again filled to the brim, and the process is repeated.

it essentially traps all water below the spill height. In our DEM we can probably assume that the surface elevations of the lakes are above the elevations of the rivers' riverbeds flowing out from the lakes. This means that the flow from a lake will continue even if its surface elevation is reduced. It is not certain that this is picked up by the DEM. If a lake has a surface elevation below that of its surrounding area, and the DEM does not have a resolution that picks up the river running out of it, the spill height for the lake's watershed will be set to a higher elevation than the lake's surface elevation. The consequence of this is that the traps might have a larger area than the lakes.

Now that we have both the watersheds and the spill heights, we can identify the trap in each watershed as the cells that are below the spill height. The final step we need to do to obtain a depressionless landscape is to raise the elevations of the trap cells to their respective spill heights. This way all cells in a watershed are either above the spill height, or at the spill height, which ensures that the flow will not stop in a depression (because they have been removed).

If the flow directions are to be used in further calculations, it is important to remember that the new elevations of the depressionless landscape will also change the flow directions; areas that were once hilly, might now be completely flat. In a

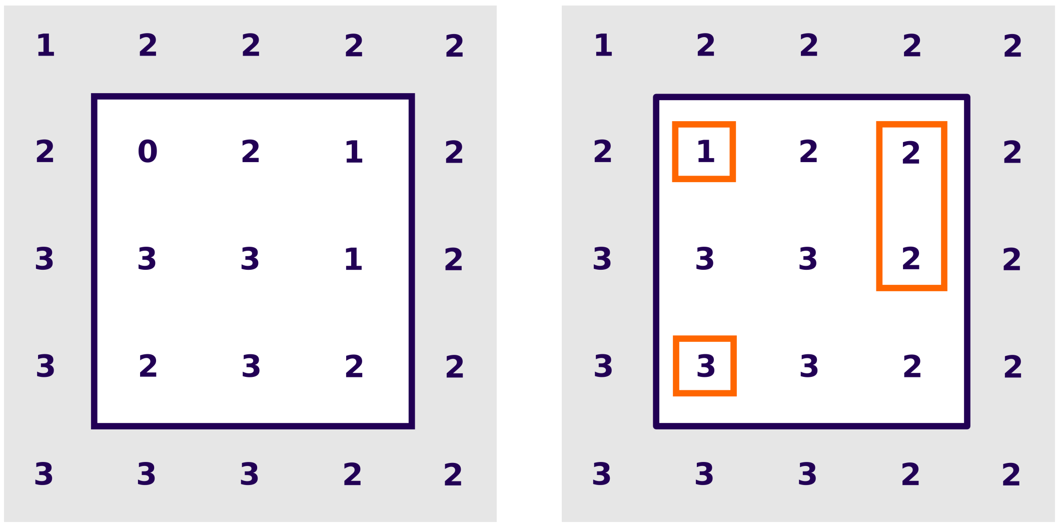


Figure 3.8: The initial landscape before we make it depressionless is shown in the left figure. In the right figure all depressions (orange borders) have been filled to make it depressionless. We can see that every cell in the depressionless landscape has a flow path to the boundary that is monotonically decreasing. This ensures that the flow does not stop anywhere.

later chapter we experienced a problem where the the delineated watershed of an outlet had areas with adjacent traps in the landscape. To avoid this problem, we will recalculate the watersheds and spill pairs.

Figure 3.8 shows how the example landscape from Section 3.1 changes when we make it depressionless. The left figure shows the initial landscape, while the right shows the resulting depressionless landscape. All depressions that have been raised in the process have an orange border around them.

Two slightly more advanced examples are presented in Figure 3.9. The top left figure shows the elevations of the landscape before it is made depressionless (E in Algorithm 1). It is the same elevations as in the right figure in Figure 3.6. The green and blue sections represent the different watersheds, and the spill points are indicated with arrows. In the top right figure, we show how the depressionless landscape will look like (dE in Algorithm 1). In the figure to the bottom left we change the elevation of one of the cells at the right boundary, which changes one of the spill points. Compared to the top right figure, this radically changes the depressionless landscape.

3.6 Create cell connectivity matrix

In order to obtain the watershed of an arbitrary cell in the landscape, we create a sparse connectivity matrix *conn_mat* that is based on the calculated flow directions,

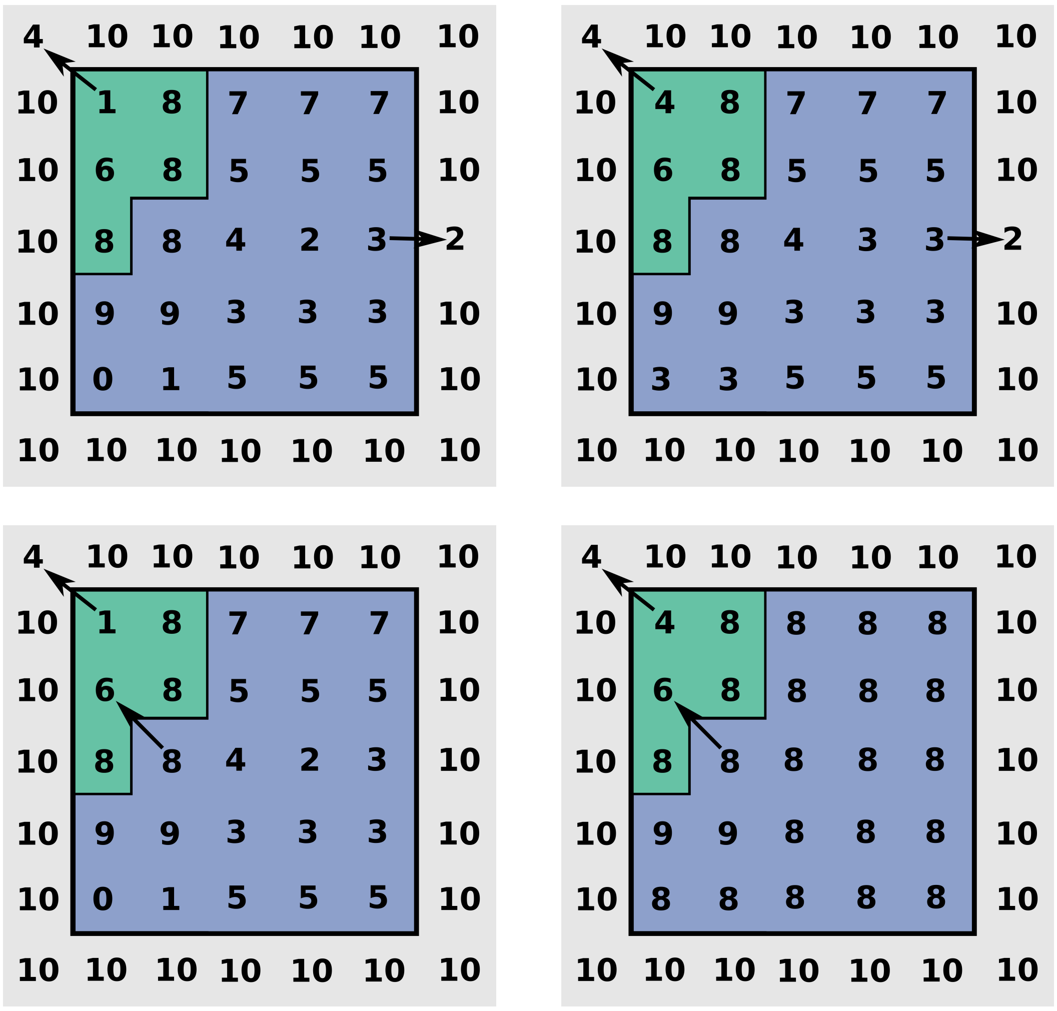


Figure 3.9: The top left figure shows two watersheds, their spill pairs, and the elevations before we make the landscape depressionless. In the top right figure we show the resulting elevations. The two bottom figures show a slightly different landscape, where we have altered the elevation of one of the boundary cells (increase from two to ten). In the bottom right figure we can see that this radically changes the depressionless landscape (compared to the top right figure).

the traps and the spill pairs. The elements in the connectivity matrix represent which cells are upslope and downslope of each other. If water in cell x flows to cell y, there will be a one in position (x, y) in the connectivity matrix. The connectivity matrix makes it possible to obtain the watershed of any cell in the landscape, by tracing its upslope cells. If our landscape is represented by a DEM of size m x n, the connectivity matrix *conn_mat* will be mn x mn, but because it is sparse it will not store that many elements. The algorithm to construct *conn_mat* is presented below in Algorithm 3.

Algorithm 3 The CreateConnectivityMatrix algorithm constructs a connectivity matrix for the cells in the landscape based on *flow_directions*, *traps* and *spill_pairs*.

```
1: function CreateConnectivityMatrix(flow_directions, traps, spill_pairs)
2:     conn_mat ← ConstructInitialConnectivityMatrix(flow_directions)
3:     ExpandConnMat(traps, conn_mat)
4:     AddDownslopeFromTrapNodes(traps, spill_pairs, conn_mat)
5:     AddUpslopeToTrapNodes(traps, conn_mat)
6:     RemoveUpslopeToOldTraps(traps, conn_mat)
7:     RemoveFlowOutOfBoundary(conn_mat)
8:     return conn_mat
9: end function
```

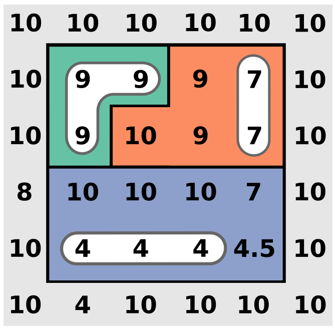
We will use Figure 3.10 to illustrate the algorithm, and the landscape elevations are shown in the left figure. The first step is to construct an initial *conn_mat* based on the flow directions in the middle figure, which we show in Example 8.

Example 8. The grid in the middle figure in Figure 3.10 has a size of 6x6, which means that *conn_mat* is 36x36. If we add a connection for every flow in the figure, the only nonzero elements in *conn_mat* will be in the positions: (9, 10), (14, 15), (15, 16), (19, 25), (20, 26), (21, 27), (22, 28) and (28, 27). The flow of water will be from the first index to the second index in each pair. Notice that there is no internal flow in each trap, which will hinder us from obtaining a correct watershed delineation for an outlet. To get it to work, we have to modify *conn_mat*.

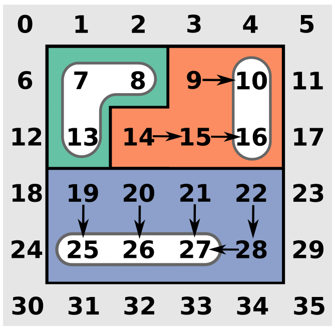
In Example 8 we mention the lack of flow between cells in the traps. We have purposely refrained from assigning flow in traps to make the code more efficient. To complete the flow within the whole landscape, we represent all cells in each trap by a single cell which we call a trap node. Afterwards we reroute all incoming and outgoing connections to the trap nodes.

In the connectivity matrix, we first expand the size of *conn_mat* by the number of traps t. The trap nodes are numbered from mn x mn up to mn x $mn + t$, which we show in Example 9.

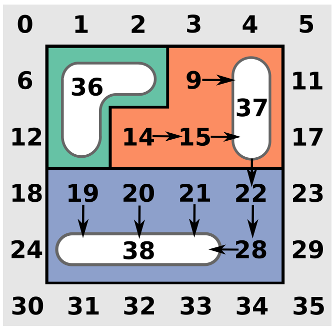
Example 9. Our connectivity matrix is based on the flow directions in the middle figure in Figure 3.10. We want to expand *conn_mat* by the number of traps t in the



(a) Landscape elevations.



(b) Flow directions.



(c) Final connections in *conn_mat*.

Figure 3.10: The left figure shows the elevations of the landscape, while the middle figure shows the corresponding flow directions which are needed to create the cell connectivity matrix. In the figures the colored areas show the watersheds, the white areas the traps, and the flow directions are indicated by black arrows. The right figure shows the flow directions after each trap has been merged to a single trap node. All incoming and outgoing flow from traps have been rerouted using the new trap node indices.

landscape, which in our case is three. This means the size of *conn_mat* is increased to 39x39. In our example the three trap nodes get the indices 36, 37 and 38. We can see the new indexing of the traps in the right figure.

Now that *conn_mat* is expanded, we will add the connections in and out of the trap nodes, remove the connections to the old traps, and also remove the flow to the boundary. In Example 10 we show how we obtain the final connections in the connectivity matrix, shown in the right figure in Figure 3.10.

Example 10. We will use the steps after *conn_mat* has been expanded in Algorithm 3 to obtain the complete connectivity matrix. First we use the spill pairs to add all outflow from the new trap nodes, i.e., the connections (36, 18), (37, 22) and (38, 31). This ensures the flow out of traps. Next we need to add the upslope connections to the trap nodes, i.e., the flow to the former trap cells. All connections that were previously routed to different cells in a trap will now be routed to the same trap node. One example is the connections to cell 37 — after we have added (9, 37) and (15, 37), we remove the former connections (9, 10) and (15, 16). We also do this for the other traps. When we have done this for all traps we get the connections: (9, 37), (14, 15), (15, 37), (19, 38), (20, 38), (21, 38), (22, 28), (28, 38), (36, 18), (37, 22) and

(38, 31). The last step is to remove all connections going to the boundary. This means that (36, 18) and (38, 31) are removed. The resulting connectivity matrix enables us to delineate the watershed of any cell in the landscape. The final connections in the connectivity matrix *conn_mat* is shown in the right figure in Figure 3.10.

We have now created a connectivity matrix where all cells are accounted for. This means that we can use *conn_mat* to calculate the watershed of any cell in the landscape. In the following, we will base our selection of a watershed outlet on the flow accumulation in the landscape. This is because a point in the landscape with a high flow accumulation will have a large watershed. In the next section we calculate the flow accumulation.

3.7 Calculate flow accumulation

Now that we have calculated the connectivity matrix, we are able to delineate the watershed of any point in our DEM. To select a point in the landscape we calculate the flow accumulation, because it shows the locations of all major rivers and lakes. Usually we are interested in the hydrograph from rivers with a large watershed, and these are now possible to find. Below we outline an algorithm to calculate the flow accumulation in the entire landscape.

Algorithm 4 The CalculateFlowAccumulation algorithm takes the connectivity matrix *conn_mat* as input and calculates the flow accumulation in the landscape.

```
1:  function CalculateFlowAccumulation(conn_mat)
2:      start_cells ← GetStartCells(conn_mat, traps)
3:      flow_acc ← AssignFlowAccToStartCells(traps, start_cells)
4:      current_cells ← GetDownslopeCellsOfStartCells(conn_mat, start_cells)
5:      while current_cells do
6:          cell_indices ← AssignFlowAccTo(flow_acc, conn_mat, current_cells)
7:          values ← GetFlowAccValues(conn_mat, flow_acc, cell_indices)
8:          flow_acc ← UpdateFlowAcc(flow_acc, cell_indices, values)
9:          current_cells ← UpdateCurrentCells(cell_indices, current_cells)
10:     end while
11:     flow_acc ← SetFlowAccToAllCellsInTraps(flow_acc, traps)
12:     return flow_acc
13: end function
```

To explain the steps in Algorithm 4 we will use the example landscape in Figure 3.10. The first step in the algorithm is to assign flow accumulation to start cells. A landscape will always have local maxima cells without any upslope cells, and these will not have any flow accumulation besides their own contribution. In our algorithm we use these cells as our starting cells, and they are the first cells we assign flow accumulations. To calculate the start nodes we remove all boundary cells, cells with an upslope, and the original trap cells. The cell indices that remain afterwards will be our start cells. We show the process in Example 11.

Example 11. The right figure in Figure 3.10 shows all connections in the landscape, i.e., the flow for each cell in the terrain. After we remove the boundary cells, and the cells with an upslope, we are only left with the cells with indices 9, 14, 19, 20, 21 and 36. These will the *start_cells* in Algorithm 4.

The flow accumulation algorithm is an iterative algorithm where we only assign flow accumulation to a cell if all its upslope cells have already been assigned a flow accumulation. The reasoning behind this is e.g., a point where two rivers meet. Until the upslope area of both rivers are known, we cannot assign the point a flow accumulation. The first cells we assign flow accumulations are the start cells. If the start cell is a single cell, its contributing area will be one. If it is a trap cell, it will be assigned a flow accumulation based on the number of cells the trap cell represents. Afterwards, we add their downslope cells to the variable called *current_cells* in Algorithm 4. This variable keeps track of cells that have been reached by either one or several upslope cells. If there are cells in *current_cells* with all their upslope cells' flow accumulation assigned, they are added to *cell_indices*, and we find their upslope areas (which also include the cells' own sizes) *values*. Afterwards we remove the cells from *current_cells*, but add their downslope cells. Then the flow accumulation *flow_acc* is updated with the new entries. We do this until *current_cells* is empty, which means that all cells have been assigned a flow accumulation. We will show an example of the algorithm in Example 12, where we will use f as a shorthand form of the flow accumulation value.

Example 12. In Example 11 we found the start cells, which we immediately can assign f-values. Every cell in *start_cells* with an index below 36 is assigned one, and cell 36 is assigned three, because it has a size of three cells. We then add their downslope cells to *current_cells*: 15, 37 and 38. Out of these three cells only one of them have all their upslope cells' f-values defined: cell 15, which only relies on the flow from cell 14. We can not assign an f-value to 37 because one of its upslope cells, cell 15, do not have an f-value yet. In the next iteration, this is no longer true, and cell 15 will be assigned an f-value of 2. Because its downslope is already in *current_cells*, it is not necessary to add it again.

In the next iteration the cells in *current_cells* are 37 and 38. Now that both upslope cells of 37 have an f-value, cell 37 is also assigned one, in this case a value of 5. The downslope of 37 is then added, so the updated *current_cells* consists of cells 22 and 38. Cell 22 is assigned an f-value of 6, and 28 is added to *current_cells*. 28 is assigned a value of 7, and finally only cell 38 remains in *current_cells*. Because its four upslope cells now have a defined flow accumulation, it is assigned a flow accumulation of 13.

The last step in the algorithm is to assign f-values to all original trap cells in the landscape. This means that e.g., cells 25, 26 and 27 receive an f-value of 13. The final resulting flow accumulation is shown in Figure 3.11. The darker the blue, the larger the flow accumulation.

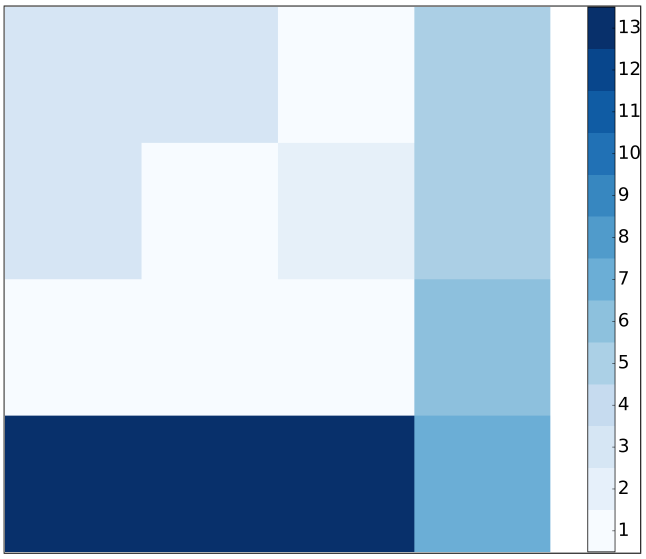


Figure 3.11: The flow accumulation of the example landscape. The trap in the lower left corner has the darkest color. It accumulates flow from all 13 cells in the watershed, as we recall that the three cells in the top left corner belong to a different watershed.

In the Results section a flow accumulation plot of the entire Tyrifjorden landscape will be presented.

3.8 Get watershed of outlet

After we have calculated the flow accumulation in the landscape, we can visually decide on a cell to be our outlet. A typical choice is a cell in a river which historically often floods, and causes damage to houses and infrastructure. These locations are often close to areas used by people, which is why the damage potential is great. Once we have the coordinates of the cell, we will use the connectivity matrix from Section 3.6 to obtain the watershed of the outlet.

Let us call the watershed of the outlet w_o. The connectivity matrix gives us the upslope cells of any given cell, so the algorithm starts by finding the upslope cells of the outlet, which are added to w_o. Sometimes it happens that an outlet point has no upslope cells from the connectivity matrix. This happens in two cases: 1. The outlet point actually has no upslope cells, which yields a w_o with only one cell, and 2. The outlet point is in a trap. We recall from Section 3.6 that only the trap cells representing the traps have any connections in the connectivity matrix. The individual cells in the traps do not have any connections, which explains why the cell has no upslope cells. In that case a mapping between cells and traps is used to locate the trap. Once we know the trap cell index, we can check if it has any upslope cells. After all upslope cells have been found, the trap cells are mapped to their traps. Then w_o has been found.

We will now shown an example of how we get the watershed of an outlet. The left figure in Figure 3.12 will be our landscape. It is the same figure as the right figure in Figure 3.10, but to make it convenient for the reader, we present it again here. As outlet we will use cell 26, which is the middle cell of the trap that trap node 38 represents. From the accumulated flow plot in Figure 3.11 we know that its flow accumulation is high.

Example 13. In this example we will calculate the watershed of outlet cell 26 in the left figure in Figure 3.12. We will call the watershed for w_{26}. In the first step we check if the outlet cell has any upslope cells. Because it is a part of a trap, there are no incoming or outgoing connections in the connectivity matrix. We use a mapping between 1D-indices and the traps to locate cell 26 in trap node 38, which gives us a temporary $w_{26} = \{38\}$. If we include cell 38's upslope cells we obtain the watershed $w_{26} = \{19, 20, 21, 28, 38\}$. We iteratively add new upslope cells of the upslope cells that were recently added, i.e., {19, 20, 21, 28}. Their only upslope cell is cell 22, which is added to w_{26}. We proceed by adding cell 22's upslope cells, which is only cell 37. Next, cells 9 and 15 are added to w_{26}, and finally their upslope cells, which is only cell 14. This leaves us with $w_{26} = \{9, 14, 15, 19, 20, 21, 22, 28, 37, 38\}$. Now the trap cells must be mapped back to its traps. Cell 37 becomes {10, 16} and cell 38 becomes {25, 26, 27}. The final delineated watershed for cell 26 is $w_{26} = \{9, 10, 14, 15, 16, 19, 20, 21, 22, 25, 26, 27, 28\}$, which we show in the right figure in Figure 3.12.

In this chapter we have created an algorithm, Algorithm 1, that is able to delineate the watershed of any location in a DEM. We will use the obtained watershed, $watershed\_of\_outlet$, in the following chapters. In the next chapter we will calculate the travel times for all cells in the watershed, which in turn will be used for the hydrograph calculations.

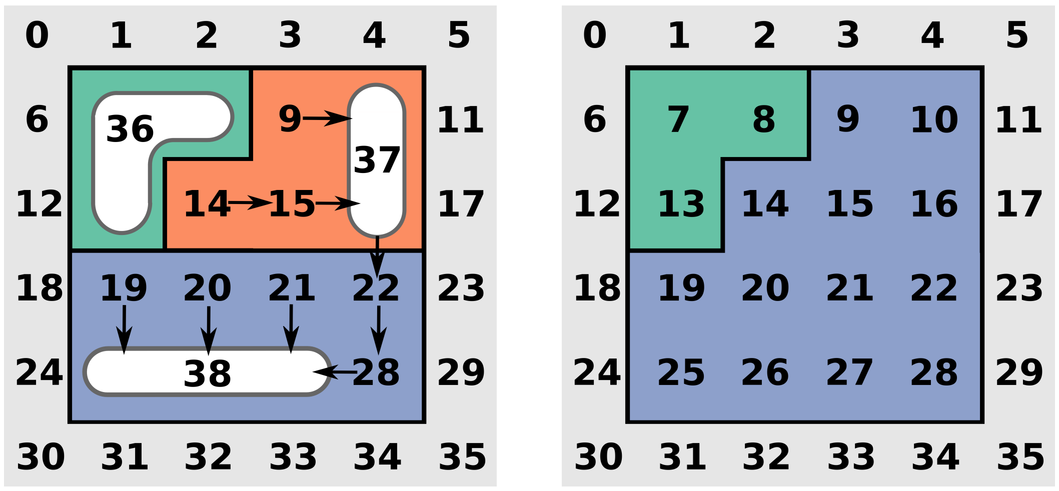


Figure 3.12: The left figure is the same as the right figure in Figure 3.10, but we show it again so it is easier to follow the algorithm. The figure shows the three different watersheds, their respective traps, spill pairs and flow directions. It illustrates the information that is used to obtain the watershed of cell 26: w_{26}. The resulting watershed is shown in the right figure, and is the entire blue region. This means that all water that falls within the blue region eventually ends up at the watershed outlet.

4 Travel time estimation

In Chapter 3 we created and implemented an algorithm to delineate the watershed of any location in the landscape. We now move on to the next part, which is to estimate the runoff after a rainfall and visualize it in a hydrograph. In order to do this, we need to know the time it takes water to travel from any location in the watershed to the watershed outlet. In the time-area method this is often referred to as the travel time [7, 30], and in this chapter we will obtain the travel time based on the so-called time-of-flight equation [21].

Chow et al. [6] defines the travel time along a path from an arbitrary point β in the watershed to the watershed outlet as

$$t(L) = \int_0^L \frac{\mathrm{d}l}{u(l)}, \tag{4.1}$$

where L is the path distance, $u(l)$ the particle speed along the path, and t the travel time. If we divide the path into segments, and assume a constant speed u_i along the path segment l_i, the equation can be discretized into

$$t(L) = \sum_{i=1}^{I} \frac{l_i}{u_i}. \tag{4.2}$$

The u_i-values usually come from formulas that estimate the speed for overland flow and channel flow (Manning's Equation), but to make it simple, tables with different slopes and terrain types are often used. This approach is similar to the segmental velocity approach that we mentioned in Chapter 2.4.

Another method to estimate travel times, is to parametrize the streamline going from point β to the watershed outlet, by using the *time-of-flight*. Lie [21, pp. 128-129] outlines this method, and we will summarize the main points here. A streamline from point β to the outlet, can be parametrized as $\vec{x}(r)$, where r is the arc length from β to the outlet. We can also use the time-of-flight as parametrization, which we will do here. Given a velocity field $\vec{v}$, the time-of-flight is defined as

$$T(r) = \int_0^r \frac{\phi(\vec{x}(s))}{\left|\vec{v}(\vec{x}(s))\right|} \mathrm{d}s, \tag{4.3}$$

where $T(r)$ represents the time it takes water to travel from β along the streamline to the outlet. We see that Equation (4.1) is equal to Equation (4.3) if $u = |\vec{v}|/\phi$.

To explain the ϕ-parameter in Equation (4.3), we assume a water flow along a streamline from position A to the watershed outlet B. The streamline can be discretized into cells that represent each path segment, shown in Figure 4.1. In the model we will use, the water must completely fill up a cell before the flow can continue to the next cell. To represent the fraction of volume that can be filled in each cell, we set the ϕ-parameter for each cell to a value between zero and one. If ϕ is low, the cell is quickly filled, and the particle speed $|\vec{v}|/\phi$ in the cell increases.

We will use $\phi = \epsilon$ for the trap cells, where ϵ is a small value. This will increase the particle speed in traps, and reduce the time it takes to cross them. We base this on the assumption that traps in the landscape are completely filled with water, so the water that enters, immediately flows out at the other side. Another way to think of ϕ, is that ϕ represents the fraction of the cell volume that is soil. If we assume a constant rate of flow (e.g., from precipitation), the water will flow faster in areas with less soil. Neither a lake, nor a rock face, contains any soil, and because our previous assumption only justifies high speed for lakes and ponds, we assume that all non-trap cells contain soil. Our interpretation of ϕ is problematic if we account for saturation, as a fully saturated cell will act like a trap cell completely filled with water. Hence, our model will not account for the degree of saturation.

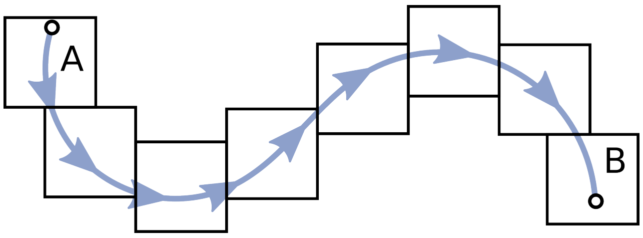


Figure 4.1: A series of cells represent a discretized streamline from point A to point B in the landscape. In the model we use, the water needs to fill each cell before it can continue to the next. The time it takes to reach point B will then depend on how much water that can be filled in each cell, which is given by ϕ.

It is possible to write Equation (4.3) as

$$\frac{dT}{dr} = \frac{\phi}{|\vec{v}|} = \nabla T \cdot \frac{\vec{v}}{|\vec{v}|}, \tag{4.4}$$

which we can write into the so-called time-of-flight equation,

$$\vec{v} \cdot \nabla T = \phi. \tag{4.5}$$

To use the equation, we need to approximate the velocity field $\vec{v}$. If we assume the flow in the watershed is a creeping flow, it enables us to use Darcy's Law for single-phase flow in porous media [21, p. 16],

$$\vec{v} = -\kappa (\nabla p + \rho g \nabla z), \tag{4.6}$$

where we obtain the so-called Darcy velocity. In the equation, κ is the permeability, p the pressure, g the gravitational constant, ρ the mass density of the soil, and z the elevation of the landscape. In a more complex model we would also include the equation for the conservation of mass.

We will use a simplified version of Equation 4.6, where we set $\nabla p = 0$. This is based on the assumption that the atmospheric variations in the landscape are small, if the differences in elevations are small. We also assume a constant pressure in the upper soil layer. To further simplify, we set the term ρg to one.

A velocity field $\vec{v}$ based on topography alone, is not sufficient to realistically represent velocities in the watershed. A simple example that shows this, is water flow down two identical hills (same slope), but with a different surface: one with sand, the other with granite. Because a granite surface has a lower friction, the water will flow at a higher velocity down this hill, even though the two hills have the same shape. These heterogeneous properties can be included in κ, which we can write as

$$\kappa = f(vegetation, snow, \ldots, soil\ type), \tag{4.7}$$

where f is a function that determines the influence each property has on the velocity $\vec{v}$. Some of these properties might also affect ϕ.

The aforementioned simplifications result in an equation that relates the speed to the slope in the terrain, and is scaled by a function of the heterogeneous properties that affect the water flow speed,

$$\vec{v} = -\kappa \nabla h. \tag{4.8}$$

If the model were to be calibrated, κ has to be determined based on how well the runoff matches historical data. Because the model is based on physical parameters, it will hopefully also work for ungauged watersheds. In our initial rainfall-runoff model, we will set $\kappa = 1$ in our implementations. When it comes to the simulation of rivers, our model will not be ideal, because we have assumed flow in a porous media.

In our calculations we will assume that the time-of-flights are constant. This is in general not true, as the flow rate and flow depth will change flow velocities in a nonlinear way [2], but if we update our velocity field $\vec{v}$ over the course of the simulation, we can change the time-of-flights, so that they are not constant.

The time-of-flight equation is popular in simulations of oil reservoirs [21, 17], because the time-of-flight T represents the time it takes a particle to travel from the nearest injector well to the production well. In our scenario we are not only interested in the time from one point to the watershed outlet, but instead the time it takes from the watershed outlet to every point in the watershed. We can find this by flipping the fluxes in Equation (4.5).

To solve the time-of-flight equation we will use a solver from the diagnostics module in the Matlab Reservoir Simulation Toolbox (MRST) [21]. It is a package tailored for the oil industry, so its functionality is mostly related to the simulation of reservoirs and subsurface flow, but because we model our flow as creeping flow, we can use it for our simulations. In the next section we will discretize Equation (4.5) using the so-called finite-volume method.

4.1 Finite-volume discretization

In order to solve Equation (4.5), we need to discretize the equation; a *finite-volume method* [35] is used to accomplish this. In the finite-volume method, a so-called control volume (a small volume) is constructed around each node point in the mesh. We will use a cell oriented arrangement of nodes, where each grid point is located in the center of a control volume. Both sides of the equation are integrated over a cell volume, and the divergence theorem is used to convert it into a surface integral. In the discretization, we will use an upwind scheme, as e.g., a centered scheme is unstable [21, p. 140].

The first operation we do is to integrate equation (4.5) over an arbitrary control volume C_i. We will think of the control volume as a cell with index i, which is the structure we will use in MRST.

$$\int_{C_i} \vec{v} \cdot \nabla T \;\, dV = \int_{C_i} \phi \;\, dV \tag{4.9}$$

The product of a scalar-valued function and a vector field can be used to rewrite (4.9),

$$\int_{C_i} \nabla \cdot (T\vec{v}) - T(\nabla \cdot \vec{v}) \;\, dV = \int_{C_i} \phi \;\, dV.$$

Next, we use the divergence theorem and obtain

$$\underbrace{\int_{\partial C_i} T(\vec{v} \cdot \vec{n}) \;\, dS}_{a} + \underbrace{\int_{C_i} T(\nabla \cdot \vec{v}) \;\, dV}_{b} = \int_{C_i} \phi \;\, dV. \tag{4.10}$$

We need to discretize both a and b in (4.10), and we start with a:

$$\int_{\partial C_i} T(\vec{v} \cdot \vec{n}) \;\, dS = \sum_{j \in U(i)} T_j f_{ij} + T_i \sum_{j \in D(i)} f_{ij}. \tag{4.11}$$

In Equation (4.11) we separate the boundary faces based on whether the flow over the face is downstream or upstream from control volume C_i. The downstream indices are $D(i)$, and the upstream indices are $U(i)$. The flux over the face between cells i and j is f_{ij}.

To discretize b in (4.10) we again apply the divergence theorem and get

$$\begin{aligned} \int_{C_i} T(\nabla \cdot \vec{v}) \;\, dV &= T_i \int_{C_i} \nabla \cdot \vec{v} \;\, dV \\ &= T_i \left(\int_{\partial C_i} \vec{v} \cdot \vec{n} \;\, dS \right) = T_i \left(\sum_{j \in D(i)} f_{ij} + \sum_{j \in U(i)} f_{ij} \right). \end{aligned}$$

We insert a and b back into (4.10) and get

$$\left(\sum_{j\in U(i)} T_j f_{ij} + T_i \sum_{j\in D(i)} f_{ij}\right) - T_i\left(\sum_{j\in D(i)} f_{ij} + \sum_{j\in U(i)} f_{ij}\right) = \phi_i\,|C_i|,$$

which we can further simplify to

$$\sum_{j\in U(i)} T_j f_{ij} - T_i \sum_{j\in U(i)} f_{ij} = \phi_i\,|C_i|,$$

and rearrange to finally obtain an expression for cell i's time-of-flight:

$$T_i = \frac{\sum_{j\in U(i)} T_j f_{ij} - \phi_i\,|C_i|}{\sum_{j\in U(i)} f_{ij}} \tag{4.12}$$

Now hat we have looked at the theory, we will turn our attention to the implementation we need to do to be able to run the solver in MRST. We recall that for an arbitrary cell i, T_i is the average time a particle needs to travel from cell i to the watershed outlet. It is an average because there are usually more than one flow path leading from cell i to the outlet.

The coordinates of the watershed we acquired in Chapter 3 is not the only thing we need in our calculations of the time-of-flight. We also need the elevations in our depressionless landscape dE, the flow directions, and traps and spill pairs. Finally, we need information about where the outlet is. In the following sections we show examples where we explain the steps in Algorithm 5.

Algorithm 5 The CalculateTimeOfFlight algorithm calculates the travel times from all cells in the watershed to the watershed outlet.

```
1: function CalculateTimeOfFlight(
       $watershed\_of\_outlet,\ dE,\ traps,\ flow\_directions,\ spill\_pairs,\ outlet$)
2:     $G \leftarrow$ TransformWatershedToGrid($watershed\_of\_outlet$)
3:     $CG \leftarrow$ OptimizeGridStructure($CG,\ traps$)
4:     $fluxes \leftarrow$ CalculateFaceFluxes($CG,\ dE,\ traps,$
         $flow\_directions,\ spill\_pairs$)
5:     $phi \leftarrow$ SetPhiValues($CG$)
6:     $time\_of\_flight \leftarrow$ CalculateTimeOfFlight($CG,\ fluxes,\ phi,\ outlet$)
7:     return $time\_of\_flight$
8: end function
```

Because MRST's grid structure and the functionality it provides is a big part of our calculations, we will start with a brief introduction to it in the next section.

4.2 MRST grid structure

In order to use the time-of-flight solver in MRST, we need to transform our delineated watershed into the grid structure that MRST uses. The blue watershed from the right figure in Figure 3.12 will be used as an example, and we will call it w_b.

In Figure 4.2 we have transformed w_b into the new grid structure, where the coordinates of the watershed have become the centroids of the cells in the grid. In this regular structured grid, the corners of each cell are the *nodes*, while the lines between the nodes are the faces. The indices of the cells and faces are shown in the figure, and positioned in the cell centroids and face centroids, respectively. Because the watershed has been delineated from a DEM of resolution 10 meters, each cell will have a size of 100 square meters, with side lengths of 10 meters.

In total the grid has 13 cells and 34 faces, and because it is a regular structured grid each cell will have four faces, of which many are shared among two cells. We will call this grid with all its cells, faces and nodes for G_b.

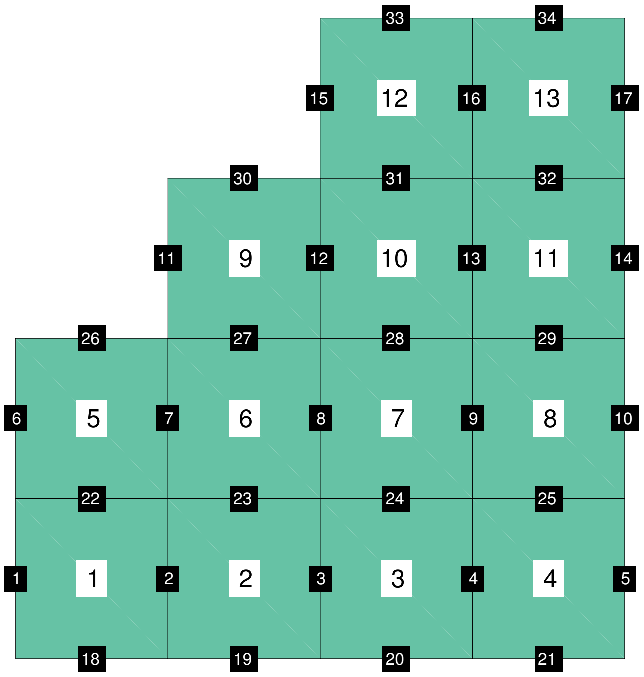


Figure 4.2: The watershed previously referred to as w_{26} from Figure 3.12 transformed into an MRST grid. The cell indices and face indices are positioned in the cell centroids and face centroids, respectively.

This introduction, although brief, will hopefully be enough to better understand the operations we do in the steps to obtain the time-of-flights (travel times).

4.3 Create grid

The first step we need to do in Algorithm 5 is to transform the delineated watershed w into the grid structure provided by MRST. We already showed one example of this in Section 4.2 when we obtained the grid structure G_b for the watershed w_b. We will come back to w_b and G_b a little later, and for now focus on the general case.

In an effort to reduce the number of cells in the grid G, we combine the cells in each trap into a single cell. This optimization will improve running time as the number of cells and faces are reduced. If w has any traps, we modify G to obtain a coarse version of the grid that we call CG. In the case where w does not have any traps, CG will simply equal G. After each trap cell has been combined, only the boundary faces of each trap remains, although their face lengths might have been altered.

We return to G_b to see how it changes when we coarsen it. From Section 3.8 we recall that there are two traps in w_b, so we expect the two areas to be combined into two cells. In Figure 4.3 we show the coarse grid CG_b, and as expected, the cells in the two traps (colored blue) have been merged. We observe that the new grid is both unstructured and irregular, and has a new ordering of face indices and cell indices. The trap cells have been assigned the highest indices on purpose, as it makes some implementation details easier. Compared to the old grid G_b, the number of cells and faces have now decreased from 13 to 10, and 34 to 28, respectively. Because of these changes, the cells are not uniform in size anymore.

4.4 Calculate face fluxes

After we have transformed the watershed to the new framework, we need to calculate the fluxes over the faces based on the velocity field $\vec{v}$ from Equation (4.8). Because $\kappa = 1$, our initial velocity field do not account for heterogeneous properties in the landscape.

We will now look at the flux over an arbitrary face F, with cells i and j as its neighbors. The flux, which is denoted f_{ij}, is an average of the flux contributions from cells i and j, as only these two cells affect the flow over the face. When we move from a framework that allows flow in eight directions, to one that only allows flow in four (the four faces), we need to project the flow on the two adjacent faces for diagonal flow. The flux over the face can be written as

$$\begin{aligned} f_{ij} &= \int_F \left(\frac{\vec{v_{ij}} \cdot \vec{n_{ij}}}{\|\vec{n_{ij}}\|^2} \vec{n_{ij}} \right) \cdot \vec{n_{ij}}\, dl \\ &= \int_F \vec{v_{ij}} \cdot \vec{n_{ij}}\, dl \\ &\approx (\vec{v_{ij}} \cdot \vec{n_{ij}}) \|\vec{n_{ij}}\| \end{aligned} \tag{4.13}$$

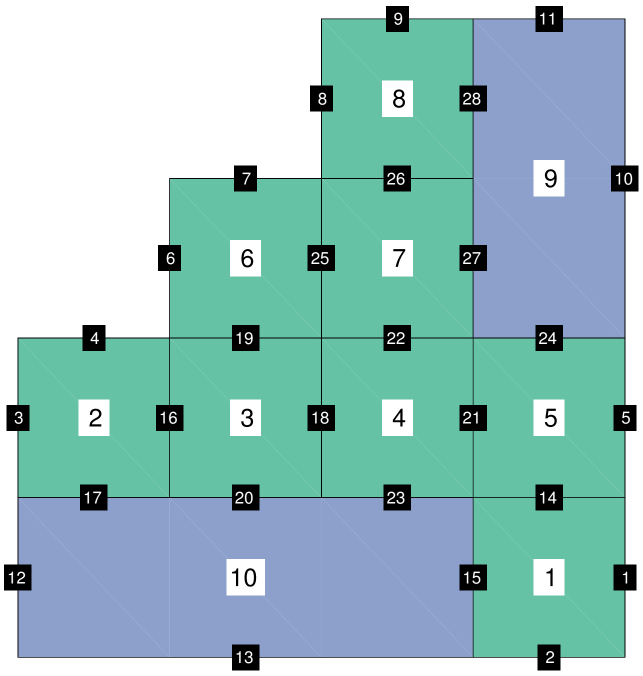


Figure 4.3: After the grid is coarsened, we are left with only ten cells. Two of them are colored blue, and are the traps in the watershed. We can see that the cells and faces have been renumbered for CG_b.

where $\vec{v_{ij}}$ is the average velocity in the face (based on the flow in the two adjacent neighbors)[1], $\vec{n_{ij}}$ is the face normal of F, and $|F| = \|\vec{n_{ij}}\|$ is the length of the face. If f_{ij} is positive, the flux is directed in the same direction as $\vec{n_{ij}}$. Otherwise, the flow moves in the opposite direction.

Estimate velocities

The velocity estimate we derived in the introduction to this chapter, is given by Equation (4.8), where we use $\kappa = 1$. We will use a discretized version of this equation, based on the slope over the faces and the flow directions we calculated in Chapter 3. We will define the average flow velocity over a face F as

$$\vec{v_{ij}} = \frac{1}{2}\left(\alpha_i^F \vec{d_i} + \alpha_j^F \vec{d_j}\right), \tag{4.14}$$

where $\vec{d_i}$ and $\vec{d_j}$ are the flow directions of face F's two neighboring cells[2], which we will refer to as the *face flow directions* of F. We scale the face flow directions by the

[1] A face at the watershed boundary only has one neighbor cell, but we do not calculate the fluxes for these.

[2] In some cases we will use another flow direction, e.g., for faces in trap cells.

α-values α_i^F and α_j^F, which for an arbitrary cell c and face F is defined as

$$\alpha_c^F = \begin{cases} 0, & \text{if } \vec{d_c} \cdot \vec{n_{ij}} = 0 \\ C, & \text{if } \frac{\Delta z}{h} \leq 0 \\ \frac{\Delta z}{h}, & \text{if } \frac{\Delta z}{h} > 0. \end{cases} \tag{4.15}$$

This means that α_c^F is either set to 0, a constant C, or $\Delta z/h$, which represents the slope between cells i and j in the direction of flow. The slope is dependent on the distance between the grid points in the DEM, which in our case is $h = 10$ m. Each of these cases require some explanation, and we start with the first case. If the flow direction of the cell and the face normal of F are perpendicular to each other, there will be no flow over the face, so α_c^F is set to zero. For the two other cases, we calculate $\Delta z/h$ based on whether the flow over the face is an inflow to c, or an outflow. This gives us the following expression for Δz:

$$\Delta z = \begin{cases} z_{in} - z_c, & \text{if flow towards } c \text{ (inflow)} \\ z_c - z_{out}, & \text{if flow away from } c \text{ (outflow).} \end{cases} \tag{4.16}$$

Thus Δz is calculated as the difference in elevation in the direction of flow, where z_{in} is the cell the flow came from, and z_{out} is the cell the flow moves to.

The second case in Equation 4.15 happens if a previously diagonal flow now flows towards a cell of higher elevation (one of its projected faces). Regardless of this, the flow has to continue, so α_c^F is set to a constant $C > 0$; in our calculations we have used a $C = 0.1$. The reason why we force the flow in an uphill direction, is that sometimes both faces adjacent to the diagonal have uphill flow. This can potentially stop the flow, and make it impossible for water in some areas to flow to the outlet in the delineated watershed. Even though MRST does not allow for eight flow directions, we still want to maintain the watershed we delineated based on eight flow directions.

Lastly, if the flow is downhill, we use $\Delta z/h$. Because diagonal flow will get different α-values based on the elevations of the adjacent faces' neighbors', the resulting flow direction will no longer be in its original flow direction (diagonally), but this is not something we will dwell upon.

In Example 14 we elaborate on the assignment of face flow directions, which might be useful for Example 15 where we present an example of how α-values are calculated.

Example 14. To calculate average velocity over a face (Equation (4.14)), we need its face flow directions. In the left figure in Figure 4.4 we see an example with four cells A, B, C and D, where we set the face flow directions for the faces of cell A, based on the flow in the cell. The four arrows show the assigned face flow directions, and these will represent the direction of the flux contribution from cell A. This is a process that is done for all the cells, as two face flow directions are needed for each face. For us to calculate the average velocity over e.g., the face between cells A and

C, we will also need to assign a face flow direction to the face based on the flow in cell C.

Example 15. In Figure 4.4 we illustrate how we can calculate the α-values for two of the faces that belong to A. We already talked about the assignment of face flow directions in Example 14, and we will use the ones assigned by cell A in the left figure to illustrate how we calculate the α-values for the northern face (F_N) and the eastern face (F_E). The elevations in the right figure are necessary for this calculation.

Because cell A has a flow direction towards the northeast (from the D8 Algorithm), the flow is projected on the two adjacent faces F_N and F_E. This will create a situation where the flux contribution from the flow in A is uphill to the north, and downhill to the east. When we calculate $\alpha_A^{F_N}$ we know the flux contribution over face F_N is an outflow, so we get $\Delta z = -5$ from Equation (4.16). This means the second case in Equation (4.15) applies, and we set $\alpha_A^{F_N} = C$.

For the eastwards face, the third case will apply, as we get a positive $\Delta z = 5$ from Equation (4.16). This yields an $\alpha_A^{F_E} = 0.5$. If we were to calculate the average velocities $\vec{v_{AC}}$ and $\vec{v_{AB}}$, we would also need the face flow directions for the faces F_N and F_E set based on the flow in cells C and B, respectively, as well as their α-values.

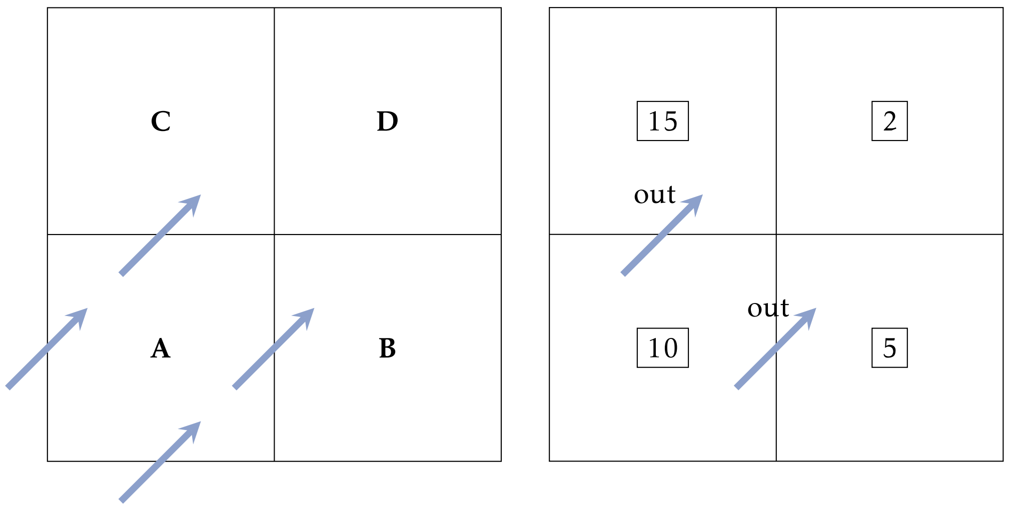


Figure 4.4: The left figure shows the assignment of face flow directions for cell A's faces based on the flow in A. In the right figure we show the elevations of the landscape and the information we need to calculate the α-values for two of its faces. In this case, the elevation difference to the north is negative, so the face's α-value is set to a constant C, while the α-value of the eastern face is set to $\Delta z/h$.

An alternative approach to estimate the velocity field $\vec{v}$, is to interpolate the flow directions and the elevations in the landscape, where we use the resulting flow directions in the face centroids. To account for topography, we can calculate the elevation difference between the face centroids and the closest cells in their new flow directions. This method might be more accurate than the one we have implemented.

Example of flux calculation

In Example 16 we show how we calculate the flux over the face between cells 1 and 5 in Figure 4.3. All relevation information from CG_b is shown in Figure 4.5. This includes the face normal of the face, and the flow directions of the adjacent cells.

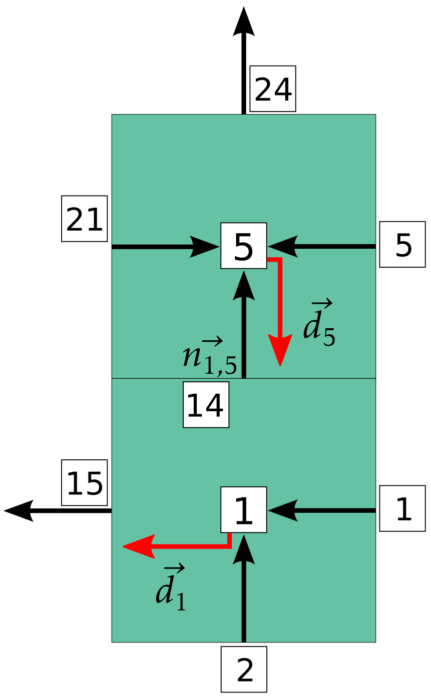


Figure 4.5: The face normals and flow directions of cells 1 and 5 from Figure 4.3. The flow directions are colored red, while the face normals are colored black. To compute the flux over the face with index 14, both the flow directions of cells 1 and 5, and the face normal of face 14 are required.

Example 16. We want to calculate flux $f_{1,5}$ for the face between cells 1 and 5 in Figure 4.5. Before we can use Equation (4.13), we need to calculate $\vec{v_{1,5}}$. In the calculations we will also need the elevations of the two cells, which are given as $z_1 = 4.5$ and $z_5 = 7.0$. The face normal is $\vec{n_{1,5}} = [0,1]$, and the flow directions are $\vec{d_1} = [-1,0]$ and $\vec{d_5} = [0,-1]$.

To calculate $\vec{v_{1,5}}$, we need to compute $\alpha_1^{F_{1,5}}$ and $\alpha_5^{F_{1,5}}$. For simplicity we will use α_1 and α_5. Because the flow direction $\vec{d_1}$ and the face normal $\vec{n_{1,5}}$ are perpendicular to each other, α_1 is simply 0. For α_5 the flow over the face is an outflow, so $\Delta z = 7.0 - 4.5 = 2.5$, which yields $\alpha_5 = 2.5/10$.

Thus, the average velocity over the face will be,

$$\begin{aligned} \vec{v_{1,5}} &= \frac{1}{2}\left(\alpha_1 \vec{d_1} + \alpha_5 \vec{d_5}\right) \\ &= \frac{1}{2}\left(0 \cdot [-1,0] + \frac{2.5}{10} \cdot [0,-1]\right) \\ &= 0.125 \cdot [0,-1] \ \text{ m/s.} \end{aligned}$$

If we insert the expressions for $\vec{v_{1,5}}$ and $\vec{n_{1,5}}$ into Equation (4.13), we get

$$\begin{aligned} f_{1,5} &= (\vec{v_{ij}} \cdot \vec{n_{ij}})\|\vec{n_{ij}}\| \\ &= 0.125 \cdot [0,-1] \cdot [0,1] \cdot 10 \\ &= -1.25 \ \text{ m}^2/\text{s} \end{aligned}$$

The negative sign means that the flow moves in the opposite direction of $\vec{n_{1,5}}$, with a flux of 1.25 m^2/s.

Implementation details

The computations of the fluxes are without a doubt the most challenging step in Algorithm 5, when we compute the time-of-flights. Both the transition from eight to four flow directions, and unexpected behavior in the coarse grid structure, created some difficulties for us. At a later time, it has come to our attention that some of the problems we experienced with the coarse grid could have been avoided, which we will talk more about later. For the rest of this section, we will talk about how we calculated the fluxes, and which problems we faced. An outline of the algorithm is shown in Algorithm 6.

In the implementation of Algorithm 6 we experienced that the majority of the effort is to get appropriate face normals and face flow directions for all faces in the grid. In some cases we have to alter face normals or face flow directions to make flow continue. The consequence if they are not altered, is that there might be faces where the flux is so low (or zero) that it effectively stops all flow from its upslope areas. Hence, the main goal in our flux calculations is to obtain face fluxes which yield time-of-flights for all cells in the watershed. This is necessary for the hydrograph creation, as precipitation must be able to flow from anywhere in the watershed to the outlet.

In Algorithm 6 we start by setting all face normals equal to the face normals in the grid object *CG*. It is only in some cases that they have to be changed to get the desired flow over a face. Next, we assign face flow directions to all non-trap cells, which are equal to their respective cells' flow directions. Like for the face normals, this usually works well, but sometimes they must be changed to get a continuous flow in the watershed.

Algorithm 6 The CalculateFaceFluxes algorithm calculates the fluxes over all faces in the coarse grid, which are used in the time-of-flight computations.

```
1: function CalculateFaceFluxes($CG$, $flow\_directions$, $dE$, $traps$, $spill\_pairs$)
2:     $face\_normals \leftarrow$ SetInitialFaceNormals($CG$)
3:     $face\_flow\_directions \leftarrow$
         SetInitialFaceFlowDirections($CG$, $flow\_directions$)
4:     $face\_normals$, $face\_flow\_directions \leftarrow$
         EnsureFlowInCornerCases($CG$, $face\_normals$, $face\_flow\_directions$)
5:     $face\_flow\_directions \leftarrow$
         EnsureFlowFromTraps($CG$, $flow\_directions$, $traps$, $spill\_pairs$)
6:     $flux \leftarrow$ CalculateFluxes($CG$, $face\_normals$, $face\_flow\_directions$, $dE$)
7:     $flux \leftarrow$ AverageFluxes($CG$, $flux$)
8:     return $flux$
9: end function
```

Unlike for the non-trap cells, where we can say that the cell's flow direction represents the flow over all its faces, we can not say the same for a trap cell. This is because the trap cell's flow direction only represents the direction of the flow at its spill point. Thus, the face flow directions over the trap faces must be defined in another way. Because the traps are essentially lakes, dams and ponds in the terrain, the neighbor cells usually have a flow in the direction of the trap. This motivates us to assign a face flow direction that is equal to the neighbor cell's flow direction (the neighbor that is not in the trap). The initial assignments of face flow directions in an L-shaped trap is shown in Example 17. The face that is closest to the spill point in the trap's flow direction will be referred to as the *spill face*, and we will give special consideration to it later, as it is very important that the trap has an outflow.

Example 17. In Figure 4.6 we show an example of how face flow directions are assigned to the faces in an L-shaped trap. The left figure shows the flow directions of the cells that surround the trap, and we will use these for the assignment of face flow directions.

In the right figure we use the neighbors' flow directions to determine the face flow directions for the trap faces. In most cases this is straightforward, but a comment is needed for the red face. If a trap cell borders to a cell on more than one side, these sides are combined into a single face. The red face is one such example, and because it is a single face with twice the length, the entire face is only assigned one face flow direction (red arrow).

The blue face shows the trap's spill face. After the face flow directions have been assigned, we have no guarantee that the one assigned to the spill face results in an outflow from the trap. In the figure we were lucky that the flow direction of the trap outlet's neighbor was towards the east, because if it had been towards the north, it would have resulted in no flow over the spill face. Thus, after we have found the face flow directions for all trap faces, we need to reexamine the face flow directions of the spill faces to make sure that there is an outflow from all traps. This will be discussed later.

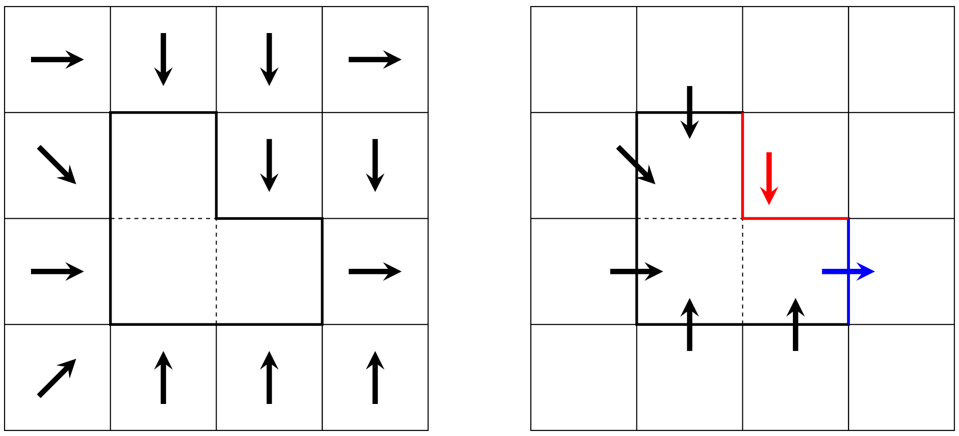


Figure 4.6: The left figure shows a trap cell and the flow directions of all surrounding non-trap cells. Unlike non-trap cells we can not assign the same face flow direction to all trap faces. Instead we do the following: For each trap face, assign the flow direction of the neighboring cell as the face flow direction. The result of this is shown in the right figure. All the faces have been assigned a face flow direction, but the L-shaped red face has only been assigned one, when we expected two. This is because the coarse grid creates a single face against a cell if several faces are shared with the trap cell. The spill face (colored blue) and its assigned face flow direction (blue arrow) results in an outflow from the trap, which is what we desire.

We will look at three corner cases where we need to take special care to obtain a time-of-flight for all cells in the grid CG. In Example 17 we showed a case where a trap bordered to a cell on more than one side, and how the faces were combined as a result of it. This merging of faces by the coarse grid functionality has created many problems for us, and we will talk about some examples where we ran into problems because of it. In Example 18 we look at the first case where a cell is completely surrounded by a trap.

Example 18. In this example we will look at the case where a cell is completely surrounded by a trap cell. Figure 4.7 shows the cell with its eastern flow direction (black arrow). The cell's only face (colored blue) has a face normal of $[0, 0]$, so no matter what the flow direction is, the flux from the cell will be zero. To fix this, we need to assign the face a face normal. To avoid zero flux from the cell, we choose the same direction as the cell's flow direction. As the face normals in CG are scaled by the side lengths, we also remember scale it by the length of a regular face.

The second example we will look at, is similar to the first, but now the cell only shares three quarters if its combined face lengths with the trap cell. If we are

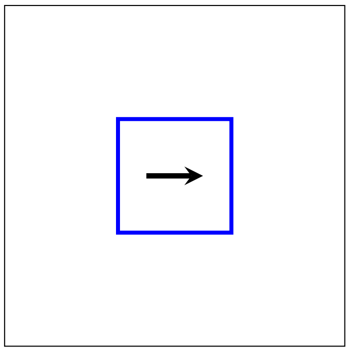


Figure 4.7: A cell surrounded by a trap cell. The border of the cell is shown in blue, and the black arrow shows the cell's flow direction. The cell has a single face with face normal $[0,0]$, which must be changed to obtain any outflow from the cell.

unlucky with the cell's flow direction, this will again lead to no flux out of the cell, which we will discuss in Example 19.

Example 19. In Figure 4.8 we show a situation where a cell only has two faces, and how this can create a problem for the outflow from the cell. The blue face has been combined from three of the cell's faces because they all border to the same trap cell, while the second face (colored red) borders to a regular cell. We use the same colors to show their face normals.

In this situation there will no outflow from the cell as the flow directions and the face normal are perpendicular to each other (α-values evaluate to zero for both neighbor cells). In this case, it does not matter what the cell's flow direction is, as long as its water flows to the trap cell, so we change the face flow directions to be in the direction of the face normal. This allows flow to continue.

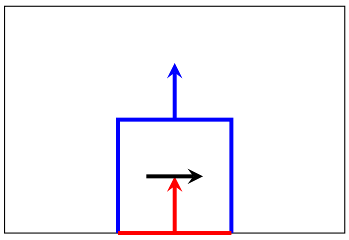


Figure 4.8: A cell surrounded by a trap cell on three sides. The cell has two faces, one colored blue and one colored red. Their respective face normals are colored with the same color. Because the face normal (blue arrow) and the cell's flow direction (black arrow) is perpendicular to each other, we need to change the face flow direction of the blue face, to have an outflow from the cell.

The third and final corner case we will show is the situation where a cell shares two opposite faces with a trap cell. An illustration of this is shown in Figure 4.9, and will be discussed in Example 20.

Example 20. If a cell shares two opposite faces with a trap cell, it still counts as one face. In Figure 4.9 this special face (blue color) has a face normal of $[0,0]$. In our case the cell's flow direction is to the east, which results in zero flux over both faces (also the red). The way we fix this, is to set the face normal of the blue face to point in the same direction as the flow direction.

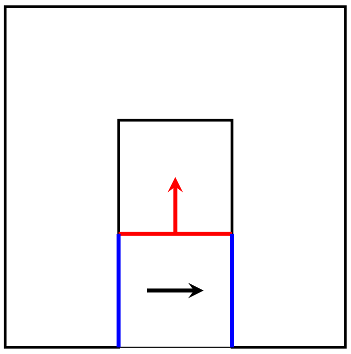


Figure 4.9: A cell is surrounded by two cells from above and below, and a trap cell at the two other sides. The coarse grid structure assigns the cell three faces, colored black, red and blue. Although it looks like the two blue lines are two separate faces, they are in fact defined as a single face with face normal $[0,0]$. When the cell's flow direction is $[1,0]$, the flux is zero over all faces. To get an outflow from the cell, the face normal of the blue face is set to the cell's flow direction.

It has come to our attention that it is actually possible to divide the trap faces into smaller faces. This would most likely have solved our problems with the corner cases, but because the code works well, we will leave it as it is for now.

With the special cases taken care of, we return to the potential problem of no outflow from trap cells. Earlier in Algorithm 6 we set the face flow directions for all faces in the traps, which by coincidence also worked for the spill face in Figure 4.6. To make sure that there is always an outflow over the spill face, we will find the spill faces in each trap, and change their face flow directions to the trap's flow direction. We will walk through how we do this for the traps. To start off, we need to locate the spill faces, which we remember are the faces closest to the spill point in the trap's flow direction. The spill point will be referred to as s.

One way to obtain the spill faces is to create vectors between the spill point s and the centroids of all trap faces, where $\Gamma = \{\vec{\Gamma}_1, \ldots, \vec{\Gamma}_n\}$ represents the set of these vectors. The choice of face/faces is then based on the angle between the vectors and the trap's flow direction vector $\vec{d}_t$. The spill faces are chosen based on the following equation where both angle and distance to spill point is taken into consideration,

$$S = \underset{i \in \{1,\ldots,n\}}{\arg\max} \frac{1}{\left|\vec{\Gamma}_i\right|} \cos\theta_i. \tag{4.17}$$

Here θ_i is the angle between $\vec{d}_t$ and Γ_i. When the angle goes to zero, $\cos\theta_i$ increases. The reciprocal of the distance makes sure that the closest angles are chosen. The set S holds the indices of the spill faces.

In Figure 4.10 we show three examples of the process. The vectors in Γ are shown with black arrows that starts in s and ends in the trap face centroids. A blue arrow represents the flow direction $\vec{d}_t$ from the spill point s, while the chosen spill face/faces are indicated by the green color. In the left figure the distances to all centroids are equal, but because the angle θ is lowest for the eastern face ($\theta = 0$), it is chosen as a result of this. In the middle figure the angles to the eastern and northern faces are both $\theta = 45°$, and because the distances to the two face centroids are also the same, both will be chosen. Finally, in the right figure, the northeastern direction is chosen, as its θ is zero.

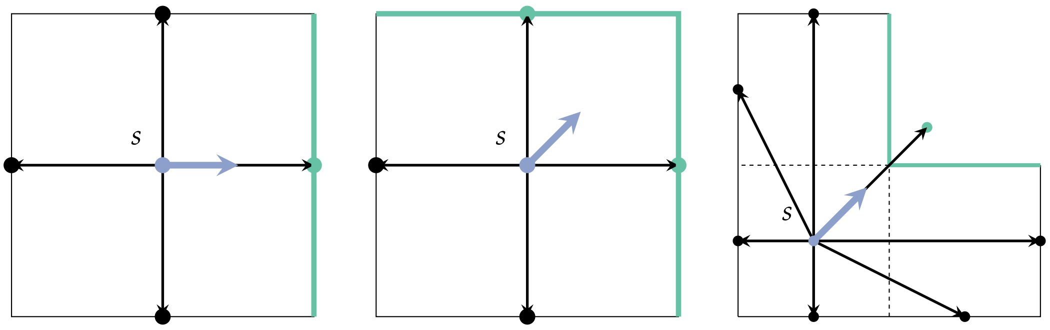


Figure 4.10: Illustration of the method to determine the spill faces in trap cells. Both the spill faces and their centroids are colored green. To determine the spill faces, we create vectors (black arrows) from the spill point s (blue point) to all trap face centroids (black points). The selection process considers the angle between the flow direction in the spill point and the vectors, as well as the distance between s and the face centroids to determine the spill faces.

After we have located the spill faces in each trap, we set their face flow directions to the trap's flow direction. This process is illustrated in Example 21.

Example 21. The left figure in Figure 4.11 shows an illustration of a trap where its initial face flow direction (black arrow) is set to an eastern direction. We can see that the spill face normal (green arrow) is perpendicular to the face flow direction. To ensure an outflow over the spill face (colored green), we change the face flow direction to the trap cell's flow direction $\vec{d}_t$, shown in the right figure.

Because each face in the grid has two face flow directions, the assignment of a new face flow direction to the spill face might disrupt the flow of the neighbor, if they have contributions in the opposite direction. In Example 22 we explain how this can happen.

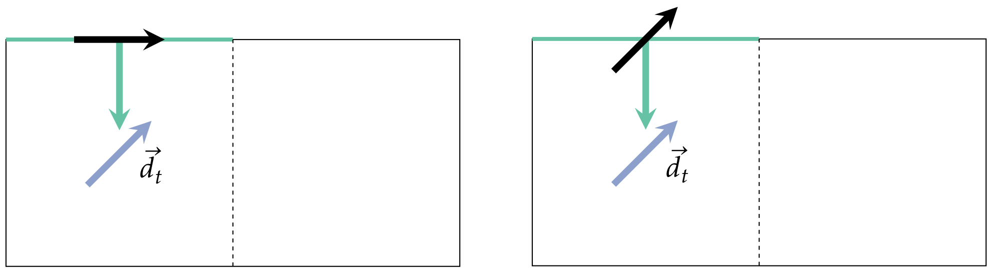


Figure 4.11: The green face shows the spill face for a trap. In the left figure there is zero flux over the face, as the face flow direction is perpendicular to its face normal. If we change the face flow direction to the trap outlet's flow direction, there will be an outflow over the spill face, which we show in the right figure.

Example 22. In Figure 4.12 we illustrate how the assignment of a new face flow direction might disrupt the flow from its neighbor cell. The left figure depicts the neighbor cell of a trap cell, and its assigned face flow directions. In an effort to ensure an outflow from the trap cell, we have set the face flow direction of the spill face (green face) to the trap outlet's flow direction $\vec{d}_t$. These are shown in the middle figure.

Because the left cell has a flow contribution towards the trap cell, and the flow contribution from the trap cell is in the opposite direction, we might get a situation where the left cell (or possibly both) loses its only outflow. To avoid this, we can change the face flow directions of the left cell to point towards the cell the trap cell would have spilled to if we had eight flow directions. This is shown in the right figure.

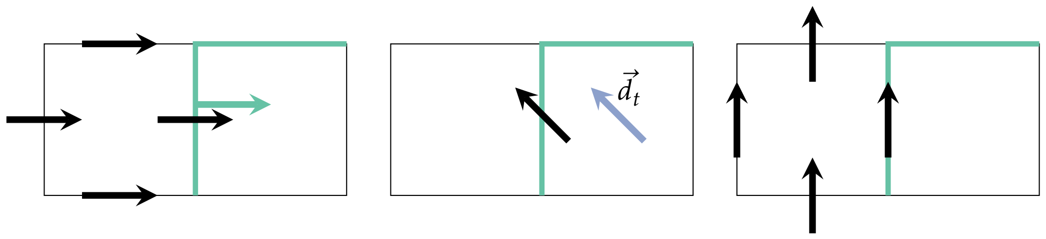


Figure 4.12: The illustrations show how a cell's face flow directions are changed to avoid a potential loss of outflow over the spill face (colored green). In the left figure, we show the face flow directions of the left non-trap cell, and we can see its flow contribution over the spill face is towards the right. When we change the face flow direction of the spill face in the middle figure, there will be a flux contribution towards the left. To make sure that there is an outflow from the left cell (and the trap cell), we change the left cell's face flow directions, which is shown in the right figure.

When we are done with all adjustments of the face flow directions and face normals, we take the dot product to obtain all flux contributions and scale them by their α-values, i.e., we have two flux contributions for each face. The last step in Algorithm 6 is to take the average of the fluxes.

4.5 Set ϕ-values

The next step in Algorithm 5 is to set the ϕ-values for the cells in the watershed. As we mentioned in the beginning of this chapter, we only use two values for ϕ: one for trap cells, and another for non-trap cells,

$$\phi = \begin{cases} \epsilon, & \text{if cell is trap cell} \\ 1, & \text{if cell is non-trap cell.} \end{cases} \tag{4.18}$$

This is based on the assumption that all traps are completely filled with water, so any inflow will result in an immediate outflow at the trap outlet — we can call this a bathtub model. This results in the simultaneous arrival of all precipitation that falls within the trap cell at a given time, at the watershed outlet. This makes sense given our assumptions, but in reality even a completely filled lake has a limited outflow at its spill point, which depends on e.g., the cross-sectional area of its outlet.

To avoid this effect, a more complex method would have to be used, and additional information would be needed. This could include taking the water storage in the terrain into account, and allowing fluctuations in the traps' water levels.

In the time-of-flight solver the pore volumes of the cells are used. This can be defined as

$$\Phi = V \cdot \phi, \tag{4.19}$$

where Φ represents the volume in each cell that can be filled with water, based on the cell's volume V, and ϕ. We now have everything we need to apply the time-of-flight solver to our watershed, and we will show some examples in the next section.

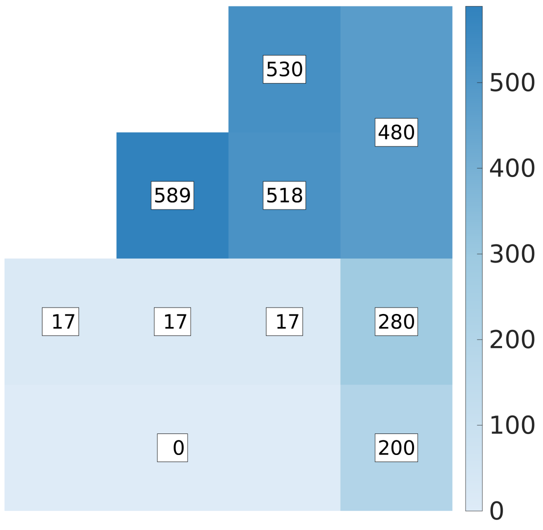
4.6 Run time-of-flight

After we have constructed the coarse grid CG, calculated the face fluxes, and set the ϕ-values for the cells, we are ready to run the time-of-flight solver in MRST. We will calculate the time-of-flights for watershed w_b. The elevations and flow directions are shown in Figure 3.10. It is expected that the travel times are affected by the elevation differences and the flow directions.

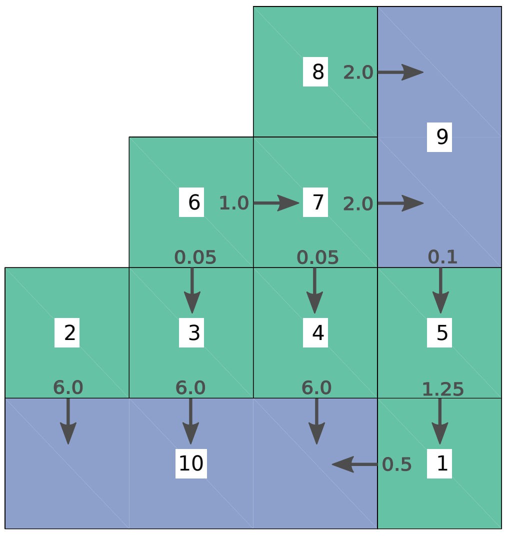
Figure 4.13(a) shows the calculated time-of-flights for the cells in w_b when the ϕ-values of the trap cells are $\phi = \epsilon = 0.1$. The numbers on white background show the time-of-flights for the cells, where T_i is cell i's travel time. The cell in the lower left corner is the source, and hence has a time-of-flight zero[3]. We will explain the results a bit more in Example 23.

Example 23. In Figure 4.13(b) we show the directions and magnitudes of the face fluxes. If we compare it to Figure 4.13(a) we can see it takes a much longer time for water to reach the outlet if it passes faces with a low flux; this occurs at locations where the slope is gentle. The closest cells to the outlet illustrate this well, as the elevation differences of cells 2, 3 and 4 are $\Delta z = 6.0$, and the difference for cell no. 1 is $\Delta z = 0.5$.

We know low ϕ-values result in a higher speed, which is the case for cell no. 9, but T_9 is actually quite high. This is not because of the ϕ-value, but instead the low slope over its spill face ($\Delta z = 0$), which results in the 'uphill scenario' in Equation (4.15), where $\alpha = C$. Because we use $C = 0.1$, the assigned flux is low. It is possible that C should be higher to avoid bottlenecks like this, but we will not focus on this for now.



(a) Time-of-flights for w_b.



(b) Flux directions and magnitudes.

Figure 4.13: (a): The time-of-flights for watershed w_b with $\phi = 0.1$ for trap cells. The values in the centroids of the cells show time-of-flights for the cells. (b): The vectors and the adjacent numbers show the directions and magnitudes of the fluxes to better understand the flow patterns. When T is calculated, all the fluxes are flipped so the water flows from the source to all the upslope cells.

To verify that the solver works as expected, we use Equation (4.12) and calculate the time-of-flight for one of the grid cells; this will in turn be compared to the

[3]The solver actually returns a non-zero time-of-flight for the source, because it factors in the time it takes to fill it. We want the time-of-flight to represent travel time from a cell to the source, so we subtract the source's time-of-flight from all cells' time-of-flights.

time-of-flight the solver yields for the same cell. Because of its small size, watershed w_b is ideal for this comparison. To make it easier to follow, we state Equation (4.12) again, with a minor change: $\phi_i|C_i|$ has been replaced by the pore volume Φ_i of cell i,

$$T_i = \frac{\sum_{j\in U(i)} T_j f_{ij} - \Phi_i}{\sum_{j\in U(i)} f_{ij}}. \tag{4.20}$$

The comparison is shown in Example 24 for cell number 6.

Example 24. To obtain the time-of-flight T_6 from Equation (4.20), we need to locate its upstream cells, and it is important to keep in mind that all fluxes must be flipped. According to Figure 4.13(b) the only upstream cells are cells 3 and 7. We use $\Phi_6 = 100\ \text{m}^2$ and the fluxes of the upstream cells to obtain,

$$\begin{aligned} T_6 &= \frac{\sum_{j\in U(6)} T_j f_{6j} - \Phi_6}{\sum_{j\in U(6)} f_{6j}} \\ &= \frac{\sum_{j\in\{3,7\}} T_j f_{6j} - \Phi_6}{\sum_{j\in\{3,7\}} f_{6j}} \\ &= \frac{100}{-1.05} + \left(\frac{-0.05 * 16.67}{-1.05} + \frac{-1.0 * 517.48}{-1.05}\right) \approx 589 \ \text{ s.} \end{aligned}$$

The first part shows the time it takes to fill the volume of cell 6, while the calculations inside the parentheses constitute a weighted average of the time the water uses if it travels south or east, respectively. We see that both the solver and the equation yield the same answer.

In Example 25 we will show two cases where we vary ϕ for trap cells to see how ϕ affects T.

Example 25. In this example we will compare different values of ϕ for trap cells, to see how it affects the time-of-flights. The first case we look at is the case where we do not account for higher flow speed in trap cells. This means that $\phi = 1$ in the entire watershed. Compared to Figure 4.13(a), where $\phi = 0.1$, we expect the time-of-flight to increase. The left figure in Figure 4.14 shows that this is the case. The areas that are downstream of cell no. 9 are unaffected, but the upslope cells 6, 7 and 8, as well as 9 itself, get much higher travel times. This is because the speed in the cell is one tenth of what it is when $\phi = 0.1$.

If we instead decrease the trap cells' ϕ to 0.01, we expect that it takes a tenth of the time to fill cell no. 9 compared to Figure 4.13(a). The right figure in Figure 4.14 shows the result. We can see that the flow speeds in the traps affect the travel times a lot, and as we previously argued, it makes more sense that the trap cells take a short time to fill, when our traps are full.

Up until now we have only calculated the time-of-flights for a small watershed, but in the results chapter we will look at a much larger one.

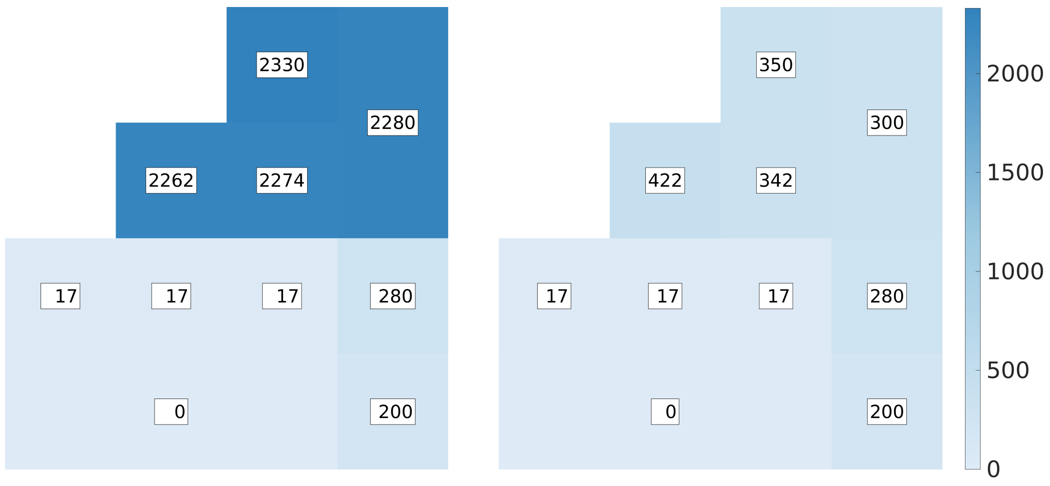


Figure 4.14: A comparison between two different ϕ-values for trap cells in the grid. In the left figure $\phi = 1$ for all cells, which yields very high T-values for cells with flow to trap cell no. 9. It is further aggravated by the low flux between cells 9 and 5. In the right figure, $\phi = 0.01$, so it takes a short time to fill cell no. 9, meaning that flow can cross the trap quickly.

5 Calculate hydrographs

In Chapter 4 we calculated the time-of-flight for every cell in the watershed. With this in place, we can introduce precipitation to our watershed. From the weather forecast we know that rainfalls vary, both in intensity and in pattern. Sometimes storm clouds release the precipitation as a soft drizzle that never seems to end, and other times it pours down with enormous force, until it abruptly stops, almost like it never happened (for example convective summer showers in Norway). Both of these cases are interesting to us, and we will construct different rainfalls where we vary parameters like intensity, duration, and which areas that are affected. By doing this, we learn how the precipitation and time-of-flight affects the hydrograph at the outlet of the watershed. We will start with a discussion on how we calculate the discharge, given information about the rainfall. This will later be put to use when we compute the hydrograph.

An equation for the discharge as a function of time, can be written as

$$h(t) = \int_{\Omega} P(x, t - T(x))\,\mathrm{d}\Omega, \tag{5.1}$$

where $h(t)$ is the discharge in time t at the watershed outlet. To represent the precipitation in the domain Ω (the watershed) at a time t, we use the function $P(x,t)$. Because of the travel time, the water that affect the discharge in time t is the water that fell a time T ago, so we use $P(x, t - T(x))$ in Equation (5.1).

To estimate the runoff from the watershed, we discretize the equation in space. For cell i we approximate its runoff contribution at time t as

$$\begin{aligned} h_i(t) &= \int_{\Omega_i} P(x,t - T(x))\,\mathrm{d}\Omega \\ &\approx A_i P(x_i, t - T(x_i)) \\ &= A_i P_i(t - T_i), \end{aligned} \tag{5.2}$$

where A_i is the cell's area, T_i its time-of-flight, and P_i its precipitation. The precipitation intensity in cell i's centroid x_i is used for the entire cell. We now use Equation (5.2) to discretize Equation (5.1) as

$$\begin{aligned} h(t) &= \sum_i h_i(t) \\ &\approx \sum_i A_i P_i(t - T_i). \end{aligned} \tag{5.3}$$

The units of the discharge $h_i(t)$ will be,

$$[h_i(t)] = [A_i\,\mathrm{P_i}(t - T_i)] = [A_i][\mathrm{P_i}(t - T_i)] = \mathrm{m}^2\frac{\mathrm{mm}}{\mathrm{hour}} = \frac{10^{-3}}{3600}\frac{\mathrm{m}^3}{\mathrm{s}} = \gamma\frac{\mathrm{m}^3}{\mathrm{s}}, \tag{5.4}$$

where we need to scale by a constant γ equal to $1/3.6 \cdot 10^6$ to get the discharge in m^3/s.

The first storm events we will look at are not based on real data. Rather, we have created scenarios that visualize how the computed time-of-flight and precipitation are coupled together to create hydrographs. In these rainfall events we will revisit watershed w_b from Chapter 4, and we will also use the time-of-flight results from Figure 4.13(a). To keep it simple, we start with the most basic example: a uniform precipitation intensity that covers the whole watershed for an equal duration. From here on we will often refer to a rainfall as a rainstorm, or simply a storm.

5.1 Uniform storm

We will simulate a uniform storm in the entire watershed. To do this we let the precipitation intensity and rainfall duration be the same for the entire area ($\mathrm{P_i} = P$ and $D_i = D$). Because the movement of water from watershed to outlet is not instantaneous, the water will continue to flow after it has stopped raining. If we use the travel time T and duration D, we can create a time interval which shows when the outlet might receive precipitation

$$t \in [\min(T),\ \max(T) + D], \tag{5.5}$$

where $\min(T) = 0$ for every watershed, if the storms are uniform. The time interval shows that the rainfall will affect the outlet a long time after it has stopped raining, and the discharge might even peak hours afterwards.

We will take a look at the resulting hydrographs for watershed w_b when we use a precipitation intensity of $P = 10$ mm/hour, and compare the following durations: one minute, ten minutes, and one hour. The travel times we use for w_b are the ones in Figure 4.13(a), where the outlet is located in the cell with time-of-flight zero. In Figure 5.1 we show the three hydrographs. Before we talk more about the results, we want you to notice the illustration in the top right corner of Figure 5.1(a). These illustrations will be used throughout our examples to indicate which areas in the watershed that receive precipitation; only the blue areas are affected by the rainfall. In this case, all of w_b will receive precipitation for the entire duration.

We will start with a small observation about the time interval in Equation (5.5). If we calculate the time interval for the three durations, $D = 60$, 600 and 3600, we get a contribution to the discharge at the outlet in the time intervals [0, 649], [0, 1189] and [0, 4189], respectively, which the three plots confirm. These time intervals will only state the earliest and the latest time the outlet will receive water, and do not say anything about what the discharge will be in the course of the interval.

Because these are the first hydrographs we show, we will take a closer look at the one in Figure 5.1(a). The figure shows the discharge at the outlet in units m^3/s over time. Because the size of the watershed is so small[1], and the duration of the

[1]We will later see that an increase in the number of grid points makes the hydrographs smoother.

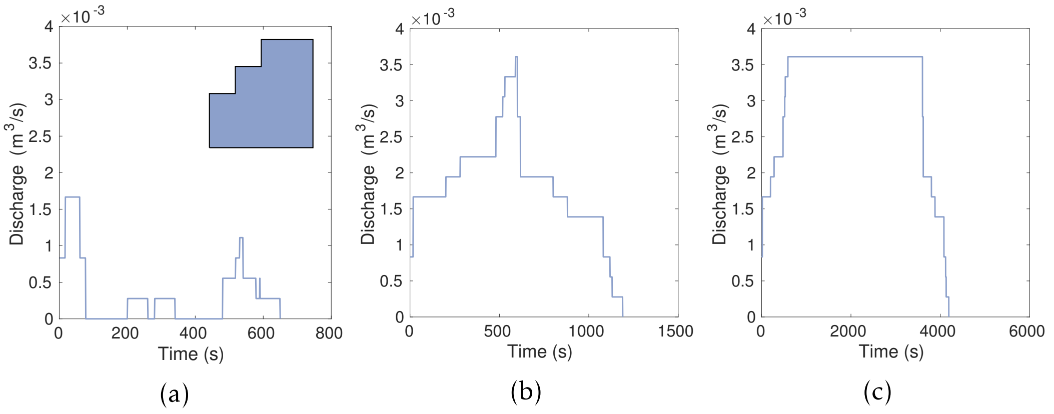


Figure 5.1: Watershed w_b's resulting hydrograph when a uniform precipitation of P = 10 mm/hour lasts for D = 60 (a), 600 (b) or 3600 (c) seconds. Note that different time axes are used in the plots. The illustration in the top right corner of (a) illustrates which areas receive precipitation, in this case all of w_b.

rainfall so short, the hydrograph will look very discontinuous, which makes the cells' individual contributions easier to see. In the beginning, the cells with the lowest time-of-flights, $T = 0$ seconds and $T = 17$ seconds, are the only contributions. The superposition of their discharges results in the maximum discharge in the time interval. Next, the two cells with $T = 200$ and $T = 280$ create two small peaks. Afterwards the discharge hits zero until $t = 480$, where it picks up again. The discharge increases at time 518, and even more at 530. The contributions from these three cells result in the second largest discharge peak. After the flow from the trap cell stops, there is a tiny overlap between the cells with $T = 530$ and $T = 589$. This yields a small peak at the end. Finally the last flow arrives at the outlet.

In the second and third figure, Figure 5.1(b) and Figure 5.1(c) respectively, the hydrographs look quite different from the first one. The longer duration of precipitation causes the flow from the different cells to have more overlap. In the third figure it is clear that the watershed cannot reach a higher discharge than 0.0036 m^3/s, a level which is reached for a small period in the second figure as well. This level is only attained when all cells in watershed w_b contribute flow to the outlet simultaneously. Both figures show that it potentially takes some time for the discharge to reach its peak.

5.2 Precipitation with varying intensity

Unlike in the previous section, where we used a uniform precipitation, we now want to do something slightly more realistic. Usually a rainfall has a storm center where the precipitation's intensity is at its highest. This intensity will decrease towards the edges of the rainfall. To accomplish this we use an intensity function $I(x)$. It has the property that the intensity is at its highest in the center of the region, and at its lowest at the boundary of it. In the following we will let the rainfall affect different areas with different intensity. This means that the rainfall's position is dependent on time — the storm moves around. In the following scenarios we will use a circular disk and a rectangle as the shape of our precipitation. The function we use is a Gaussian function that we have tailored to our needs,

$$I(x) = Ae^{-\frac{x^2}{2(R/3)^2}}. \tag{5.6}$$

The intensity $I(x)$ will represent the precipitation's intensity a distance x from the center of the rainfall. When the precipitation is shaped like a disk, x is simply the distance from the disk's center. For a rectangular shaped rainfall, we define the center of the rainfall as the rectangle's centroid. The distance from the center x will then be the distance from the centroid in the direction of movement. We use the amplitude A to represent the maximum precipitation intensity. This value is given in mm/hour, and represents the intensity in the storm center. The value of R is the radius of the disk for the first case, and half the width of the storm front in the second case. As an example, the intensity $I(x)$ for a radius $R = 10$ m and an amplitude $A = 5$ mm/hour create the bell curve seen in Figure 5.2.

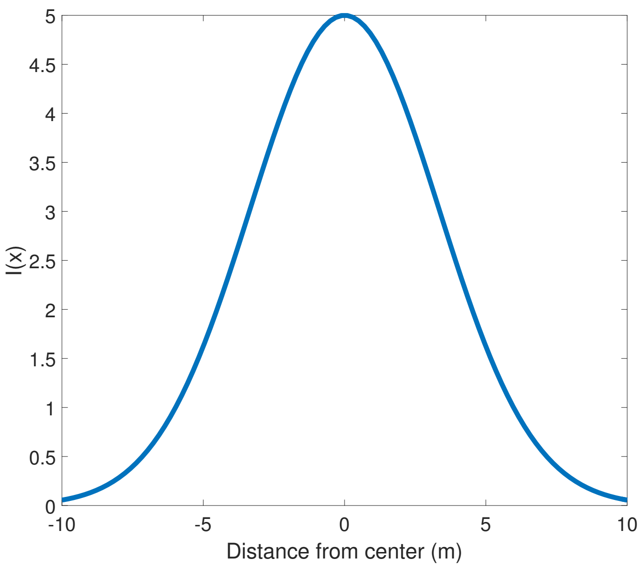


Figure 5.2: The intensity function $I(x)$ for a radius $R = 10$ m, and an amplitude $A = 5$ mm/hour.

$I(x)$ has an intensity equal to its amplitude A when the distance from the center is zero, and is close to zero if the distance is R.

In the rest of this chapter we look at scenarios where the precipitation moves around. In the calculations we let the precipitation stand still in time intervals of length Δt. If we assume that the last runoff comes in time t_{max}, we divide the time interval $[0, t_{max}]$ into Δt-intervals, where one interval is denoted as $[t_n, t_{n+1}]$ where $n \in \{0, \dots, N\}$. We let the precipitation's location be constant in each time interval, for which its position in the middle of the time interval, i.e., $t_{n+\frac{1}{2}}$ is used. In this chapter we will only use $\Delta t = 1$ second, but in Section 6.4 we will use different values for Δt to speed up the computations.

5.3 Moving storm front

We will simulate a storm front sweeping across a watershed w, and based on the fallen precipitation construct a hydrograph. The storm front is represented by a rectangle of width w, length l and center (c_x, c_y), which moves at a given speed in one of the four directions: north, south, west or east. Only the cells with their cell centroids within the storm front will receive precipitation. This means that even if only half the cell is inside, all of it will receive precipitation. This goes both ways, and is a discretization error.

As we mentioned in Section 5.2, the intensity $I(x)$ of the precipitation will vary depending on the cells' distance from the storm center. Equation (5.6) is used to calculate this intensity. When the movement is meridional, i.e., movement north or south, x represents the distance $|y - c_y|$ where $x \in [0, w/2]$, and analogously for zonal movement, i.e., movement east or west. The intensity $I(x)$ is at its largest when $x = 0$, and at its lowest when $x = w/2$. Our moving front scenario is different from the scenario in Section 5.1 because not all cells are affected, and the cells that are, get a different amount of rain. The cells at the outside of the storm front will not receive any precipitation.

Figure 5.3 shows a storm that is moving north across w_b at a speed of $v = 5$ m/s. In the center of the storm, the precipitation has an intensity of $A = 10$ mm/hour, while the intensity in the rest of the front is given by $I(x)$. The front is represented by the black rectangle, and the centroids of the affected cells in each time step have been marked with black asterisks. If we compare the grid to Figure 4.3, it is clear that it is the centroids of the fine grid that are used in the algorithm (not the centroids of the trap cells). The reason is that we want to avoid situations where an entire trap cell is assigned precipitation just because its centroid was inside the storm. This is a problem because the trap cells are potentially much larger than regular cells (see Section 4.3). However, we still need information from the coarse grid, as we have to map the time-of-flight results from the coarse grid to the fine grid in our calculations.

Initially, the storm starts halfway into w_b, and it barely covers four of the cell centroids. Because these are the only cell centroids within the front, they are the only cells to receive precipitation. The first time step is also a good example of how discretization errors can cause trouble. In our algorithm we assign precipitation to cells inside of the front. In this case, only half of their areas are inside, but we let the entire cell areas receive precipitation. This is not the only problem. The four cells will all receive precipitation at the lowest intensity, even though there are

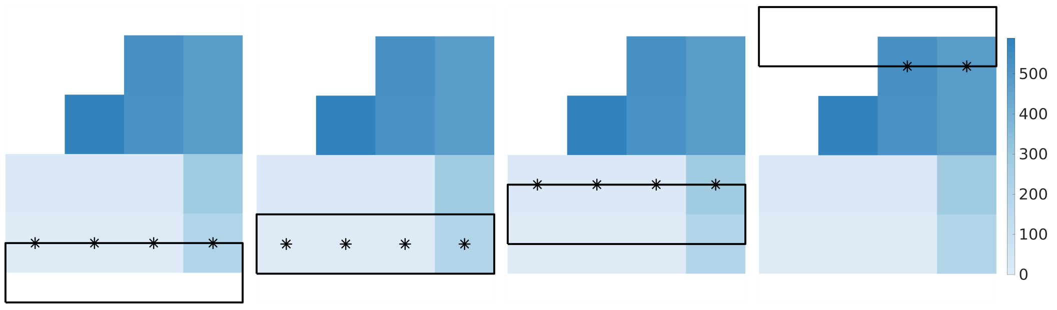


Figure 5.3: A storm front crossing the watershed w_b, represented by a black rectangle moving north at a speed of $v = 5$ m/s. The intensity of the precipitation $I(x)$ is given by (5.6). Each time step is a second, and we show the first three seconds, along with the last time step. The cells affected by the storm front have their centroids marked with black asterisks.

sections within the cells that are closer to the storm center. Both of these problems are hopefully something that evens out in the end, and are masked by uncertainties inherent in precipitation measurements and other errors/uncertainties. The second figure shows the next time step, where the cells now receive precipitation at full intensity, and in the third time step four new cells are affected. The front continues to move until it has crossed the watershed, and the last time step where cells in w_b receive precipitation is shown in the fourth figure.

The hydrograph the storm front creates is shown in Figure 5.4. In the process to obtain the hydrograph, we use the time-of-flight results from Figure 4.13(a). It is not a very interesting result as the discharge peaks are very concentrated, but it would surely be interesting to observe the watershed outlet when multiple tiny flash floods arrive. However, lack of discharge at times is expected if baseflow and interflow are excluded. We will also observe later that longer rainfall durations reduce the time periods of zero discharge at the outlet. In the figure we observe that the beginning is the only time when discharges arrive simultaneously. Later, the discharge from each cell arrives separately. As we saw in Figure 5.3, each cell is only affected for a few time steps, so a cell's discharge is very concentrated. If the cells were affected for a longer time, the hydrograph would look different. We see in the figure that each cell's discharge comes after the sum of their time-of-flight value and the time it takes the front to get there, and it lasts for a few seconds. We will do a more thorough walk-through of why the hydrograph looks as it does in a later example.

We mentioned that longer durations would affect the appearance of the hydrograph, and we will do a comparison to verify this. In order to accomplish longer precipitation durations we reduce the speed of the front. The speeds we have used in the comparison are: $v = 0.5$ m/s, $v = 0.15$ m/s, $v = 0.10$ m/s and $v = 0.05$ m/s. In all cases the maximum intensity is $A = 10$ mm/hour. The result of the comparison is shown in Figure 5.5. The sequence of hydrographs shows that the peaks gradually

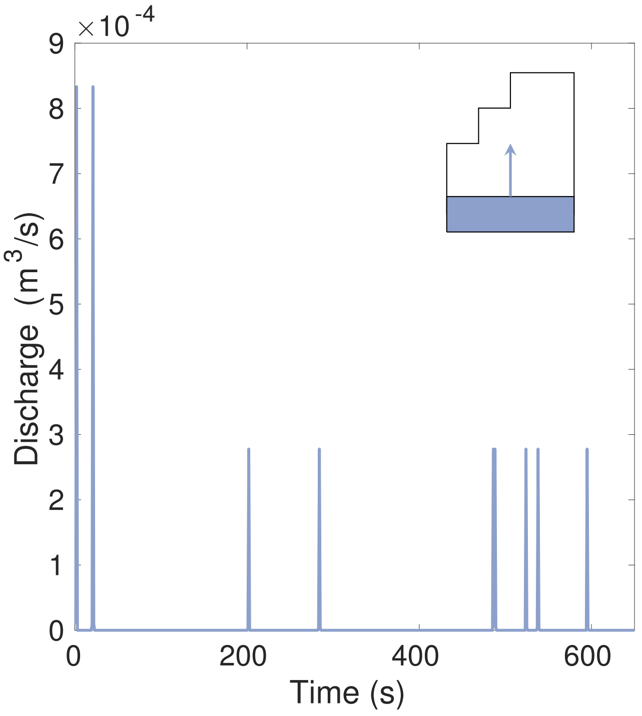
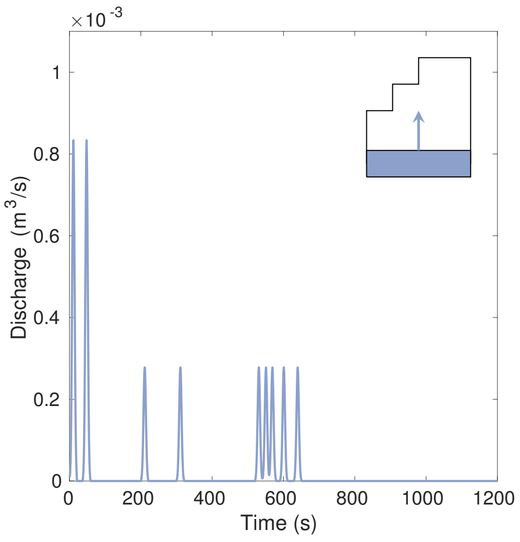


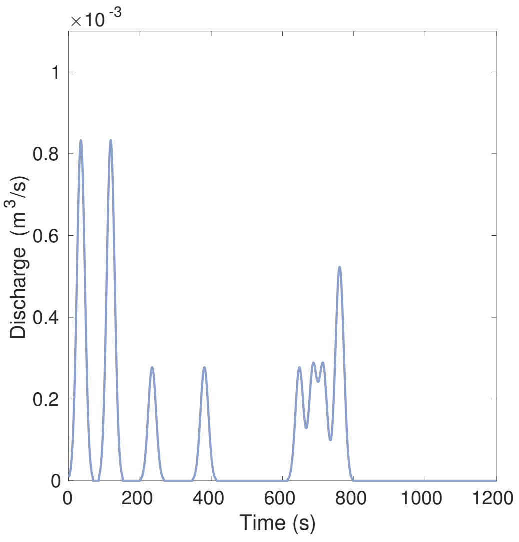
Figure 5.4: The hydrograph created by a front moving north across w_b at a speed of $v = 5$ m/s with maximum intensity $A = 10$ mm/hour. Because the storm crosses the watershed so quickly, the discharge is low and concentrated around the individual cell's discharge.

gets smeared out as the speed of the front decreases. This leads to fewer time periods of zero discharge at the outlet. In the process the peaks slide over into each other, and as this happens, the peaks grow taller. The longer durations cancel out the differences in time-of-flight between the individual cells, so that their discharges overlap. A hydrograph is only a superposition of the individual cell's hydrograph after all. Over the course of the four figures, we can see that the number of peaks is reduced from nine to four. If the speed was further decreased, only one peak would eventually be left.

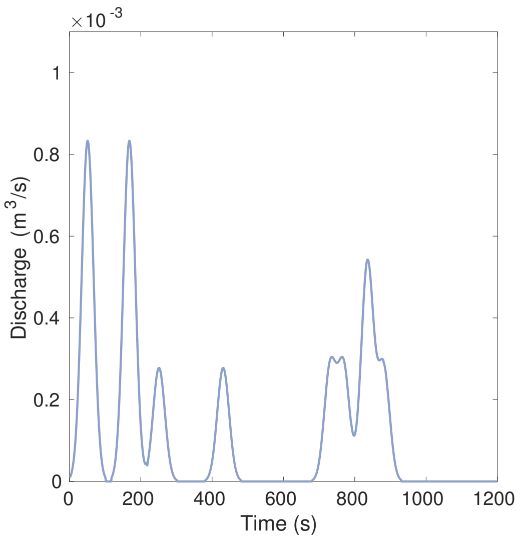
In the previous examples we considered a storm front that moved northwards. In the following example we observe how the appearance of the hydrograph is changed when the storm front's direction of movement is changed. We will see that the change of direction alters when the first discharge arrives at the outlet, and when the last water arrives at the outlet after the storm front has passed. We reduce the speed of the front to $v = 0.1$ m/s, while keeping maximum intensity at $A = 10$ mm/hour. The four directions we will compare are towards the north, south, east and west. The front will start with one half inside of w_b, and the other half outside of it.



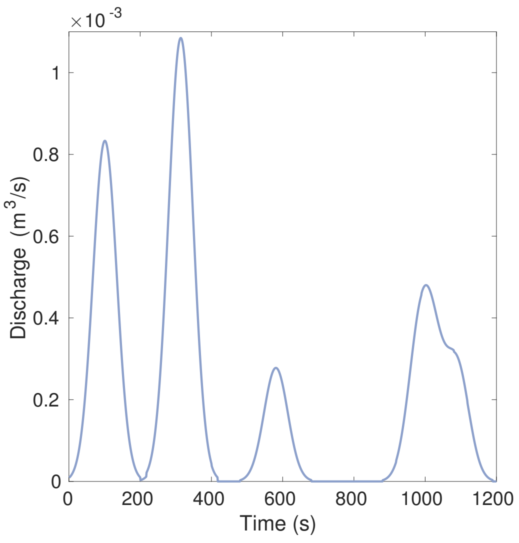
(a) $v = 0.5$ m/s



(b) $v = 0.15$ m/s



(c) $v = 0.1$ m/s



(d) $v = 0.05$ m/s

Figure 5.5: A comparison of the hydrographs for watershed w_b when the speed v of the storm front is varied. The direction of movement is north, and the maximum intensity is $A = 10$ mm/hour. When the speed decreases, the area that contributes discharge to the outlet in each time step, grows larger.

The resulting hydrographs vary quite a bit, and we will do a thorough explanation of one of the examples, in addition to some comments on the rest. Instead of a meridional movement, we turn our attention to a zonal one — west. Compared to the northern movement in Figure 5.5(c), there are some differences we immediately notice in Figure 5.6(d), such as differences in when the outlet receives flow in the beginning and the end, and the differences in peak heights. It all comes down to when the storm front arrives at the cells, and their time-of-flights. If you look at the sixth peak in the figure. This will be our reference discharge, as this is the discharge that comes from a single cell (100 m^2). If a peak is not a multiple in height of this peak's height, the cells that contribute discharge simultaneously have different time-of-flights.

We will now study Figure 5.6(d) more in detail. The first cells that receive precipitation have time-of-flights $T = 200$, $T = 280$ and $T = 480$. This means that no discharge will reach the outlet from any of them before 200 seconds have passed. However, the front only uses 100 seconds to move to the cells with time-of-flights $T = 0$ and $T = 17$, so their combined discharges create the first peak. Because of the time difference, the peak is not quite twice the height of our reference peak, nor is the second peak three times as large, even though it is caused by the discharge from three cells: those with $T = 0, 17$ and 200, specifically. Nevertheless, this is the maximum discharge at the outlet in the recorded time. The last peak before a period of zero discharge comes from the last pair of cells with $T = 0$ and $T = 17$, in addition to the cell with $T = 280$. The former cells receive precipitation after 300 seconds, so the overlap is not too great, which causes the peak to be only slighter taller than the discharge from two cells. In terms of time-of-flight the watershed is kind of separated in two, the bottom and the top region. This causes the discharge to be zero for a little while, before the flow starts to come from the top region. The fourth peak is a result of the discharge from the second trap cell with $T = 480$. It is exactly twice the height of our reference peak. It takes 100 seconds before the precipitation reaches the next cells, $T = 518$ and $T = 530$. The final peak is our reference peak from the cell with $T = 589$, which arrives after 789 seconds, 200 seconds after the rain started to pour there.

We will not analyze the other flow directions as much, but instead focus on some key points. If we compare Figure 5.6(a) to Figure 5.6(b) we can see that the first two peaks overlap slightly in the southern direction. When the front is moving south, the cells with $T = 17$ reaches the outlet 17 seconds before the precipitation does, so before it starts raining in the bottom trap cell where the outlet is, there is already a discharge. The fact that the storm starts in the cells furthest away from the outlet, makes the discharge have more overlap, which can be seen in the rest of the figure. In this case, it also causes a larger peak discharge in the end, almost as tall as the ones in the beginning. The east-direction is the direction that causes the shortest peaks, and three similar peaks in the beginning from the cells with $T = 0$ and $T = 17$. This direction is the only direction with the tallest peak in the end. It turns out that watershed w_b results in very different hydrographs depending on which direction the storm front is moving in.

Although the examples we used in this section did not show it, situations might arise where we get a double counting of the precipitation to the cells. This is caused by a round-off error in centroid coordinates at machine epsilon level. This results in situations where the cell centroids are just barely inside, or outside of, the storm front. So sometimes, if the storm front is exactly in the middle between two cells, like in the third time step in Figure 5.3, it is somewhat arbitrary how many of the centroids that are included. We see that only the centroids at the top of the storm front are within the rectangle, but sometimes you get the centroids at both sides, which leads to counting their contributions twice. This is not possible to see when we work with large watersheds, so we will ignore it, but an example is shown in Figure 5.7. The intensity is uniformly set to $A = 10$ mm/hour, i.e., not a Gaussian distribution. The front moves at a speed of $v = 1$ m/s, so the cell with time-of-flight $T = 480$ will be reached after $t = 20$ seconds, but with a width $w = 10$ m completely overlap the cell after $t = 30$ seconds. We can see that this coincides with a double counting of the contributions in time 511[2].

5.4 Moving disk

In the previous section we looked at how the hydrographs turned out when the precipitation was shaped as a rectangle that represented a storm front. We will now simulate precipitation as a moving disk. The intensity of the precipitation still depends on the distance from the storm center, which for a disk is the euclidean distance from the storm center. We denote this distance r, so our equation becomes

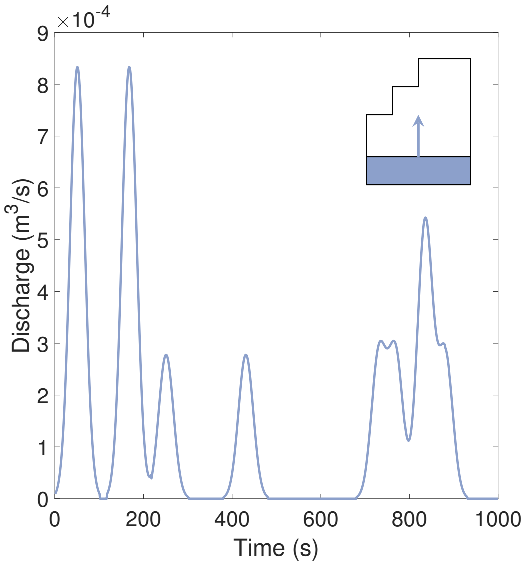
$$I(r) = Ae^{-\frac{r^2}{2(R/3)^2}}, \tag{5.7}$$

where the maximum intensity is denoted A, and the radius of the disk is R. We will look at different scenarios where we vary the direction the storm moves in, and the radius of the storm.

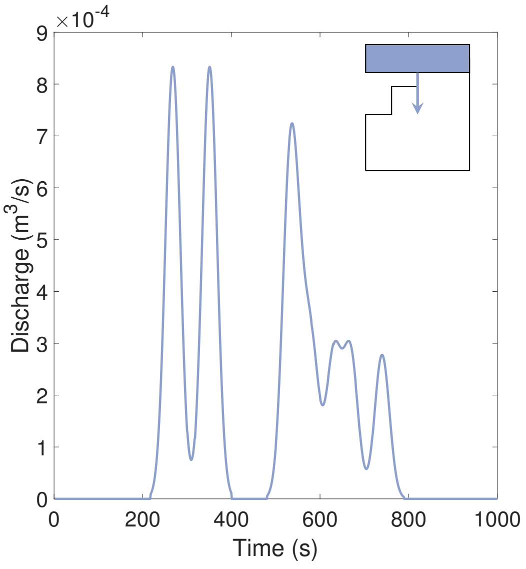
We start like we did in Section 5.3 with an illustration of a storm crossing w_b. The storm is shaped like a disk and starts with the center in the lower left corner of w_b, and moves north east at a speed of $v = 5$ m/s. The maximum intensity is $A = 10$ mm/hour, and the radius of the disk is $R = 10$ m. Because of the high speed, the disk crosses the landscape very fast. The first three time steps, as well as the final time step before the disk stops to contribute precipitation, is shown in Figure 5.8. The centroids of cells that receive precipitation are shown with black asterisks. Their distance from the center determines the amount of precipitation they receive.

The hydrograph in Figure 5.9 is the result of the moving storm in Figure 5.8. It resembles the hydrograph we saw in Figure 5.3, but because the disk does not cover all cells in w_b in the course of its movement, there are fewer discharge peaks. Similar

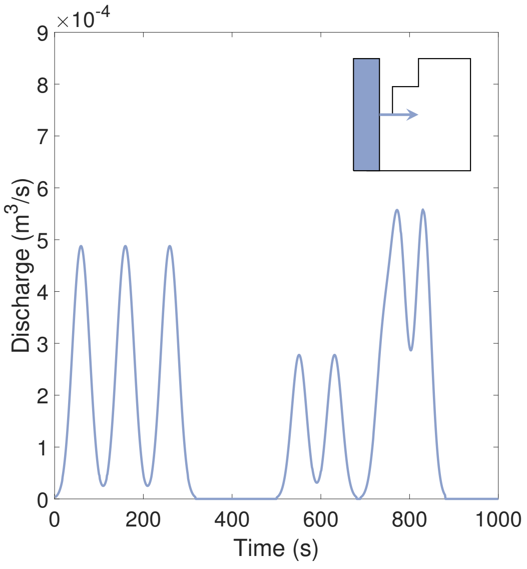
[2]It should be at time $t = 510$, but because of Matlab's one based indexing, everything is shifted by one.



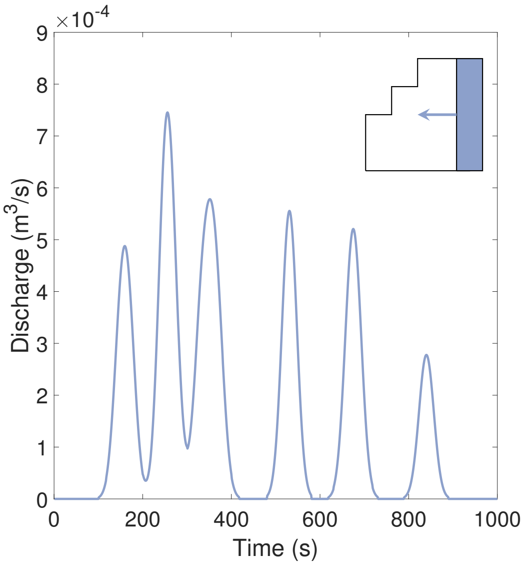
(a) North



(b) South



(c) East



(d) West

Figure 5.6: When a 10 by 40 meters large storm front crosses the watershed w_b in the four cardinal directions, these are the resulting hydrographs. The starting point and direction of the storm front is shown in the top right corner of each hydrograph. In all cases the maximum intensity is $A = 10$ mm/hour and the front travels at a speed of $v = 0.1$ m/s.

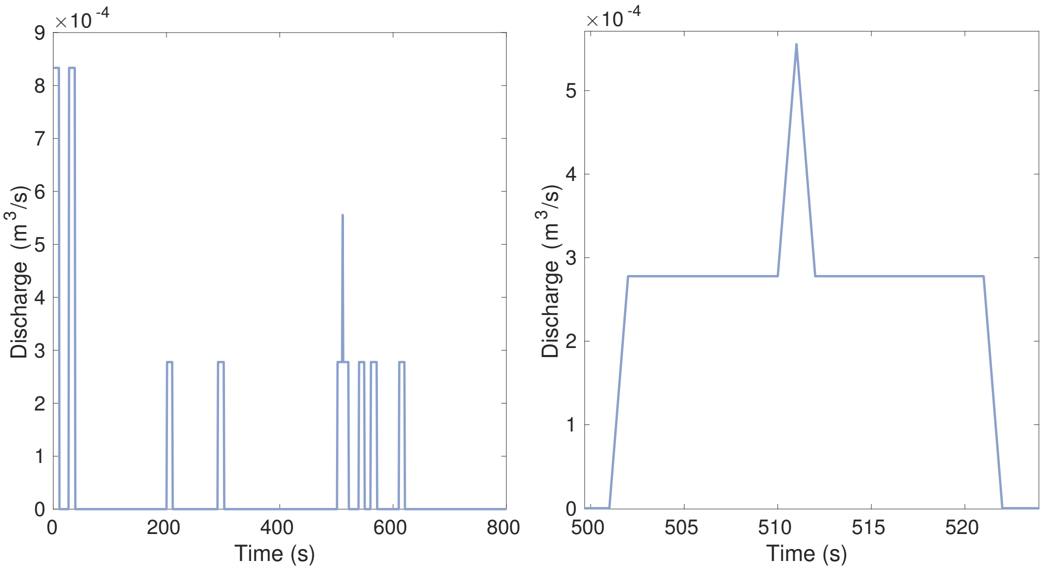


Figure 5.7: A round-off error at machine epsilon level leads to a double counting of precipitation in cases where the storm front and cell faces overlap. This is not noticeable in hydrographs for larger grids.

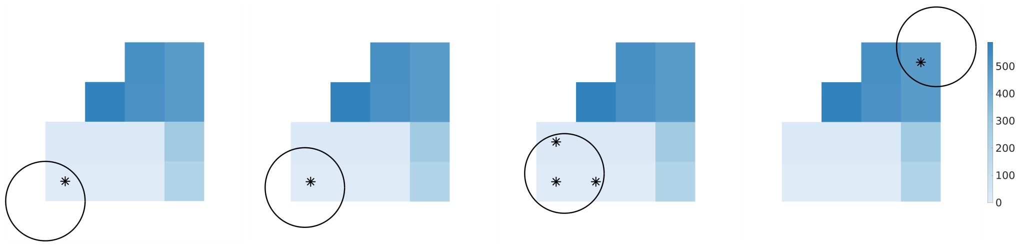


Figure 5.8: A storm crossing watershed w_b is represented by a disk outlined by a black circle. The disk moves northeast at a speed of $v = 5$ m/s. The intensity of the precipitation is given by (5.7), where $A = 10$ mm/hour. Each time step is a second, and we show the first three seconds, along with the last time step. The cell centroids of the affected cells are marked with black asterisks. The time-of-flights for the cells are shown in the background.

to the moving front, the Gaussian function determines how much precipitation each cell gets based on the cell's proximity to the center. In the first time step, only one cell receives precipitation, but because its centroid is close to the border of the disk, it does not receive much. In the second time step, the cell will receive more. The contributions from the cells with low time-of-flights creates the two large peaks in the beginning, and the rest of the peaks are all from the upper part of w_b.

Next, we compare two storms against a storm that moves in the northeastern direction, to see how the direction affects the hydrographs. The two storms are

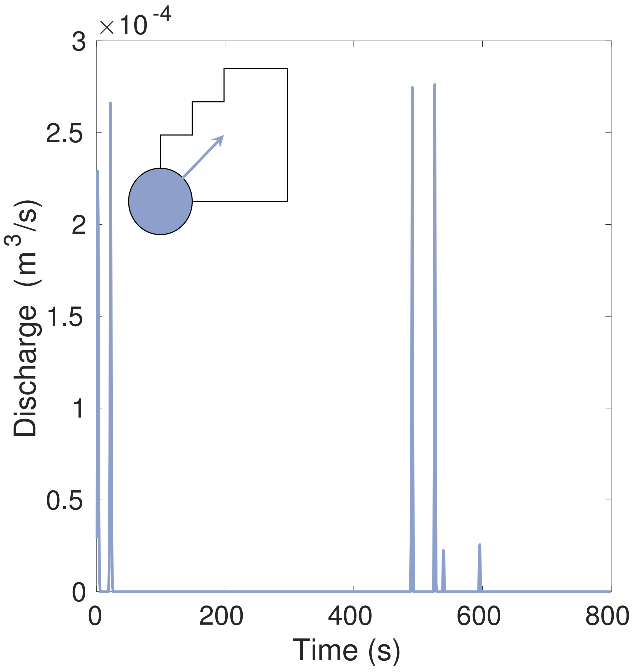


Figure 5.9: The hydrograph from a storm moving northeast across w_b at a speed of $v = 5$ m/s and a maximum intensity of $A = 10$ mm/hour. Because the storm crosses the watershed so quickly, the discharge is concentrated around each cell's discharge and passes the outlet very quickly.

shifted 15° away from the northeastern direction, in either the northern or southern direction. The result is seen in Figure 5.10. When the movement is changed, the storm will either no longer release rain to the same cells, or they will receive rain at different intensities. This is reflected in the hydrographs. Both this example and the one we saw in Figure 5.6 shows that the direction a storm is coming from is not insignificant for how the outlet in the watershed is affected.

In the following we will look at how the size of a storm affects the landscape and the accompanying hydrograph differently. The radius of each disk will determine the size of the storm. We will ensure that the volume of water that falls within each disk in a given time period is equal by using different intensity functions. More specifically this is achieved by varying the maximum intensity A. To calculate the volume within each disk, we first integrate the intensity function (5.7) for an

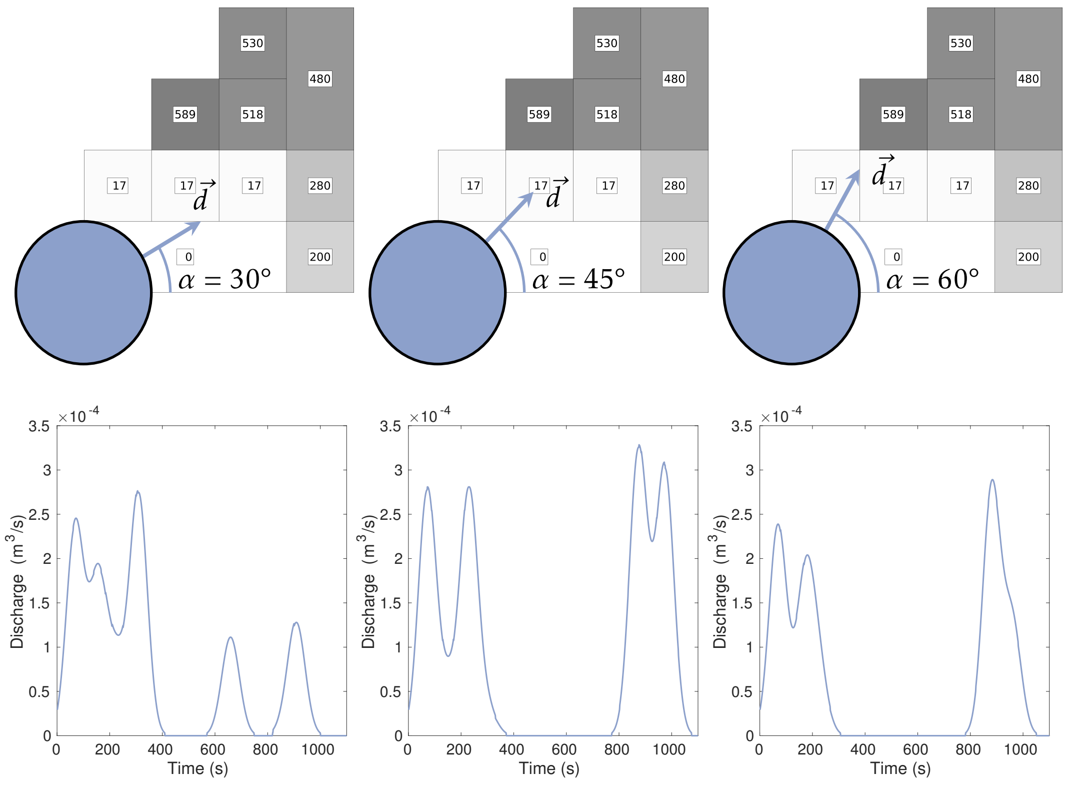


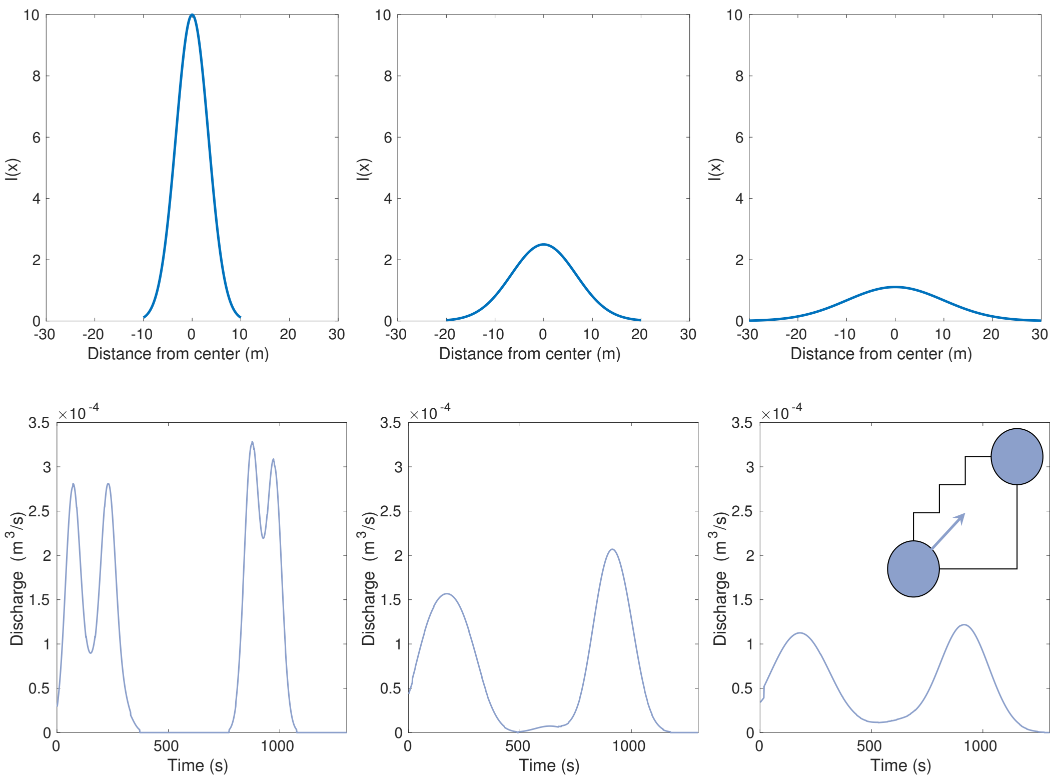
Figure 5.10: A storm with radius $R = 10$ m, and maximum intensity $A = 10$ mm/hour has an initial storm center located in the lower left corner of the domain. The storm moves at a speed of $v = 0.1$ m/s across the domain in direction $\vec{d}$, an angle α north of east.

arbitrary disk of radius R

$$\begin{aligned} V &= \int_{t=0}^{t=t_f} \int_{\theta=0}^{\theta=2\pi} \int_{r=0}^{r=R} I(r)r \; \mathrm{d}r\mathrm{d}\theta\mathrm{d}t \\ &= 2\pi A t_f \int_{r=0}^{r=R} r \exp\left(\frac{-r^2}{2(R/3)^2}\right) \; \mathrm{d}r \\ &= \frac{2}{9}\pi R^2 A t_f \left(1 - \frac{1}{\mathrm{e}^{\frac{9}{2}}}\right), \end{aligned} \tag{5.8}$$

where t_f is the time where the storm no longer contributes precipitation to the watershed. The volume V of fallen rain is measured in m^3.

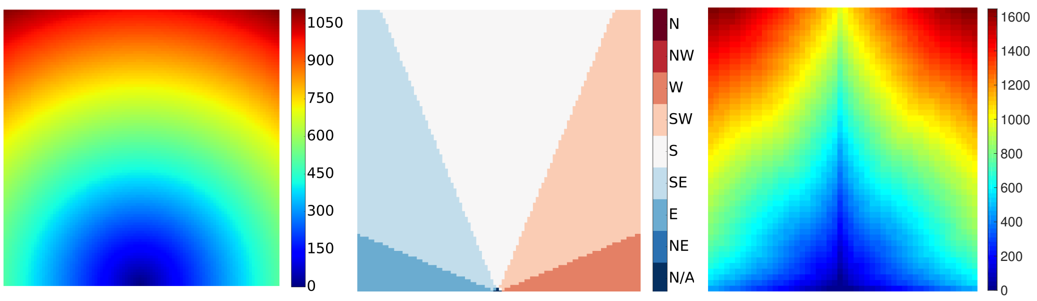
We will compare three different storms where the radii are $R = 10$ m, $R = 20$ m and $R = 30$ m. The maximum intensity for the storm with a radius of 10 meters is $A = 10$ mm/hour. We will use $t_f = 1$ s, i.e., the volume of water that is released from the storm in a second, to determine how the intensity functions for each storm



(a) $R = 10$ m, $A = 10$ mm/hour (b) $R = 20$ m, $A = 2.5$ mm/hour (c) $R = 30$ m, $A = 1.11$ mm/hour

Figure 5.11: The three figures show the hydrographs corresponding to a northeastern movement where the radii are ranging from 10 m to 30 m. Different Gaussian functions ensures that the disk theoretically receives equal amounts of water. Note that the illustration in the top right corner of (c) only shows storm movement, not radius. The radii will vary in the three cases.

will look. We get $A = 2.5$ mm/hour for $R = 20$ m, and $A = 1.11$ mm/hour for $R = 30$ m. All storms start in the bottom left corner of w_b, and move in the northeastern direction until no more cells are affected by the rain. The storm speed is $v = 0.1$ m/s. The intensity functions, as well as their hydrographs are shown in Figure 5.11. A scenario like this is very interesting because it looks at whether large soft storms, or small intense storms are the most dangerous. In the figure we can see that the smallest storm produces the largest discharge peaks, whereas the larger storms create more steady flow at the outlet, but at lower discharge rates. This makes sense, because if a large amount of rainfall is released to an area where the distance to the outlet is close to identical, the probability that affected cells have a similar time-of-flight is very high.



(a) Elevations (b) Flow directions (c) Time-of-flight T

Figure 5.12: We create a 1000 x 1000 meters large landscape to check conservation of water. The landscape (a) is shaped so it drains to the outlet in the middle at the bottom, which the flow directions in (b) show. The time-of-flights (c) are also shown. The watershed covers the entire landscape, and is called w_s.

5.5 Conservation of water

If the total volume of water the rainfall releases to the watershed, is equal to the total volume of water that leaves the watershed at the outlet afterwards, we have conservation of water (mass). This means we must make sure all water we put into the system, eventually goes out of it. We will run a test to confirm that this holds. We construct a 1000 by 1000 meters large watershed where the topography is shaped so the precipitation drains to the same outlet. We call this watershed w_s. The landscape is represented by a 100 x 100 grid, so each cell is 100 square meters large. The elevations of the watershed, the corresponding flow directions and the time-of-flights for the landscape are shown in Figure 5.12. As a side note, the figure actually shows the limitations of the D8 Algorithm in a good way.

We let a storm shaped like a disk cross w_s at a speed of $v = 1$ m/s, and we will calculate the theoretical volume of water V_t from the precipitation. This is shown below in Example 26.

Example 26. We let a disk with radius $R = 20$ m cross watershed w_s. The storm starts just within the top left corner of the grid, and its precipitation will stop when the storm reaches the domain boundary in the bottom right corner. The intensity in the middle of the storm is $A = 2.5$ mm/hour. We want to check the total volume of water the storm releases to the landscape, and we use Equation (5.8) to calculate the theoretical value. As the velocity of the storm is known, we only need to know the distance the storm travels within the watershed. This is $s = \sqrt{960^2 + 960^2}$ m. Because the speed is $v = 1$ m/s, t_f is roughly 1358 seconds. This means that according to

Equation (5.8), we get a theoretical precipitation volume of

$$V_t = \frac{2}{9}\pi R^2 A t_f \left(1 - \frac{1}{e^{\frac{9}{2}}}\right) \tag{5.9}$$

$$= \frac{2}{9}\pi \cdot 20^2 \cdot 2.5 \cdot \sqrt{960^2 + 960^2} \cdot \left(1 - \frac{1}{e^{\frac{9}{2}}}\right) \cdot \frac{1}{10^3 \cdot 3.6 \cdot 10^3} \tag{5.10}$$

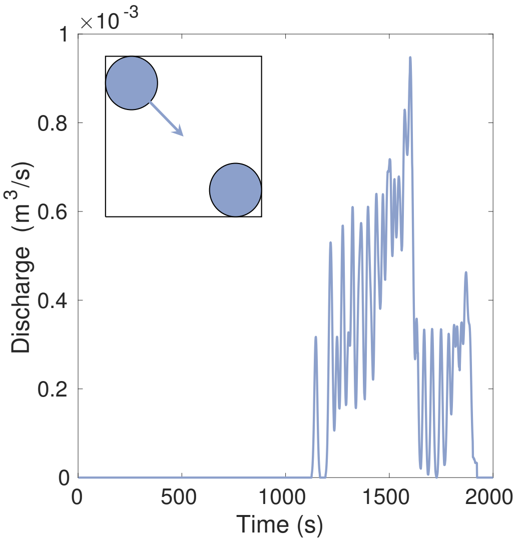
$$\approx 0.2604 \ \ \mathrm{m}^3. \tag{5.11}$$

Now that we know the theoretical volume input V_t, we calculate the numerical volume input V_n to the watershed during the rainfall. If we sum up the contributions in every time step, from the precipitation starts until the disk has reached the opposite corner of w_s, we get $V_n = 0.2602$ m^3. This shows that the water is conserved for the storm. Out of curiosity we also check V_n if we decrease the number of grid points to 50 x 50, and also increase it to 250 x 250 and 500 x 500. First we check the 50 x 50-grid. Our algorithm yields $V_n = 0.2676$ m^3, which is higher than the theoretical volume. We recall the discretization errors we mentioned in Section 5.3, an error which will decrease when the number of cells increase. Thus we try to increase the grid size. When we use a 250 x 250 grid, V_n is much closer to V_t, with 0.2603 m^3. Finally, the 500 x 500-grid has a numerical water input of $V_n = 0.2604$ m^3. It is clear that a finer grid will result in more accurate calculations, but sometimes finer grids are not available, or not practically possible.

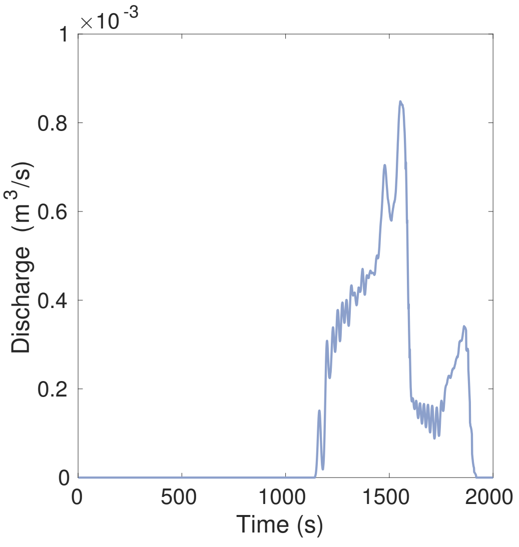
5.6 Grid cell size 'convergence'

In nature, precipitation does not behave as an analytical function. Rain clouds come in all shapes and sizes, and the intensity varies greatly. This means that the error we do when we approximate rainfall over an entire cell is not necessarily that significant. Sometimes a cell receives a bit too much water, other times to little. However, it is still interesting to briefly look at how a finer grid affects the hydrograph from the same rainfall. The storms are identical, so the only difference is how fine the grids are. In Figure 5.13 we can see how the different grid resolutions affect the hydrographs. When the number of cells increase, the rainfall in each cell will decrease. The effect is that the arrival of water at the outlet is more frequent, but the volume per arrival is less. This results in less spikes and smoother hydrographs.

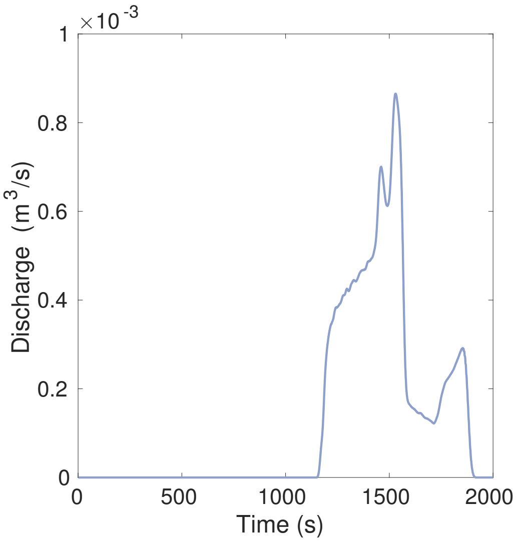
In the next chapter we finally look at the results for a real life watershed.



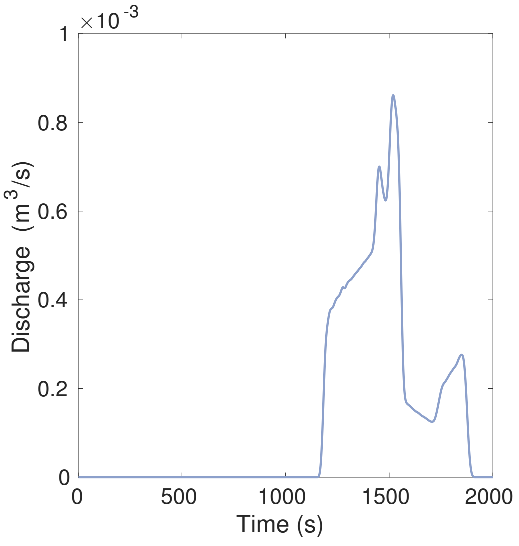
(a) $N = 50$



(b) $N = 100$



(c) $N = 250$



(d) $N = 500$

Figure 5.13: A disk of radius $R = 20$ meters, with maximum intensity $A = 2.5$ mm/hour crosses the watershed w_s in a southeastern direction. The figures show how the number of grid cells N x N affects the hydrographs.

6 Results for real landscape data

Up until now, we have only shown our algorithms for simple examples, but the real fun starts when we apply them to real data and landscape information. In this chapter we will delineate a watershed, calculate the travel times, and use our rainfall-runoff model to predict the discharge in the river after a rainfall.

In this chapter we will look at the landscape that surrounds the fjord Tyrifjorden. To get a better understanding of the topography of the landscape, we show a hillshade plot in Figure 6.1. The elevations of the landscape range between 13.8 meters at the lowest to 710.5 meters at the highest. Because Tyrifjorden acts as a basin for the majority of the landscape, the watershed we will look at will cover most of the hillshade plot.

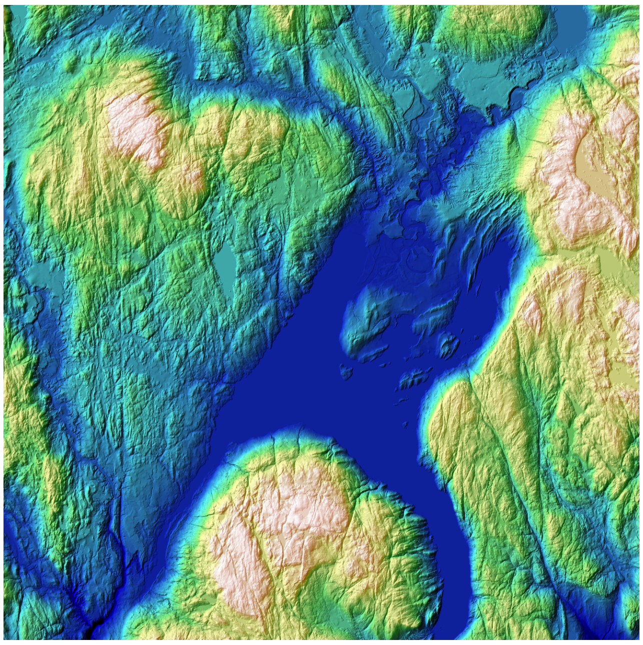


Figure 6.1: A hillshade plot of the landscape. The large flat in the middle is Tyrifjorden.

In the next section we take a look at the flow accumulation results for the entire DEM.

6.1 Flow accumulation

A flow accumulation plot is great for two reasons: it shows all large rivers and lakes in the landscape, and we can use it to select the outlet that we delineate the watershed of. In Algorithm 4 we used the cell connectivity matrix to calculate the accumulated flow in the landscape. If we do this for the Tyrifjorden landscape, we obtain the plot in Figure 6.2. The plots shows the number of upslope cells for each cell in the landscape.

In this particular landscape, the accumulated flow plot does not convey the details of the river network very well, because some areas have a colossal accumulated flow. One example is Tyrifjorden, which has around ten million upslope cells. This amounts to an area of around one billion square meters. Besides Tyrifjorden, we can also see some of the rivers that flow to the lake, in addition to the lakes' outflow river at the south west side of the lake. This river will merge with a tributary river from the west further south.

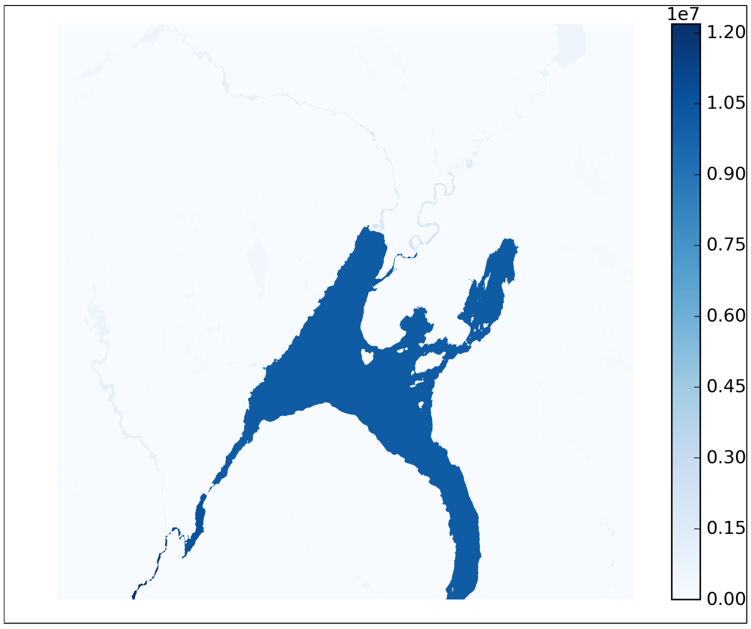


Figure 6.2: Flow accumulation of the Tyrifjorden landscape. The colorbar shows the number of upslope cells for each cell in the landscape; those with the highest flow accumulation have more than 10 million upslope cells.

To better visualize the river network in the landscape, we plot the 10-logarithm of the flow accumulation, which we show in Figure 6.3. The left figure depicts the entire 40 x 40 km landscape, and we can see that major lakes and rivers are much more visible now. In the right figure we zoom in on a section at the south west side of the lake to see how rivers, or the lack of rivers, show the valleys in the terrain. The rivers also show the connections between all lakes, dams and ponds at their maximum water levels.

The flow accumulation is actually one of the parameters that could have been included in κ in Equation (4.7), as a way to increase speed in areas with a large flow accumulation.

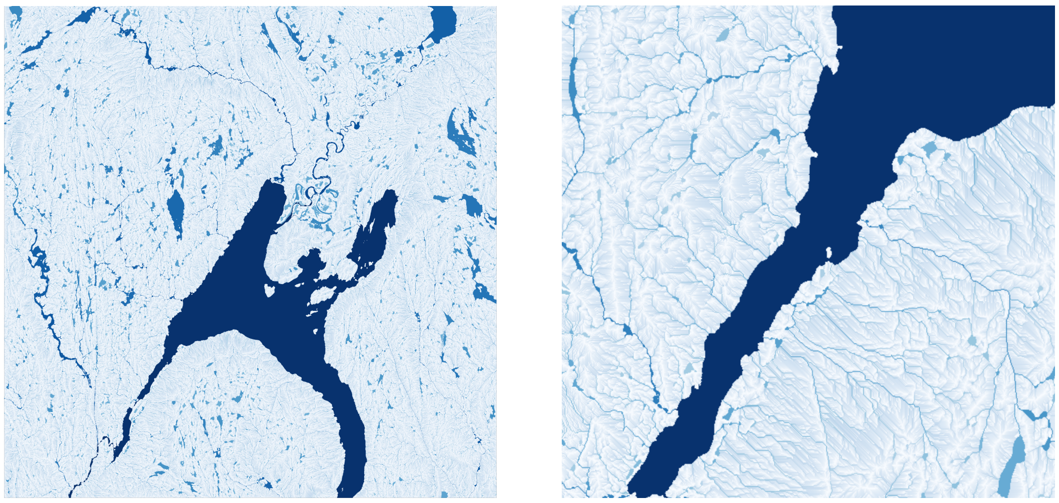


Figure 6.3: To better illustrate rivers and smaller lakes in the Tyrifjorden landscape we take the 10-logarithm of the flow accumulation. The left figure shows the entire landscape, while the right figure shows a small section to the south west of Tyrifjorden. It is much easier to see the river network, and how the lakes are connected in the last figure, compared to Figure 6.2.

Next, we look at the outlet selection strategy. To properly test our implementation we choose the location with the highest flow accumulation in the landscape, which is located in the river/lake at the west side of the southern boundary of the domain in Figure 6.2. The outlet's watershed will collect water from almost the entire landscape in Figure 6.1. In the next section, we will transform the outlet's delineated watershed into the grid structure in MRST.

6.2 Coarse grid from delineated watershed

In Section 4.3 we showed how the watershed w_b could be transformed into a grid in MRST. Although the watershed was small and fictional, its small size made it easy to explain how the time-of-flights had been obtained, and how the hydrographs came to look like they did. We will now do the same for a much larger watershed.

If we choose an outlet in the lower left corner of Figure 6.2 (outlet located in the river at the southern domain boundary), we obtain a watershed that we call w_T. We will transform w_T into the grid structure used by MRST, and call this CG_T, which we show in Figure 6.4. All trap cells have been colored blue, while the non-trap cells are colored grey.

If we plot the elevations of the watershed in 3D, and look from the north, we get the plot in Figure 6.5. The plot gives a different perspective, and the watershed outlet is not located in the upper right corner of the figure.

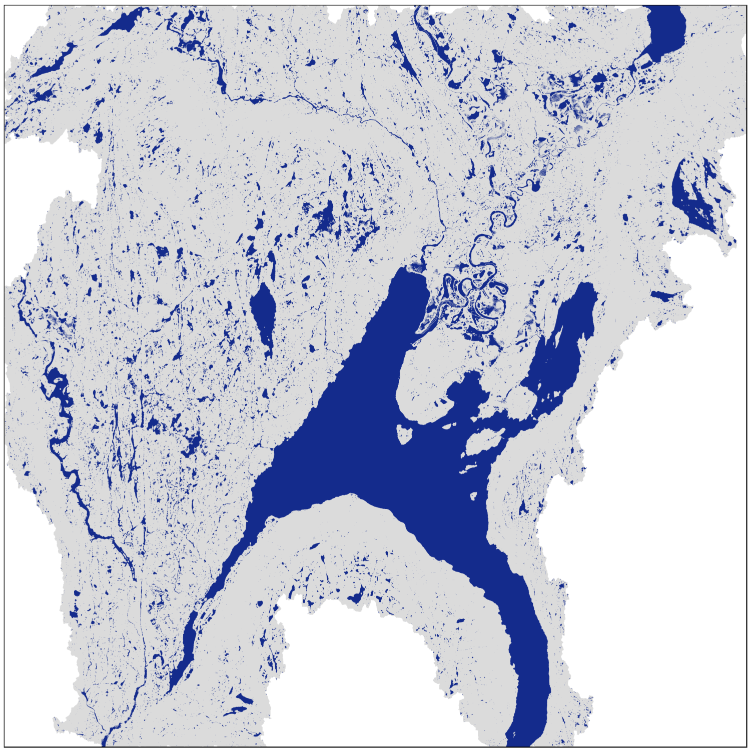


Figure 6.4: The coarse grid CG of the largest watershed in the Tyrifjorden landscape. All trap cells have been colored blue, and the regular cells are grey.

6.3 Time-of-flight for landscape

In Chapter 4 we created a framework that allows us to estimate the travel times for the cells in a watershed, and is the major component of our rainfall-runoff model. This estimate uses a rough approximation of the velocity field $\vec{v}$ in the landscape based only on topography, as we set $\kappa = 1$ in Equation (4.8). Because of the framework we have created, it is possible to account for the heterogeneous properties by adjusting κ, and obtain more accurate velocity fields. This will in turn yield better estimates of the time-of-flights. There are many possible improvements that can be made to the model, and we will come back to some of them in the conclusion and future work.

In Chapter 4 we obtained time-of-flights for the small watershed w_b, but we will now do it for the cells in watershed w_T. Because some of the trap cells are huge, we set $\phi = 10^{-7}$. This will ensure a high speed in the Tyrifjorden trap cell, which enables water to quickly flow from one end of Tyrifjorden to the other end. If we had not done this, the trap cell would have been a major bottleneck in the watershed. In Figure 6.6 we show the time-of-flights for watershed w_T. The travel times are given in seconds.

In Figure 6.6 the time-of-flights range between $T = 0$ and $T = 699004$ seconds.

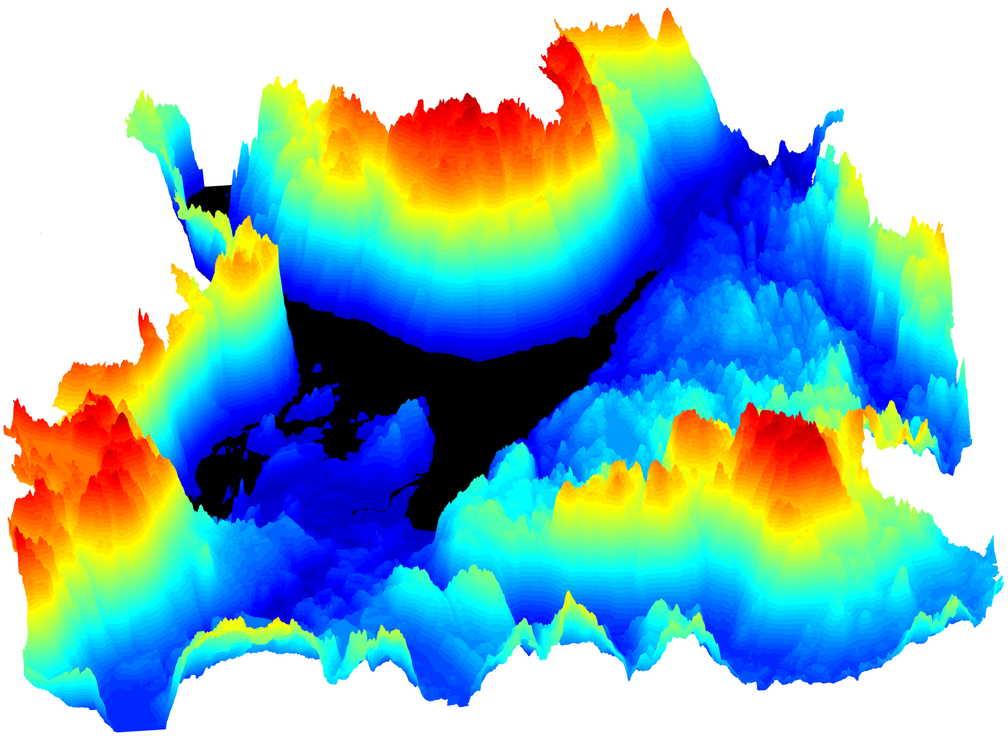


Figure 6.5: A 3D-plot that shows the elevations of the Tyrifjorden watershed seen from the north to give another perspective on the area. We have colored Tyrifjorden black so it is easier to see.

This means that the cell with the longest travel time takes about 194 hours to arrive at the outlet, which is longer than a week. This sounds like a very long time. It is not an easy task to determine how long it takes water to flow to the outlet, because there are so many parameters that will alter the estimate. To obtain a better estimate of the travel times, we would have to change our velocity field. And perhaps the constant C we used over faces with zero or negative elevation difference also need to be changed.

If we have a side-by-side comparison of the elevations in the landscape and the time-of-flights, it is easier to see how they are related. In Figure 6.7 we plot the time-of-flights to the left and the elevations to the right using only 15 colors to make the comparison easier. In general, we can see that the time-of-flight is low for areas close to Tyrifjorden, which also have a steep slope towards it. When the distance to Tyrifjorden increases, and the slope is gentle, the travel time also increases. In this particular watershed Tyrifjorden will have a huge impact on the travel times, and the close to instantaneous movement across makes the distance to Tyrifjorden a very important factor.

To explain the high time-of-flight for the area located a bit south of the upper right corner, we can look at Figure 6.5, where Tyrifjorden is colored black. The

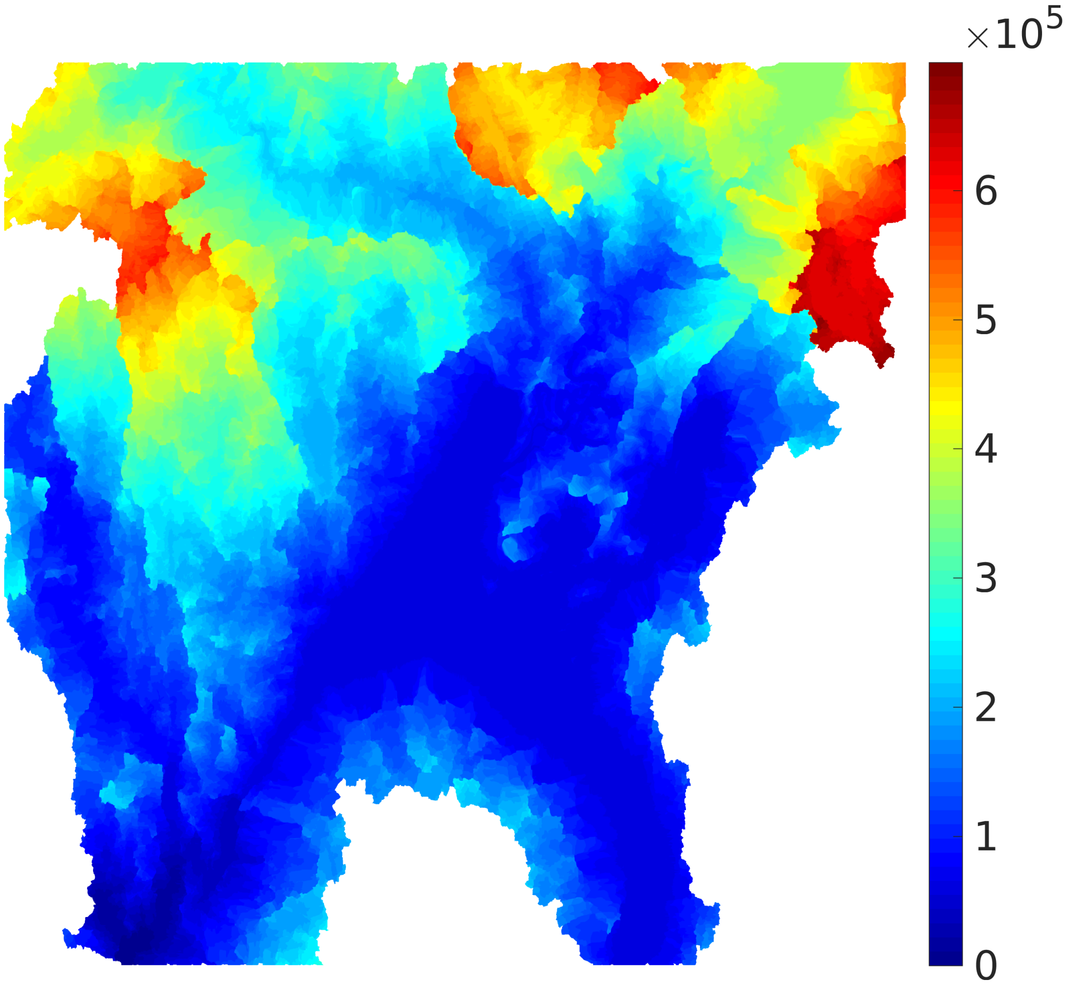
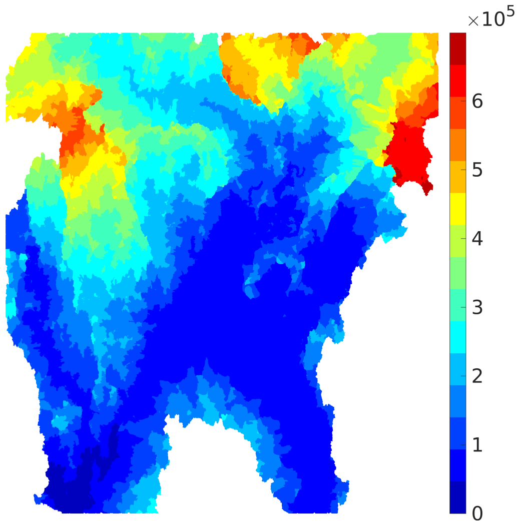


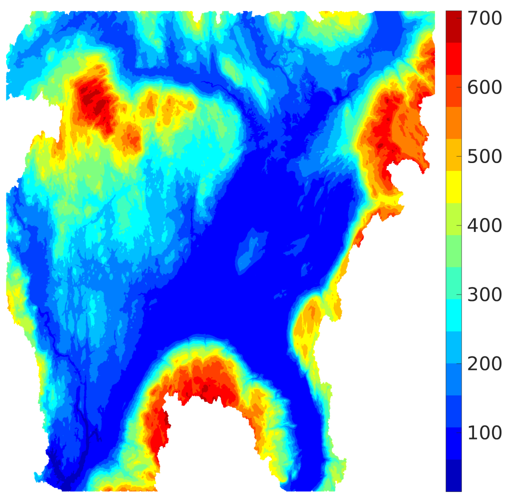
Figure 6.6: The time-of-flight for w_T when $\phi = 10^{-7}$ for traps.

area in question is shown in the lower left corner, and we see that it is far away from Tyrifjorden. There will be other trap cells it has rapid flow in, but it takes some time to get to them, and parts of the area within the mountains are quite secluded. The plot also shows the gentle slopes in the watershed, which explains higher time-of-flights.

When the trap cells grow as large as the Tyrifjorden cell, the ϕ-parameter will have a lot to say. We will compare a choice of $\phi = 10^{-3}$ to $\phi = 10^{-7}$ for the trap cells in Figure 6.8. The left figure shows the largest value, and it is clear that there is a bottleneck in the Tyrifjorden trap cell, as the speed is very slow in this area. The plot clearly shows the part of the watershed that does not go via Tyrifjorden, i.e., the watershed of the river that merges with the river from Tyrifjorden.



(a) Time-of-flights



(b) Elevations

Figure 6.7: Comparison of time-of-flights and elevations for watershed w_T, when $\phi = 10^{-7}$ for trap cells. To make the comparison easier, we have only used 15 colors in the plots.

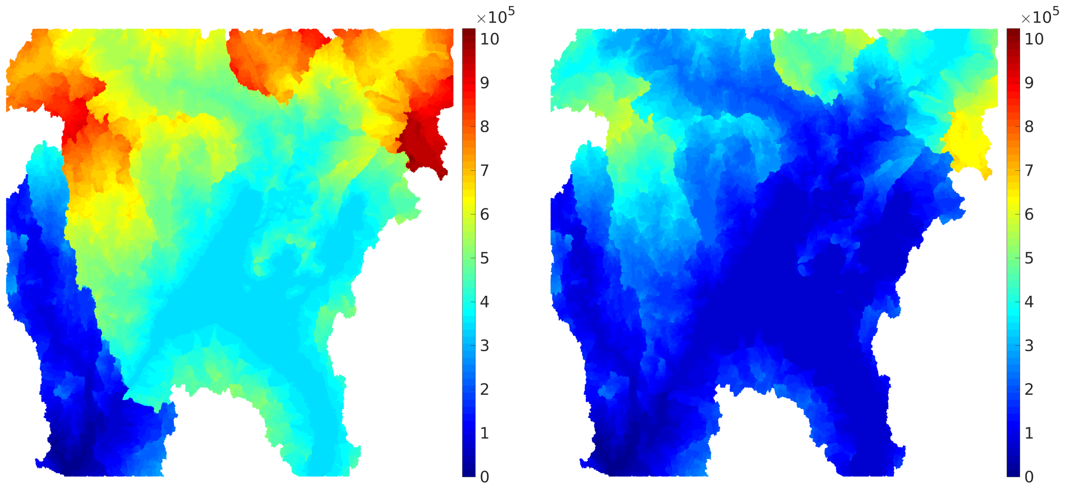


Figure 6.8: A comparison of how the time-of-flights change when $\phi = 10^{-3}$ (left) and $\phi = 10^{-7}$ (right) for trap cells.

6.4 Optimization of hydrograph calculations

The computation of the hydrograph can take a very long time if the watershed and the storm are large. This is because we have to calculate which cells the precipitation affect in each time step. If we had used real life-precipitation data in a raster format, this would not be a problem. One way we can speed up the computations, is to increase the Δt from Section 5.2. Up until now we have used $\Delta t = 1$ seconds, which means constant precipitation for 1 second before the rainfall changes location. From a weather simulation perspective, this is a very low Δt, because the changes that happen from one second to the next are very small. As an example, The Norwegian Meteorological Institute[1] uses a Δt of 7.5 minutes for their radar images that visualize precipitation.

When Δt increases, we let the discharge from each cell be constant for a duration of Δt seconds. We will now look at different Δt-values to see how much the hydrographs change, and we let the hydrograph with $\Delta t = 1$ be our reference solution. Because we let the precipitation be constant in our computations, the hydrograph will look like a discontinuous stepwise function. In Figure 6.9 a storm front of width 10 km crosses w_T at a speed of $v = 1$ m/s in the northern direction, with a maximum intensity in its storm center of $A = 10$ mm/hour. The left figure shows a good overlap between the hydrographs. However, when we zoom in on the hydrographs, shown in the right figure, we can clearly see the discontinuities, and how the 'steps' grow smaller when Δt decreases.

To smooth the discharges in Figure 6.9, we average the discharge for each time interval $[t_n, t_{n+1}]$, and plot it in time $t_{n+\frac{1}{2}}$. The resulting hydrographs are shown in Figure 6.10. The left figure shows the entire hydrograph, while the right figure shows the same time interval as the right figure in Figure 6.9. We see that the functions are much smoother now and closer to the reference solution. The hydrograph that $\Delta t = 60$ seconds produces is almost identical, and $\Delta t = 600$ seconds is also very good. When Δt is as large as 3600 seconds, we see its hydrograph is not ideal; from the small example in the right figure the discharge is off by more than 5 m^3/s.

We had some struggles with this process because we used the precipitation's position in time t_n as basis for the next Δt seconds. This made the hydrographs lag a distance $\Delta t/2$ behind our reference solution ($\Delta t = 1$), which is shown in Figure 6.11. When we use the rainfall's position in time t_n as basis for the interval $[t_n, t_{n+1}]$, the hydrographs will only agree with the reference solution in time t_n. A better solution is to instead use the storm front's position in time $t_{n+\frac{1}{2}}$ to estimate the rainfall. This way the hydrographs follows the reference solution much more closely.

We want to increase the time step Δt to something that still qualitatively gives the same hydrograph, as this will optimize our running time. We want to figure out the error we make when we discretize the precipitation and consider it piecewise

[1]Website at met.no.

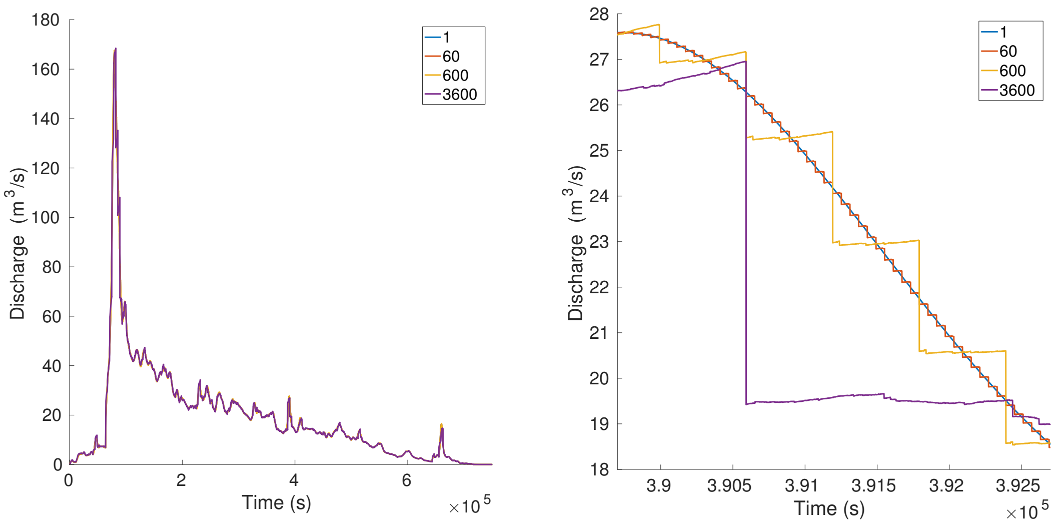


Figure 6.9: Front movement towards the north at a speed of $v = 1$ m/s. The left figure shows the resulting hydrographs if the precipitation stands still for $\Delta t = 1$, 60 and 600 seconds. The precipitation in time $\Delta t/2$ contributes a constant discharge in the time interval $[0, \Delta t]$. In the right figure we see the discontinuities in the hydrographs.

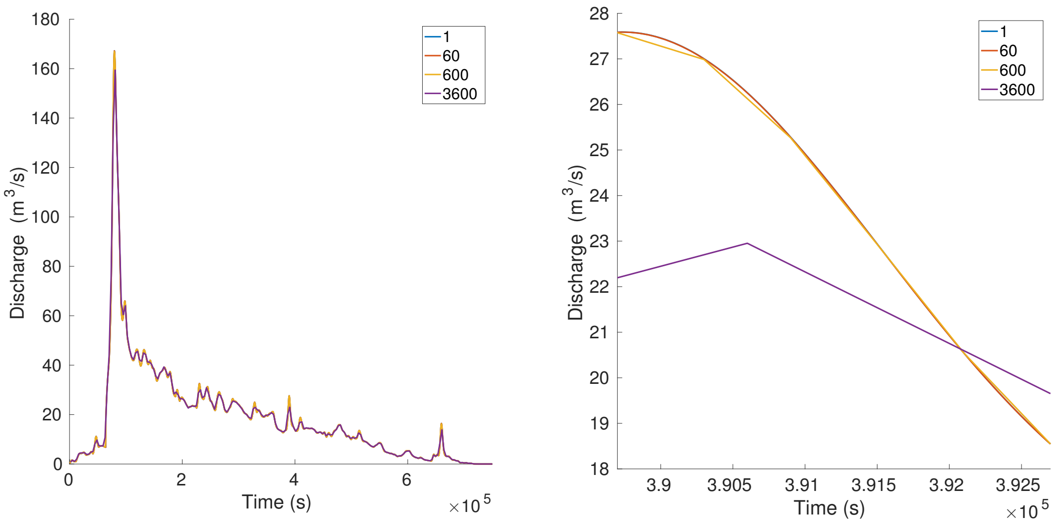


Figure 6.10: If we let the precipitation be constant for Δt seconds before we move it, the discharge from each cell will be constant for the same duration. We can smooth the resulting hydrograph by averaging the discharge over every Δt seconds, and plot the discharge in time $t_{n+\frac{1}{2}}$. The left figure shows the result after smoothing the data. We can compare the right figure with the right figure in Figure 6.9 to see how it was before the smoothing.

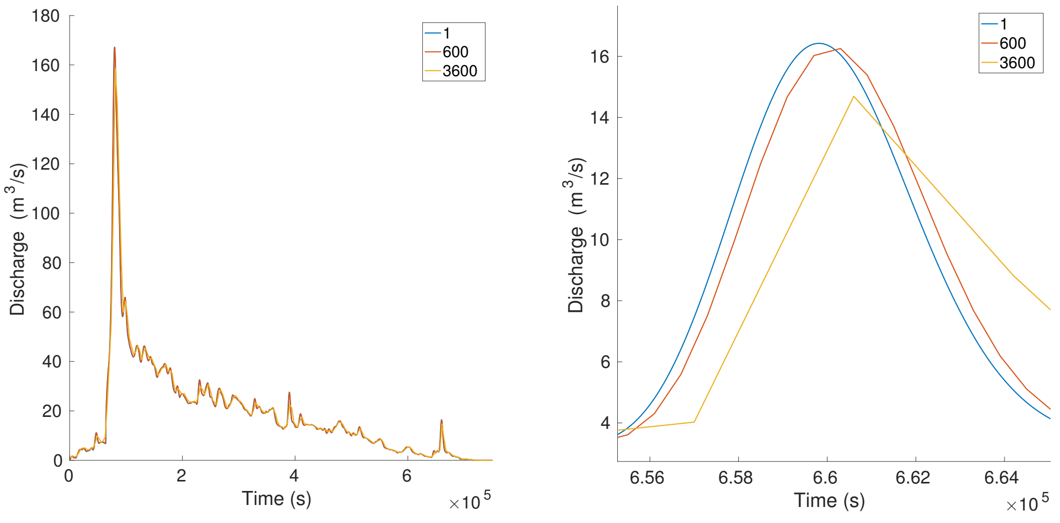


Figure 6.11: Initially we used the storm front's position in time t_n as basis for the time interval $[t_n, t_{n+1}]$. This means that the precipitation only agrees with the reference solution in time t_n, which leads to a lag in the hydrographs by $\Delta t/2$. If we instead use the storm front's position in time $t_{n+\frac{1}{2}}$, we avoid this effect.

constant. We expect the error to be $\mathcal{O}(\Delta t)$ from

$$P(t, T(x)) \approx P(t_c) + P'(t_c)(t - t_c) = P(t_c) + \mathcal{O}(t_c),$$

where t_c is

$$t_c = t_{n+\frac{1}{2}} = \left(n + \frac{1}{2}\right)\Delta t.$$

To show this, we calculate the norm of the error $e(t) = h(t) - h_{\Delta t}$, where $h(t)$ is the function in Equation (5.1). The approximate hydrograph $h_{\Delta t}$ is the resulting hydrograph from a piecewise constant precipitation, where the discharge is averaged over each Δt-interval. We define a norm of the error we make globally as

$$\mathcal{E}_2 = \frac{||e(t)||_{L^2}}{||h(t)||_{L^2}}, \tag{6.1}$$

where $||e(t)||_{L^2}$ is a discretized version of the L^2-norm:

$$\begin{aligned}||e(t)||_{L^2} &= \sqrt{\int_0^{t_{max}} |h(t) - h_{\Delta t}(t)|^2 \,\mathrm{d}t} = \sqrt{\sum_n \int_{t_n}^{t_{n+1}} |h(t) - h_{\Delta t}(t)|^2 \,\mathrm{d}t} \\ &\approx \sqrt{\sum_n \left|h(t_{n+\frac{1}{2}}) - h_{\Delta t}(t_{n+\frac{1}{2}})\right|^2 \Delta t} = \sqrt{\Delta t}\sqrt{\sum_n \left|h(t_{n+\frac{1}{2}}) - h_{\Delta t}(t_{n+\frac{1}{2}})\right|^2}. \end{aligned} \tag{6.2}$$

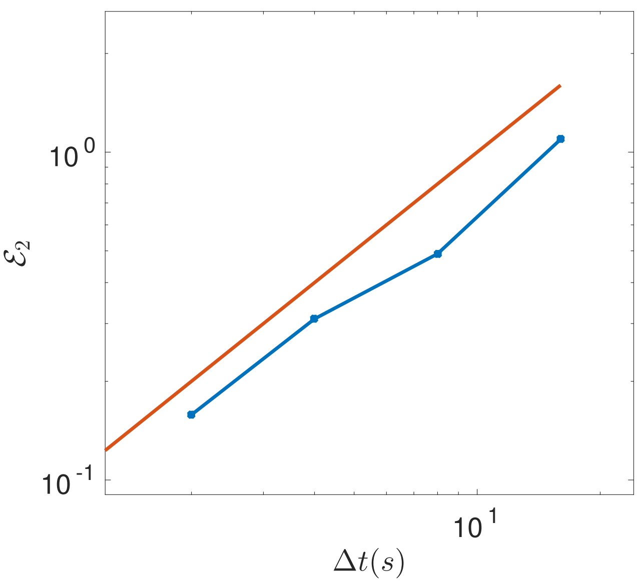


Figure 6.12: The plot shows the $\mathcal{E}_2$-norm from Equation (6.1), which is the global error we make when we use a piecewise constant precipitation in Δt-intervals. The error scales by $\mathcal{O}(\Delta t)$.

The norm of our reference solution is discretized in a similar manner. We want to confirm that $\mathcal{E}_2$ scales by $\mathcal{O}(\Delta t)$. The example watershed w_b from Chapter 4 is used to show this. In Example 27 we let a storm front cross w_b in a northern direction. It is important to use Δt-values such that the rainfall does not jump over cells, i.e., $v\Delta t < w + c$, which we can write as

$$\Delta t < \frac{w+c}{v}, \tag{6.3}$$

where w is the front width, c the width of a cell, and v the velocity of the front. If our choice of Δt does not satisfy (6.3), we will entirely jump over cells, and get a poor hydrograph approximation.

Example 27. We consider $h_{\Delta t=1}(t)$ our reference solution, which we will compare to $h_{\Delta t}(t)$. We use the Δt-values $\{2, 4, 8, 16\}$. In this example the velocity of the storm front is $v = 1$ m/s, and the front width is $w = 25$ m wide. Based on Equation (6.3) we could also have included $\Delta t = 32$, but because it is so close to the threshold, we will omit it. In all cases $A = 10$ mm/hour and $\phi = 0.1$ for trap cells. If we plot the error $\mathcal{E}_2$ in Figure 6.12, we see that it scales by $\mathcal{O}(\Delta t)$.

We will now visually inspect which Δt-values that are appropriate at different velocities v for the Tyrifjorden watershed w_T, if a storm front of width $w = 10$ km with a Gaussian distribution crosses the watershed in a northernly fashion. In all cases we have used $\phi = 10^{-7}$ for trap cells to calculate the time-of-flight, and a

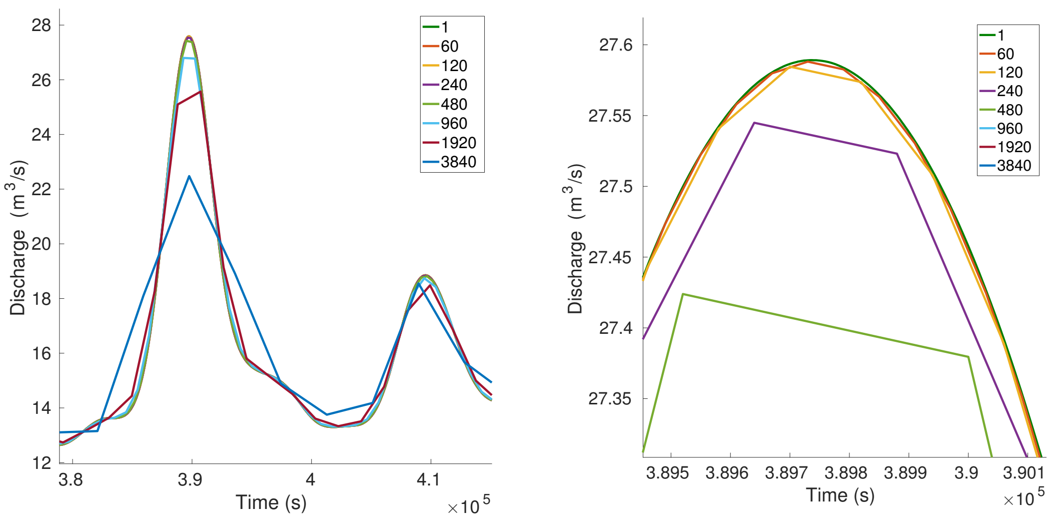


Figure 6.13: We will compare the hydrographs if we move the precipitation in every second ($\Delta t = 1$ s), to if we move it every Δt seconds. In the left figure we show a section of the hydrograph where it is easy to see that some of the Δt-values fail to give a good approximation of the reference hydrograph, in this case especially $\Delta t = 3840$ seconds. In the right figure we zoom in even more and see that e.g., Δt-values of 60 and 120 result in very close fits to the reference solution.

precipitation intensity maximum of $A = 10$ mm/hour. We first check the case $v = 1$ m/s, and we use the Δt-values $\{60, 120, 240, 480, 960, 1920, 3840\}$ seconds, which all satisfy Equation 6.3. Just like in the left figure in Figure 6.10 they look similar from afar, so we will zoom in to look at the differences. In the left figure in Figure 6.13 we can see that especially $\Delta t = 3840$ performs poorly compared to the other Δt-values. If we zoom further in we get a closer look at the hydrographs the other Δt-values produce. At this level the discharges are not far from the reference solution in terms of discharge.

Criteria for Δt-selection

Because we need to calculate hydrographs for different velocities v, we need some criteria to base our selection of Δt on. The first one we will use is a threshold on the error $\mathcal{E}_2$, which acts as an estimate for global error. This will ensure an overall good fit with our reference solution. The second criterion is a threshold on the supremum norm, which we call $\mathcal{E}_\infty$, and is given as

$$\mathcal{E}_\infty = ||e||_{L_\infty} = \sup_{t \in [0, t_{max}]} |h(t) - h_{\Delta t}(t)|. \tag{6.4}$$

In order to get hydrographs where the discharge peaks closely resemble each other, we need to know if there are any local points where the error is extra large. Sometimes it is very important that the maximum discharge is correct, one example being

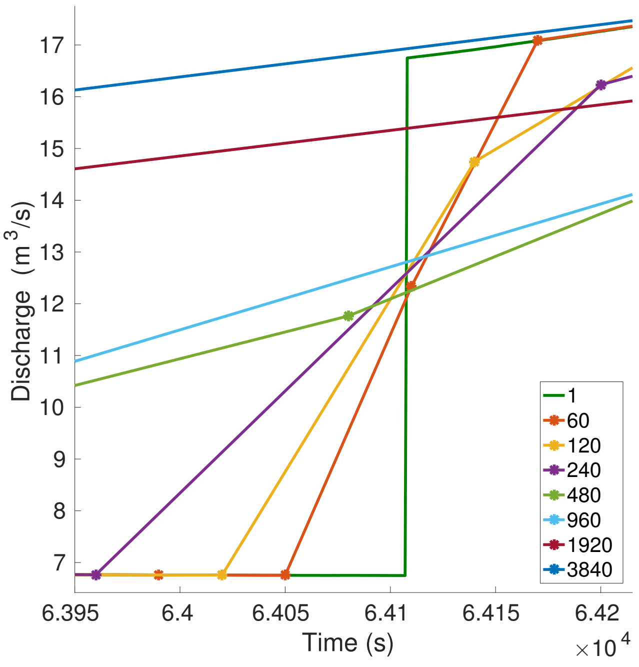


Figure 6.14: Discontinuity in the hydrograph of the reference solution. The leap occurs because the initial position of the storm front is halfway onto the watershed, which includes a section of Tyrifjorden. Because the entire fjord has a time-of-flight of 64107 seconds, the water will arrive simultaneously at the outlet. This results in a large error for small Δt-values.

the design flood estimations we mentioned briefly in Section 2.4. If we know that a structure is only capable to withstand a discharge of 200 m^3/s, it is important that we get the maximum discharge correct. Thus, our third criterion is that the difference between the maximum discharge in the reference solution and for our approximated solution is below a threshold.

We will start by looking at the error $\mathcal{E}_2$ for the different Δt-values. After we calculated the norm for the different Δt-values, we noticed that small Δt-values performed surprisingly poorly. Upon closer inspection we found a discontinuity in the hydrograph of the reference solution, which occurs because the storm front's initial position is halfway onto the watershed. The storm front contains a part of Tyrifjorden, which has a time-of-flight value of 64 107 seconds. This fits well with the discontinuity shown in Figure 6.14. As we can see from the figure, the discontinuity causes the hydrographs for e.g., $\Delta t = 60$ and $\Delta t = 120$ to have a larger error than the hydrograph for $\Delta t = 240$. To get more accurate estimations of the norm, we instead calculate the norm based on the discharge values after the discontinuity.

We will now show an example where we select a Δt for a velocity of $v = 1$ m/s. We explain the reasoning behind the choice we make of Δt in Example 28. The thresholds we present in the example are the same ones that we will use for different velocities.

Example 28. The first criterion we check is the $\mathcal{E}_2$-norm. To ensure a low global error we set a threshold of 0.01. For a speed of $v = 1$ m/s, the only Δt-values that yield a $\mathcal{E}_2$-value below the threshold are 60, 120, 240 and 480 seconds; these Δt-values give hydrographs that come very close to the reference solution in the right figure in Figure 6.13. Next we look at $\mathcal{E}_\infty$, where we set a threshold of 2 m^3/s. This means that we do not allow a local error in the discharge of more than 2 m^3/s. All Δt-values beside 1920 and 3840 are below this threshold, which means that we keep all Δt-values that passed the first criterion.

In the final criterion, we compare the difference in maximum discharge for the hydrographs, and we will allow a difference of maximum 2 m^3/s. Only the two largest Δt-values fail this test. This means that based on our three criteria, we can choose Δt-values of either 60, 120, 240 or 480 seconds. To get an accurate hydrograph, but at the same time a low computation time, we choose the largest feasible Δt, which in this case is $\Delta t = 480$ seconds. This shows that we can use a stepwise constant function, where each step is eight minutes long, and still get a sufficiently good hydrograph.

If we use the three criteria for the velocities $v = 5.6$ m/s and $v = 12.5$ m/s, we obtain maximum Δt-values of 240 seconds, and 120 seconds, respectively. This shows that rainfalls moving at a higher speed, requires more data to obtain an accurate hydrograph, which is not surprising. In the following we will use a Δt-value of 480 seconds for $v = 1$ m/s, $\Delta t = 240$ for $v = 5.6$ m/s, and $\Delta t = 120$ for $v = 12.5$ m/s. The chosen speeds are not random, but represent warm ($v = 5.6$ m/s) and cold ($v = 12.5$ m/s) storm fronts. Based on these results, we hope that The Norwegian Metereological Institute only uses a $\Delta t = 7.5$ minutes for their visualizations, and not their calculations.

It is also possible to use the accumulated flow as an extra criterion. In Figure 6.15 we have plotted the accumulated flow for the three velocities $v = 1$ m/s, $v = 5.6$ m/s and $v = 12.5$ m/s, from left to right. The plot shows how the highest Δt-values completely fall through in terms of estimating the total discharge when the speed increases. Thus we must reduce Δt if v is high to get accurate precipitation data. We can also see that the maximum accumulated flow decreases when the velocity of the storm front v increases. This is natural as the storm front moves over the landscape quicker, so it sheds less water. To obtain accumulated flow we use linear interpolation between the points of the $h_{\Delta t}(t)$-functions, which gives us discharge values for every second.

6.5 Hydrographs

The optimization we looked at in Secion 6.4 do not apply to the scenarios with uniform precipitation in the entire watershed. Rather, we use it in scenarios where

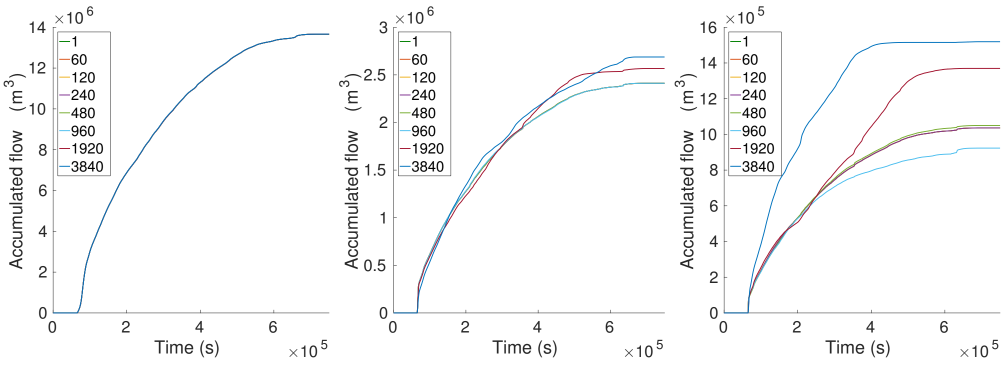


Figure 6.15: Accumulated flow over time when the rainfall contributes to the discharge. In the left figure the velocity of the stormfront is $v = 1$ m/s, in the middle one $v = 5.6$ m/s, and the right one is $v = 12.5$ m/s. When v increases, we must decrease Δt to capture the required information. We can also see that the total rainfall decreases when the speed increases.

the precipitation takes the form of a storm front or a disk, which moves at a constant speed. In Figure 6.16 we look at two hydrographs after a uniform rainfall of $P = 10$ mm/hour in the entire watershed. If the duration of the rainfall is one hour (left figure), Tyrifjorden will contribute with a very large discharge peak after approximately 20 hours, but the effect of regions that simultaneously contribute discharge is much more noticeable if the duration of the rainfall increases. A duration of ten hours (right figure) gives a higher maximum discharge, as well as an increase in the overall discharge at the outlet. Because of the large rainfall in the entire watershed, the outlet will experience a much larger discharge compared to non-stationary rainfalls which affect only parts of the watershed.

A uniform precipitation scenario in an entire landscape is something that do not happen very often, so we will look at scenarios where the precipitation varies in shape, intensity and location, just like we did in Chapter 5. We will now take a look at storm fronts that move across w_T. In all examples we use a storm front of width $w = 10$ km, and a precipitation intensity that varies in the storm front according to Equation (5.6), where $A = 10$ mm/hour and $R = w/2$. In the following we will vary both the direction of travel and the speed at which it travels.

We begin with a comparison of the hydrographs for a storm front that travels in the directions north, south, west and east. When the storm front starts its movement further away from the watershed outlet, we expect peaks to come earlier, and be somewhat larger, because the movement of the rainfall, and the flow of the water, are in the same direction; this is an effect we noticed in our hydrographs in Section 5.3. In Figure 6.17 we notice this effect as well. The top left figure shows the northern movement, while the top right shows the southern movement, and we can see that the latter has a larger maximum peak. Likewise, the maximum peak of a western movement (bottom left) is larger than the eastern movement (bottom right). Besides

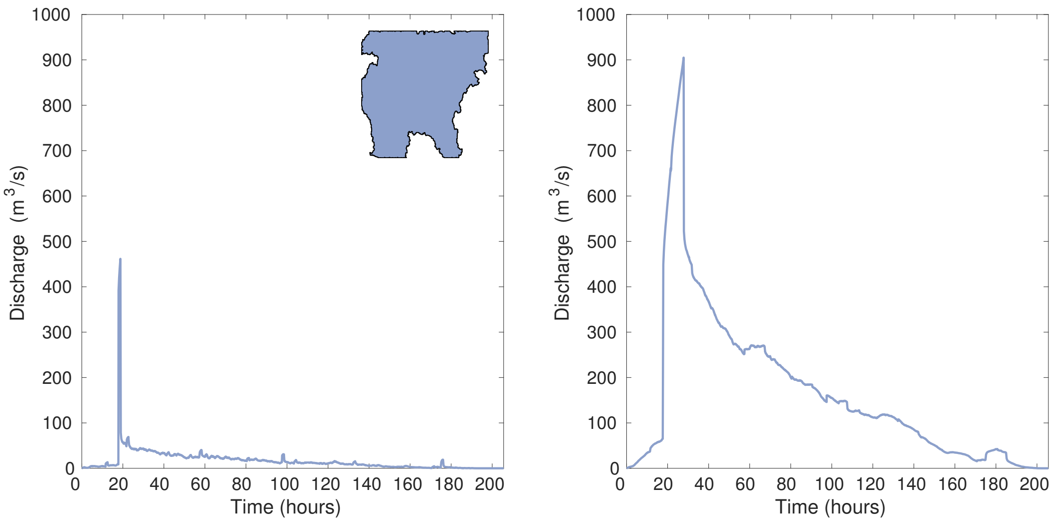


Figure 6.16: A uniform precipitation of $P = 10$ mm/hour in the entire watershed lasts for one hour (left figure) and ten hours (right figure). In both cases the contribution from Tyrifjorden creates the largest discharge peak, but when the rainfall only lasts for an hour, there will not be much overlap between the discharge from different regions. In the right figure this is much more noticeable; both peak discharge and the overall discharge at the outlet will increase because more water arrive simultaneously.

these effects, the hydrographs look very similar, most likely because the storm front crosses w_T relatively fast compared to the time-of-flights.

The various peaks in the hydrographs usually come from large lakes (where all cells in the lake has the same time-of-flight T). If we look at the five largest traps in the area, the Tyrifjorden trap has a time-of-flight of 17.8 hours, and is over 26 times larger than the second trap on the list. This explains the huge difference in peak heights. The other four traps have time-of-flight values (in hours) of $T = 97.5$, $T = 175.0$, $T = 57.4$ and $T = 21.9$, where the largest trap is listed first. In all four figures we can make out the contribution from the first, second and third largest lakes. The fourth is also quite visible, while the fifth is somewhat masked by the Tyrifjorden trap.

In Section 5.3 we saw that a slower moving storm front resulted in taller discharge peaks, a consequence of increased overlap between discharge contributions from different regions. The total volume of rain will also increase, as the cells are exposed to precipitation for longer periods. We will compare three velocities for a storm front that moves towards the west. The first one, $v = 1$ m/s is more of a reference solution, while the other two represent typical velocities for cold and

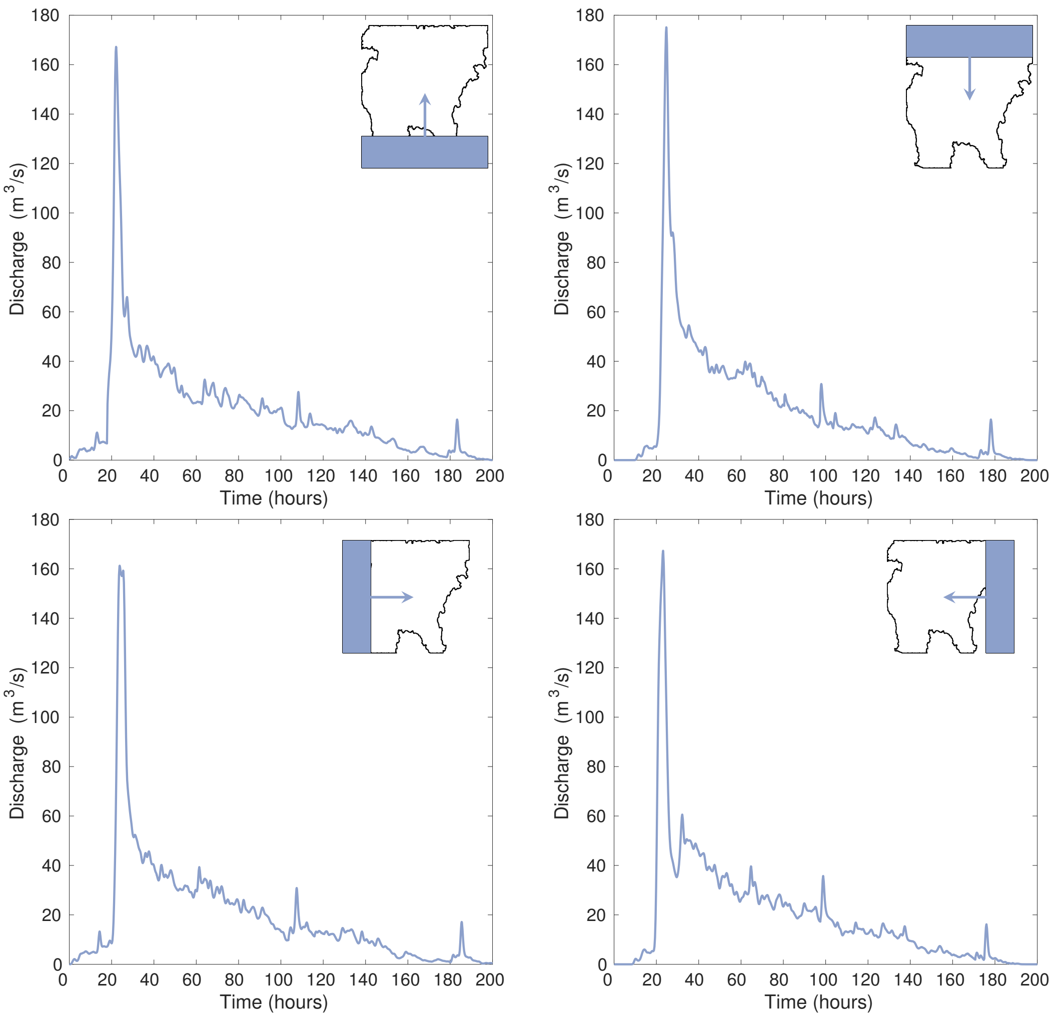
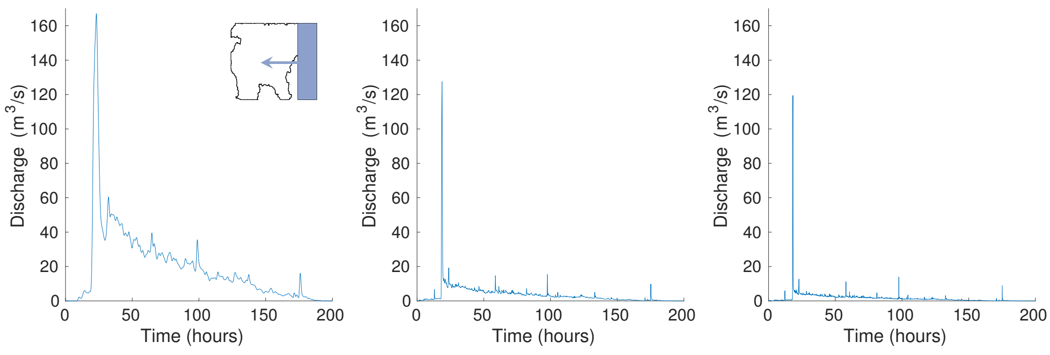


Figure 6.17: A storm front sweeps across the watershed at a speed of $v = 1$ m/s. When the storm front comes from different directions, the hydrographs will differ from each other. Here we show a storm front that moves northwards (top left), southwards (top right), eastwards (bottom left) and westwards (bottom right). In each scenario the storm front starts with half the front at the inside of the watershed, and the storm has passed when the entire front has left the watershed.



(a) $v = 1$ m/s (b) Warm front $v = 5.6$ m/s (c) Cold front $v = 12.5$ m/s

Figure 6.18: Resulting hydrographs for a storm front that crosses w_T from the east (western movement) at different velocities.

warm weather fronts[2]. To represent a warm weather front we use $v = 5.6$ m/s, and $v = 12.5$ m/s for a cold weather front. In Figure 6.18 we compare the three speeds.

In Chapter 5 we also modeled the precipitation as a moving disk. Perhaps the most interesting aspect of this was the question about the impact of a disk shaped storm with a large radius, but low maximum intensity versus a storm with relatively small radius, but larger maximum intensity. We will study this for w_T as well. In Figure 6.19 we compare the hydrographs of disk-shaped rainfalls where we vary radius and maximum intensity. We use the radii 5000 m, 3000 m and 1000 m. If we assume a maximum intensity $A = 10$ mm/hour in the largest disk, we get a maximum intensity of $A = 250$ mm/hour for the smallest disk, which is not so realistic, and very extreme. However, to keep the total volume of rainfall for all three disks equal, we will still use it.

The plots have some differences. It is especially interesting to see that the maximum discharge increases when the radius grows smaller. This is expected as it increases the chance that an area with approximately the same time-of-flight receives a lot of precipitation simultaneously. If this happens in a narrow valley, this can have large consequences for flow at the watershed outlet. We have only looked at the hydrograph for the entire watershed outlet in w_T, but if we had also looked at the discharge for the said valley, the impact of this local rainfall would be much larger.

If an area with similar time-of-flights receive large amounts of precipitation simultaneously, this can result in a large discharge peak. We saw an example of this effect in Figure 6.19, where the maximum discharge increased when the area of the rainfall decreased (constant volume). One type of precipitation that can have a

[2]In an article from yr.no called 'Hvor fort går egentlig været?', they discuss typical speeds of weather fronts, and estimate cold weather fronts to have a speed between 30-80 km/h, and warm weather fronts a speed around 20 km/h.

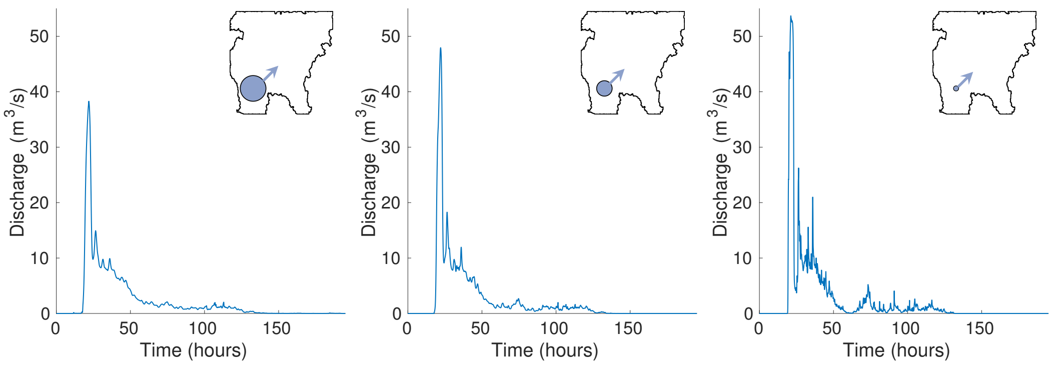


Figure 6.19: The hydrographs after three disk-shaped rainfalls cross watershed w_T at a speed of $v = 1$ m/s. The volume of water contributed to w_T is equal in all three cases. The left figure shows a disk with radius $R = 5000$ m, and maximum intensity of $A = 10$ mm/hour. When the radius grows smaller (left and right figure), the intensity in the disk center increases. The middle figure shows the resulting hydrograph from a rainfall of radius $R = 3000$ m and maximum intensity $A = 27.8$ mm/hour, while the same data for the right figure is $R = 1000$ m and $A = 250$ mm/hour.

very large impact on the terrain is the convectional rainfall[3], which is very local and often quite intense. When it occurs in cities, it often results in floods because of the many inpermeable surfaces. In Example 29 we look at an event where convectional precipitation in the Oslo area flooded parts of the city.

Example 29. In August 2016 convectional precipitation caused parts of Oslo to flood. It was reported that 54.7 mm rain fell in the course of two hours[4]. Based on the article's table of precipitation in different parts of the city, the rainfall intensity varies greatly in the different parts of the city. We want to simulate the rainfall by using a small disk of radius $R = 1000$ m, and a maximum intensity of $A = 27.35$ mm/hour. The rainfall will last for two hours.

In Figure 6.20 we compare the hydrographs for a uniform and a Gaussian distributed rainfall in the disk. We have used the same location for the disc center as in the right figure in Figure 6.19. The two different distributions will change the intensity the different cells experience, which will change the hydrographs. Because the rainfall is so intense and concentrated around an area, it is possible that the discharge is closer to the hydrograph obtained from a uniform distribution, than the one we obtained with a Gaussian distribution.

[3]Convectional rainfalls do not last long, but are often very intense. The sun heats up the landscape, which causes warm, moist air to rise. The air is then cooled high up in the atmosphere, so the water condenses and forms clouds.

[4]This is reported in the article from yr.no titled 'Meteorolog etter ekstremregn:- Vi varsler ikke godt nok' from August 2016.

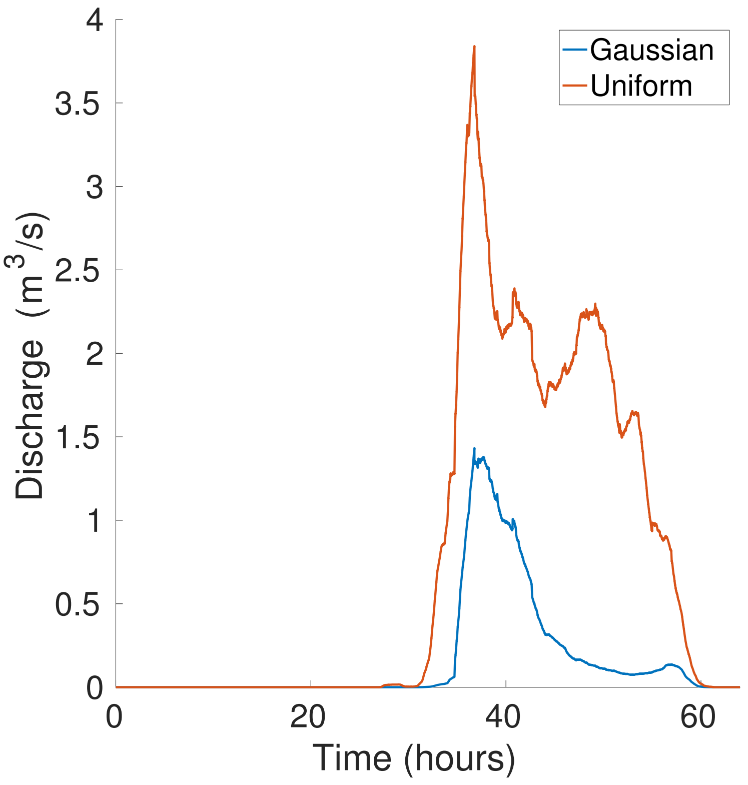


Figure 6.20: If we compare a stationary Gaussian distributed disk of rainfall, to one with a uniform distribution, we obtain different hydrographs. The intensity in the affected cells will change the shapes of the hydrographs.

Obviously the impact in a city will be larger than in our landscape, as the dimensions of infrastructure can not compete with the dimensions of rivers in the landscape. There are also much more permeable surfaces in the terrain, and more vegetation, but we have not accounted for this yet, as we view our landscape as impermeable.

The rainfall-runoff model that we have developed in this Master's thesis can be very useful when paired with data from weather forecasts, as it lets us predict future river discharge. If the river has an upper limit to how much water it can handle (before it floods), we can check if this threshold will be surpassed by the discharge from the forecasted rainfall.

7 Conclusion and future work

A rainfall-runoff model has been developed to provide runoff estimates for forecasted precipitation. The model can be used to predict floods in rivers that present a danger to human lives, buildings or infrastructure.

To delineate the river's watershed, an algorithm for automatic delineation of watersheds has been outlined, in which flow directions are calculated using the D8 Algorithm. By storing upslope and downslope neighbors for each cell, we can delineate the watershed of any location in a digital elevation model. Based on the same flow information, we can also calculate the flow accumulation in the area, in which areas of large flow accumulation show rivers, lakes, dams etc.

A distributed rainfall-runoff model has been developed to exploit the steady increase of computation power, high quality remote satellite data, and accurate weather forecasts. It is based on a distributed version of the time-area method, which traditionally divides the watershed into large areas of approximately equal travel time. We do this on a much finer scale, using cells the size of our DEM's resolution. To estimate the velocity field, we simulate flow as creeping flow, and obtain travel times by solving the so-called time-of-flight equation. The hardest part was to obtain travel times for all cells; the conversion from eight possible flow directions to four, and the trap cell optimization, resulted in entire regions without valid travel times. After careful examination, we observed several special cases we had to deal with. Our implemented framework also allows for heterogeneous properties to be included in the velocity field.

Usually all rainfall-runoff models must be calibrated before reliable runoff estimates can be provided. This is outside the scope in this thesis, and because the heterogeneous properties are a part of this calibration, our velocity field is only based on topography. We want our model to be capable of predicting runoff for gauged and ungauged watersheds, alike, given a calibration based on historical input-output data from a gauged watershed. Because the model has the potential to account for all heterogeneous properties (which represent physical parameters), it only needs to be calibrated once.

To test our rainfall-runoff model we use synthetic rainfalls in our watershed. The hydrograph response is then studied, and we look at both stationary and dynamic rainfalls. We compare durations, intensities, directions, shapes and speeds. To decrease running time for hydrograph calculations, we only compute the precipitation's affected cells every Δt seconds. To optimize, we choose the largest Δt, which qualitatively gives the same hydrograph based on some requirements. This optimization works well, but the faster the storm moves, the lower we must set Δt. One example is a 10 km wide storm front that moves at a speed of 1 m/s, for which we can use a Δt of 480 seconds. If we increase the speed to 5.6 m/s, our chosen Δt decreases to 240 seconds.

A simplification we have done, is to let all precipitation become runoff, which means the hydrograph shows all newly fallen precipitation. The rainfall-runoff model works well, given its limitations; it enables us to estimate the discharge for a river, given a forecasted rainfall, which is a major part of flood predictions.

Future work

Now that we have developed a framework for a distributed rainfall-runoff model, there are many improvements, possible expansions and opportunities. We will mention some of the ideas we have for the model.

The equation we used to create our velocity field, Equation (4.8), made it possible to include heterogeneous properties to adjust the velocity field based on local variations in the landscape. Examples of this include type of soil, vegetation, snow cover etc. If these effects are included, the permeability κ is changed. New flow directions can be calculated on the basis of this, so that both the velocity field's magnitudes and directions are changed. The permeability κ can be continuously expanded to account for more and more effects, which will improve the travel time estimates.

Because we consider our flow as creeping flow, our approximation of speed in rivers is not a good model, as the speed tend to be higher in these regions. Instead, a combination of shallow water equations for rivers (e.g., cells where flow accumulation is large), and porous media flow elsewhere, is an alternative which could have offered more accurate travel time estimates.

A simple model where we allow storage of water in the terrain can be implemented. This means varying elevations of lakes, dams and ponds. An improved model for inflow and outflow of these water accumulations would also be needed.

Something we did not focus much on in this thesis, was the comparison of our flow accumulation plots to river maps. It is suspected that the artifacts of the D8 Algorithm gives some mismatching, but overall a good fit. A comparison of these could be interesting to look more into.

A Data set and tools

A.1 The data set

To automatically delineate watersheds, we have used high resolution elevation data in a raster format. The topographic surface of the terrain is represented by a two-dimensional grid, where each grid point is assigned the measured elevation of the topography. In this work the resolution of the grid is 10 x 10 meters, and the accuracy of the elevataion data is ± 5 m[1].

The Norwegian Mapping Authority (NMA) has made DEM data for all of Norway publically available. The landscape has been divided into zones that cover 50 x 50 km, and each of the zones have some overlap to the adjacent zones. Figure A.1 shows the zones for the southern part of Norway. We have used a 40 x 40 km section of the zone with the blue color[2]. The area surrounds the large lake Tyrifjorden, which is located to the northwest of Oslo.

The chosen grid resolution of 10 x 10 meters should be sufficient to obtain an accurate delineation of the watersheds in the area. A resolution of 1 x 1 meters is available, but the gain in accuracy would probably not make the substantial additional computation cost worth it, as it would increase the size of the grid from 4000 x 4000 to 40 000 x 40 000.

A.2 Languages and packages used

Our work can be divided into two parts. In the first part we make the landscape depressionless, and calculate the watersheds and the flow accumulation for the landscape. In the second part we use the watershed we obtain to estimate the travel times from each cell in the watershed to the outlet. We use this to create hydrographs for different precipitation scenarios.

In the first part we primarily rely on Python[3], which is a high-level language that is well suited for scientific programming. Because there are many packages available, and since it is open-source, it is a good choice for the implementation of the algorithm. Numpy[4] has been used in the manipulations and calculations performed using one- or multidimensional arrays. When the matrices were sparse, the Scipy[5] library was useful. In the problems that could be solved using graph

[1]In the metadata for our dataset, NMA states that the accuracy for the elevations will vary for different areas, depending on the availability of data. In some areas the accuracy is $\pm 2-3$ m, while other areas have an accuracy of $\pm 4-6$ m.

[2]The DEM from NMA is called 'Digital terrengmodell 10 m, UTM 33'.

[3]We used Python 2.7 [22].

[4]Numpy can be downloaded from http://www.numpy.org/.

[5]Scipy is a collection of open-source software that can be found at https://www.scipy.org/.

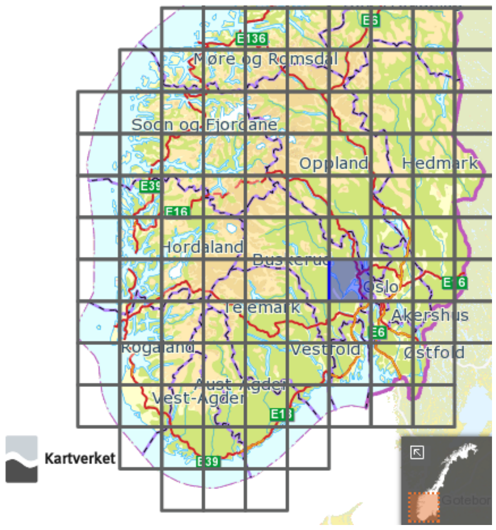


Figure A.1: The different sections in the DEM data of Norway. The Tyrifjorden landscape is a section of the blue area.

algorithms, NetworkX[6] worked well given that the number of vertices was not too large[7]. To test the implemented methods, the tool Pytest[8] was used. The implemented algorithm turned out to be very suitable for unit testing, which made it a lot easier to rewrite and improve code.

To calculate the hydrographs and the travel time, we use Matlab R2016a[9]. Unfortunately Matlab is not open-source as the other software we have mentioned, but is very popular among scientists and engineers. Matlab is paired with the open-source Matlab Reservoir Simulation Toolbox (MRST) [21] used primarily for oil reservoir simulations. It should be noted that at least parts of MRST is also available in Octave, but we have not tested this ourselves.

[6]The graph package is available at https://networkx.github.io/.

[7]It worked well for around 100 000 cells, but did not work if all 16 000 000 cells were used as vertices.

[8]The pytest testing tool can be found at http://doc.pytest.org/.

[9]The documentation is found at: https://www.mathworks.com/help/matlab/

B Drawbacks with the D8 Algorithm

There are a plethora of choices for flow determination algorithms, and they all have some disadvantages. Here we will outline some of the disadvantages with the one that we have implemented — the D8 Algorithm.

Because of the limited number of flow directions from a cell in the D8 Algorithm, the pathways in the drainage network tend to flow in parallel lines along directions that are multiples of 45° [10]. In Figure 6.3 the plots visualize this effect quite well, especially the right figure. We will now show a worst-case scenario of this effect in Example 30, where water flows on an angled plane.

Example 30. In this example we will show how the performance of the D8 Algorithm is dependent on grid orienation. We will compare two tilted planes represented by grids with different orientations. The actual flow direction is 22.5° east of the southern direction, which is indicated by blue arrows in in Figure B.1.

The two grid orientations we will compare are one where the grid is aligned with the direction of steepest descent (right figure), and one where the grid is shifted 22.5° away from the direction of steepest descent. In both cases, the chosen flow direction should ideally be parallel to the actual flow direction of the plane for every point on the grid. We will see that a combination of a low number of acceptable flow directions and bad luck with the grid orientation[1] can produce a worst-case scenario where the flow direction is consistently off by 22.5°.

We will start In both cases we will look at the flow from location a. If we examine the left figure first, we see that the direction of steepest descent will be none of the eight directions. In fact, the slope is the same to both b and g. Because the method is deterministic, one of the two alternatives are chosen in all similar cases. If we assume the southern direction is chosen, the water flows to b. From b, the same choice has to be made, which results in flow to c. After the flow has visited both d, it stops in e. The error is quite significant after only four steps. If the grid is aligned with the direction of steepest descent, as shown in the right figure in Figure B.1, there will be no error, and it makes the correct choice in every step.

Parallel lines are a problem for most algorithms with a discrete set of directions, but the effect is less pronounced for rugged terrain.

B.1 Difference between D8 and D4

The time-of-flight solver in MRST uses a finite volume scheme which considers flow over the four faces of the cell. This means that diagonal flow can be represented as

[1]The difference in two computational results, which is caused by differences in grid orientation, is called *grid orientation effects*.

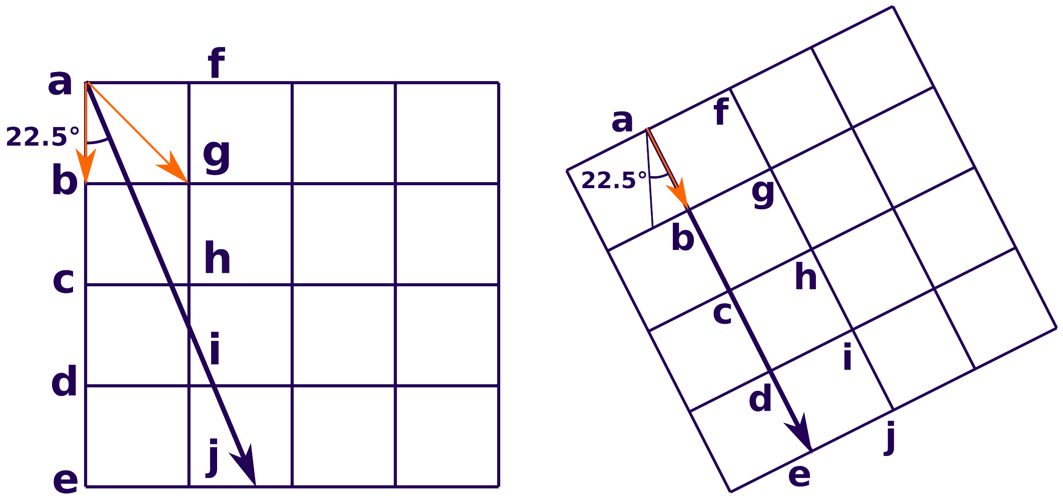


Figure B.1: A tilted plane with a steepest descent in a direction 22.5° east of south (indicated by blue arrows). The left figure shows the worst-case scenario for the D8 Algorithm. The flow starts in *a*, and because the slope is the same to both *b* and *g* (orange arrows), it is the same which one is chosen. However, because the algorithm is deterministic, the same direction will be chosen every time (south or southeast). No matter if the flow moves south or southeast, the flow direction will be off by 22.5°, which over time accumulates to a large error. In the right figure the grid orienation is changed, which yields no error with the D8 Algorithm. The example shows that if you are unlucky with the grid orientation, the error can potentially get quite big.

flow to the two faces that are adjacent to the diagonal. To remedy this, one idea is to use the D4 Algorithm, which uses only four flow directions in the calculations of the watershed. Unfortunately, this is not a good solution either, as the watersheds the D4 Algorithm yields do not look very realistic. This is because the D4 Algorithm only allows flow in the cardinal directions, which handles diagonal flow poorly. So even though the D8 Algorithm is far from perfect, the alternative is worse. This can be seen in Figure B.2, which shows the watershed of the cell colored black. The green dots show the intersection between the two algorithms, whereas the blue dots are the cells only present in the D4 Algorithm's watershed. Lastly, the red dots show the cells that are exclusive to the D8 Algorithm. We see that the D8 Algorithm delineates a more natural looking watershed.

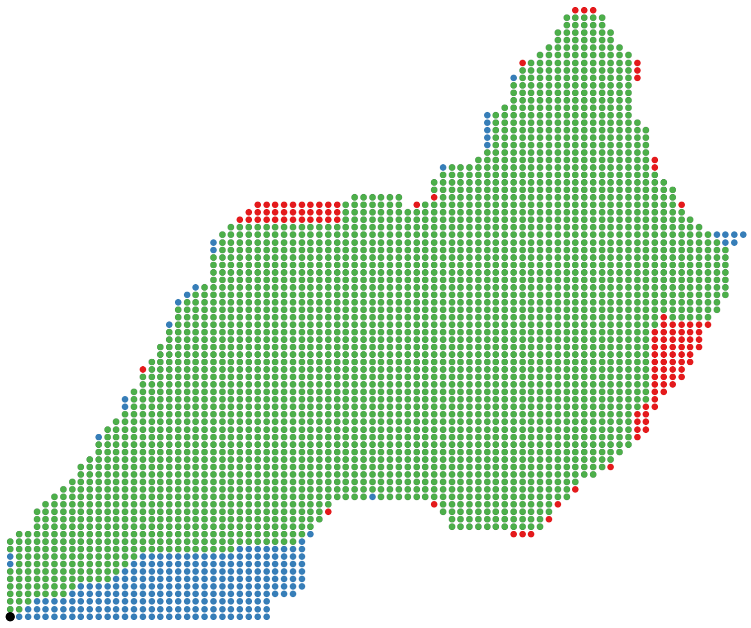


Figure B.2: The watershed of a cell (colored black) delineated by the D8 and the D4 Algorithm. The intersecting cells are colored green, while the cells that are exclusive to the D4 and the D8 Algorithm are colored blue and red, respectively.

Bibliography

[1] A. Ammann et al. *Method for the comparative evaluation of nontidal wetlands in New Hampshire*. New Hampshire Dept. of Environmental Services, Jan. 1991.

[2] K. Beven. *Rainfall-Runoff Modelling: The Primer*. Wiley-Blackwell, 2012. isbn: 047071459X.

[3] K. Beven and M. Kirkby. "A physically based, variable contributing area model of basin hydrology / Un modèle à base physique de zone d'appel variable de l'hydrologie du bassin versant". In: *Hydrological Sciences Bulletin* 24.1 (Mar. 1979), pp. 43–69. url: https://doi.org/10.1080%2F02626667909491834.

[4] R. Burnash et al. *A Generalized Streamflow Simulation System: Conceptual Modeling for Digital Computers*. U.S. Department of Commerce, National Weather Service, and State of California, Department of Water Resources, 1973.

[5] National Weather Service: National Operational Hydrologic Remote Sensing Center. *A Time-Area Instantaneous Unit Hydrograph*. Implementation Plan for the Development of Synthetic Unit Hydrographs.

[6] V. Chow, D. Maidment, and L. Mays. *Applied Hydrology*. McGraw-Hill Science/Engineering/Math, 1988. isbn: 0070108102.

[7] C. Clark. "Storage and the unit hydrograph". In: *Transactions of the American Society of Civil Engineers* 110.2261 (1945), pp. 1419–1446.

[8] N. Crawford. *The Synthesis of Continuous Streamflow Hydrographs on a Digital Computer*. Stanford University. Dept. of Civil Engineering. Technical report, no. 12. Department of Civil Engineering, Stanford University, 1962.

[9] A. El-Nasr et al. "Modelling the hydrology of a catchment using a distributed and a semi-distributed model". In: *Hydrological Processes* 19.3 (2005), pp. 573–587. url: https://doi.org/10.1002%2Fhyp.5610.

[10] J. Fairfield and P. Leymarie. "Drainage networks from grid digital elevation models". In: *Water Resources Research* 27.5 (May 1991), pp. 709–717. url: http://dx.doi.org/10.1029/90WR02658.

[11] Task Committee on GIS Modules and Distributed Models. *GIS Modules and Distributed Models of the Watershed*. American Society of Civil Engineers, 1999. isbn: 9780784474730.

[12] J. Garbrecht et al. "GIS and Distributed Watershed Models. I: Data Coverages and Sources". In: *Journal of Hydrologic Engineering* 6.6 (Dec. 2001), pp. 506–514. url: https://doi.org/10.1061%2F%28asce%291084-0699%282001%296%3A6%28506%29.

[13] The Federal Interagency Stream Restoration Working Group. *Stream Corridor Restoration: Principles, Processes, Practices*. Natl Technical Information, Oct. 1998.

[14] M. Hrachowitz et al. “A decade of Predictions in Ungauged Basins (PUB)—a review”. In: *Hydrological Sciences Journal* 58.6 (June 2013), pp. 1198–1255. url: https://doi.org/10.1080%2F02626667.2013.803183.

[15] A. Jain et al. “Rainfall runoff modelling using neural networks: State-of-the-art and future research needs”. In: *ISH Journal of Hydraulic Engineering* 15.sup1 (Jan. 2009), pp. 52–74. url: https://doi.org/10.1080%2F09715010.2009.10514968.

[16] S. Jenson and J. Domingue. “Extracting topographic structure from digital elevation data for geographic information system analysis”. In: *Photogrammetric Engineering and Remote Sensing* 54 (Nov. 1988), pp. 1593–1600.

[17] M. King and A. Datta-Gupta. “Streamline simulation: A current perspective”. In: *In Situ* 22.1 (1998), pp. 91–140.

[18] H. Kling and H. Gupta. “On the development of regionalization relationships for lumped watershed models: The impact of ignoring sub-basin scale variability”. In: *Journal of Hydrology* 373.3-4 (July 2009), pp. 337–351. url: https://doi.org/10.1016%2Fj.jhydrol.2009.04.031.

[19] N. Lea. “An aspect-driven kinematic routing algorithm”. In: *Overland Flow: Hydraulics And Erosion Mechanics*. 1992, pp. 374–388.

[20] H. Li, Y. Zhang, and X. Zhou. “Predicting Surface Runoff from Catchment to Large Region”. In: *Advances in Meteorology* 2015 (2015), pp. 1–13. url: https://doi.org/10.1155%2F2015%2F720967.

[21] K. Lie. “An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST)”. Dec. 2016.

[22] M. Lutz. *Programming Python: Powerful Object-Oriented Programming*. O’Reilly Media, 2010.

[23] S. Margulis. “Introduction To Hydrology”. Used as textbook in C&EE 150: Introduction to Hydrology" undergraduate course at UCLA. Aug. 2015.

[24] I. Muzik. “Flood modelling with GIS-derived distributed unit hydrographs”. In: *Hydrological Processes* 10.10 (1996), pp. 1401–1409. issn: 1099-1085. url: http://dx.doi.org/10.1002/(SICI)1099-1085(199610)10:10<1401::AID-HYP469>3.0.CO;2-3.

[25] S. Nelson. *River Flooding*. Note about River Flooding in course EENS 3050 at Tulane University. Oct. 2015.

[26] J. O’Callaghan and D. Mark. “The extraction of drainage networks from digital elevation data”. In: *Computer Vision, Graphics, and Image Processing* 28.3 (Dec. 1984), pp. 323–344. doi: http://dx.doi.org/10.1016/S0734-189X(84)80011-0.

[27] R. Pachauri et al. *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, 2014.

[28] K. Paik. "Global search algorithm for nondispersive flow path extraction". In: *Journal of Geophysical Research* 113.F4 (Oct. 2008). url: http://dx.doi.org/10.1029/2007JF000964.

[29] P. Pilesjö. "An Integrated Raster-TIN Surface Flow Algorithm". In: *Advances in Digital Terrain Analysis*. 2008, pp. 237–255. url: http://dx.doi.org/10.1007/978-3-540-77800-4.

[30] V. Ponce. *Engineering Hydrology, Principles and Practices*. Second edition of the book published by Prentice Hall in 1989. 2014.

[31] J. Refsgaard. "Parameterisation, calibration and validation of distributed hydrological models". In: *Journal of hydrology* 198.1 (1997), pp. 69–97.

[32] J. Refsgaard and B. Storm. "Mike she". In: *Computer models of watershed hydrology* 1 (1995), pp. 809–846.

[33] V. Rumynin. "Overland Flow Dynamics and Solute Transport". In: *Theory and Applications of Transport in Porous Media* 26 (Oct. 2015). url: http://dx.doi.org/10.1007/978-3-319-21801-4.

[34] B. Saghafian, P. Julien, and H. Rajaie. "Runoff hydrograph simulation based on time variable isochrone technique". In: *Journal of Hydrology* 261.1-4 (Apr. 2002), pp. 193–203. url: https://doi.org/10.1016/s0022-1694(02)00007-0.

[35] M. Schäfer. *Computational Engineering — Introduction to Numerical Methods*. Springer Nature, 2006, pp. 77–103. url: https://doi.org/10.1007%2F3-540-30686-2.

[36] A. Tribe. "Automated recognition of valley lines and drainage networks from grid digital elevation models: a review and a new method". In: *Journal of Hydrology* 139.1 (1992), pp. 263–293. url: http://dx.doi.org/10.1016/0022-1694(92)90206-B.

[37] J. Vaze et al. "Guidelines for rainfall-runoff modelling". In: *Australian Government Department of Innovation, Industry, science and Research* (2012).

[38] A. Voldsund. Implementation of watershed delineation, time-of-flight functionality and hydrograph calculations. 2017. url: https://github.com/avoldsund/watershed-v2.

[39] A. Voldsund. "Automatic Delineation and Analysis of Watersheds". The work in my specialization project at NTNU. Precursor to Master's thesis. 2016.

[40] B. Zhang and R. Govindaraju. "Prediction of watershed runoff using Bayesian concepts and modular neural networks". In: *Water Resources Research* 36.3 (Mar. 2000), pp. 753–762. url: https://doi.org/10.1029%2F1999wr900264.