UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ANAYLSIS OF SURFACE RUNOFF ON TOPOGRAPHICAL SIMULATIONS

IS-MARI WHEELER

A project dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING (CIVIL ENGINEERING)

In the

FACULTY OF ENGINEERING

UNIVERSITY OF PRETORIA

July 2018

DISSERTATION SUMMARY

ANAYLSIS OF SURFACE RUNOFF ON TOPOGRAPHICAL SIMULATIONS

IS-MARI WHEELER

Supervisor: Me. I Loots

Department: Civil Engineering

University: University of Pretoria

Degree: Bachelor of Engineering (Civil engineering)

During rainfall, several factors influence the runoff of the water on the terrain. Factors such as the slope of the topography, terrain coverage, i.e. vegetation or man-made, and soil infiltration can have an impact on the measured rainfall runoff. It is however uncertain as to which factor has the largest impact on runoff. This dissertation investigates each factor, and its effect on runoff. Simulation-based experiments were designed, executed and results analysed to determine the most influencing factor. It was found that soil hydraulic conductivity has the largest impact on runoff, but also that none of these factors can be assessed on their own. Each factor is intertwined with the other factors, and thus each factor's effect will change according to the involvement of the other factors.

DECLARATION

I, the undersigned hereby declare that:

- I understand what plagiarism is and I am aware of the University's policy in this regard;
- The work contained in this thesis is my own original work;
- I did not refer to work of current or previous students, lecture notes, handbooks or any other study material without proper referencing;
- Where other people's work has been used this has been properly acknowledged and referenced;
- I have not allowed anyone to copy any part of my thesis;
- I have not previously in its entirety or in part submitted this thesis at any university for a degree.

DISCLAIMER:

The work presented in this report is that of the student alone. Students were encouraged to take ownership of their projects and to develop and execute their experiments with limited guidance and assistance. The content of the research does not necessarily represent the views of the supervisor or any staff member of the University of Pretoria, Department of Civil Engineering. The supervisor did not read or edit the final report and is not responsible for any technical inaccuracies, statements or errors. The conclusions and recommendations given in the report are also not necessarily that of the supervisor, sponsors or companies involved in the research.

Signature of student: 

Name of student: **Is-Mari Wheeler**

Student number: **11027089**

Date: **17 July 2018**

Number of words in report: **11 866 words**

ACKNOWLEDGEMENTS

I wish to express my greatest appreciation to the following persons who made this project dissertation possible:

a) Me. I Loots for always being available and going above and beyond to help finding the relevant statistics, theories and guidance.

b) Ms. MS Willers for her undivided support and never ending guidance with simulation work and proof reading.

c) Mr. CJ Willers for proof reading and help with Python language background information.

d) Mr. DB Kretschmer for his continuous support and proof reading.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

ψ	wetting front soil suction head (m)
θ	water content
K	hydraulic conductivity (m/h)
$F(t)$	cumulative depth of infiltration (m)
t	time (h)

1 INTRODUCTION

1.1 PROBLEM STATEMENT

During rainfall there are several factors that influence the runoff of water over terrain topography. It is known that topographical slope, coverage (vegetation and man-made development) and soil infiltration are the main factors that influence the water-runoff on a terrain. The purpose of this study is to investigate (a) the balance of the factors' effects, and (b) to determine which one of these factors (if any) has the largest impact on the volume and rate of runoff. Answers to these questions are sought by utilising a computer simulation model of rainwater runoff in a series of well-defined experiments.

1.2 HYPOTHESIS

As can be generally assumed, the effect of topographical slope, coverage (vegetation and man-made development) and soil infiltration on rainfall runoff on a terrain are interdependent. For example, if the slope is steep, it is expected that the infiltration/permeability will have less effect on the water-runoff. If the vegetation density or urbanisation is high, the slope and/or the permeability is expected to have an effect on the runoff. If the soil infiltration is high, the terrain slope and vegetation might play a less significant role on water runoff.

It is proposed that the rainfall runoff is determined by the combination of all factors, but under certain conditions some factors may dominate to the extent that the effect of less dominant factors could be safely ignored. The purpose of this study is to determine when some factors may be considered to have no significant effect.

1.3 METHODOLOGY

To test the Hypotheses, the Landlab (Hobley, 2017) simulation tool was used to study the impact and interaction of the main factors on water-runoff. Test scenarios were defined for analysis in simulation. The range of topographical slope, surface roughness and soil infiltration depths were set to contain the data set within practical limits. Combinations and permutations of these free parameters were executed within the simulation environment. The simulation results were analysed, grouped, filtered and graphically presented to confirm the impact of these factors on runoff and to determine how they are interlinked.

The simulation was set up with the required combinations of input parameters using Python scripts and Jupyter notebooks. The Jupyter Notebook is an editable, interactive electronic lab-book running on a computer. The notebook comprises cells that can be live-running code, text, multimedia or visualisations. The primary output of the simulation, as executed here, was the hydrograph (discharge rate at a specific location as function of time), as indicative of water run-off behaviour. Landlab also provides many other outputs, but these were not used in the present study. A simple topography of a single valley between two planes with known

slopes was used. The topography boundary conditions (watershed) was defined to allow discharge only through a single discharge node at the bottom of the valley.

The large number of simulations covering all the combinations were run in batch mode, and the results stored on disk for subsequent analysis. The raw hydrograph data are difficult to interpret because of the sheer volume of data points. Therefore, analysing the data required aggregation by grouping, filtering and extracting key parameters from the hydrographs. The hydrographs were used to determine maximum discharge, cumulative discharge and time to maximum discharge. In other words, the full hydrograph was simplified to three key numbers. Data were then grouped and filtered into subsets and then eliminating common values (notably, the boundary conditions, soil infiltration depth), thereby reducing the input dimensionality to three parameters: slope, hydraulic conductivity (soil type) and surface roughness (Manning n, coverage).

The data were then presented in heatmaps representing one of the hydrograph results (e.g., maximum discharge), versus two of the input parameters (e.g,. slope and hydraulic conductivity). A heatmap is a graph where the magnitudes of values contained in a two-dimensional (2-D) matrix are represented by colours. The heatmaps greatly simplify the analysis process, but there still remain many combinations, and hence a large number of heatmaps. A number of conclusions were then made from observations of the heatmaps.

1.4 OBJECTIVES

The aim of this dissertation is to investigate, through simulation, what the impact is of topography, coverage and soil infiltration on rainfall-runoff, and how each one of these factors influence the water-runoff on the terrain. The simulations will also show the interplay between these components and their dependence of one another.

1.5 ORGANISATION OF THE REPORT

The report consists of the following chapters and appendices:

- Chapter 1 serves as introduction to the report.
- Chapter 2 contains a technical introduction based on a literature study.
- Chapter 3 explains the setup of the simulation done as to support the literature study.
- Chapter 4 describes the analysis of the simulated observations.
- Chapter 5 contains the conclusions and recommendations of the simulation studies.
- The list of references follows at the end of the report.
- Appendix A contains the calibration catchment procedures and scripts.

- Appendix B contains the experiment procedures and scripts.
- Appendix C contains the Evaluation Form.

The Python code, input and output data, relevant articles and other work done in preparation and execution of the dissertation can be found at the following Github Uniform Resource Locator (URL):

https://github.com/ismari92/SurfaceRunoffModel

2 LITERATURE REVIEW

2.1 INTRODUCTION

According to Masoudian and Theobald (2011) there are five elements that affect a flood, i.e. water-runoff during rainfall. Firstly, Meteorological elements, such as temperature, rainfall and evaporation affect a flood significantly. Secondly, the characteristics of the soil, for example, the soil type, hydraulic conductivity and field capacity can alter the runoff of a flood. Thirdly, terrain topographical elements, such as topographical slope, river profiles and cross sections are greatly responsible for the direction and accumulation of flow. Fourthly, the land use elements, like an agricultural area or settlement changes the flow-rate effectiveness. Lastly, the river network elements (or man-made structures like roads and storm water conveyance systems) affect runoff such as to divert the flow.

Each element has a unique way of influencing rainfall-runoff. Different factors were separately researched, to find expert conclusions towards which factor has the most prominent influence. Applicable published literature on field experiments and simulated experiments were studied to understand the benefits and complexities of each approach. Articles were consulted to obtain guidance on how to conduct simulation experiments.

Simulation, as experimental method, is preferred above the traditional build-and-break experimentation method (CSIR, 2014). A hardware-based experimental development cycle is often labour, time scale and material intensive. To test a hypothesis, hardware is built and tested various times before technological maturity is attained. Figure 2.1 demonstrates how modelling and simulation provides the opportunity to experiment at lower cost, risk and shorter timescales. Once the design is well tested in simulation, the physical model can be built. The modelling and simulation approach provides insight and understanding at an earlier stage of development. Technology matures earlier which leaves the opportunity for refinement and improvement.

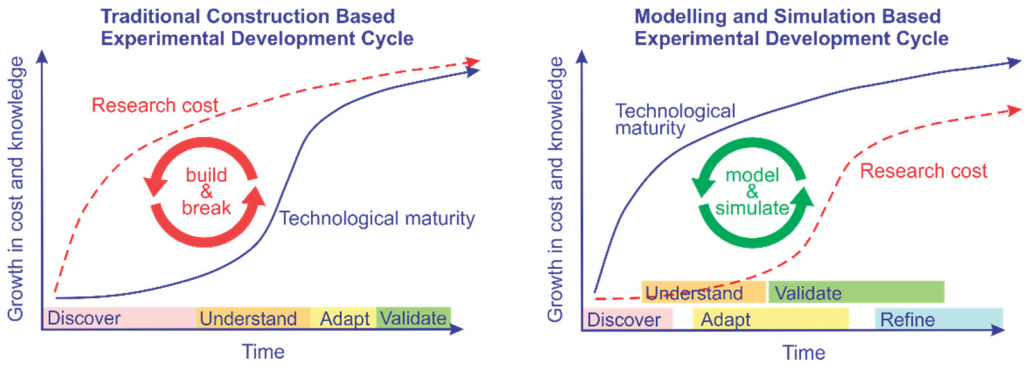


Figure 2.1: Traditional versus simulation based development cycles (CSIR, 2014)

2.2 THE IMPACT OF TOPOGRAPHICAL SLOPE ON RUNOFF

Terrain topographical slope has a large impact on rainfall-runoff. In a study by Zhang et al. (2018) an experimental field plot observation method was used to investigate the effects of the terrain slope gradient and length on runoff and soil loss. Physical rectangular plots, of different lengths, were built in a mountainous area to monitor the effects of different slopes and soils on runoff. This study showed that water-runoff is not linearly related to the slope gradient. Surface roughness (rock outcrops), the continuity or discontinuity of overland flow, soil infiltration capacity and the sealing layer of topsoil are some of the complicating factors.

It was also shown that differences in slope length affect the generation and spatial distribution of surface runoff. Zhang et al. (2018) highlights the well-known phenomenon called the scale effect, where runoff generation decreases with increasing area. An interesting observation is that when rainfall duration is equal to or longer than the concentration time, overland flow continuity occurs. Rainfall-runoff showed a strong relationship with rainfall amounts, and intensities. As the intensity increased, the runoff increased, which directly influenced the relationship with the effect of the slope.

A hydrograph is a useful and efficient visual aid when working with water-runoff. A hydrograph is a graph that represents the flow (as volume rate) as a function of time, at a specific chosen location. By choosing different locations on a terrain, the spatial effect of any influencing factor can easily be visualised.

Masoudian and Theobald (2011) did a study where a base catchment was simulated, and only the topographical slope of the catchment was changed 24 times, using GIS ArcView software. Using a NASIM rainfall-runoff model, hydrographs were created from these simulated catchments.

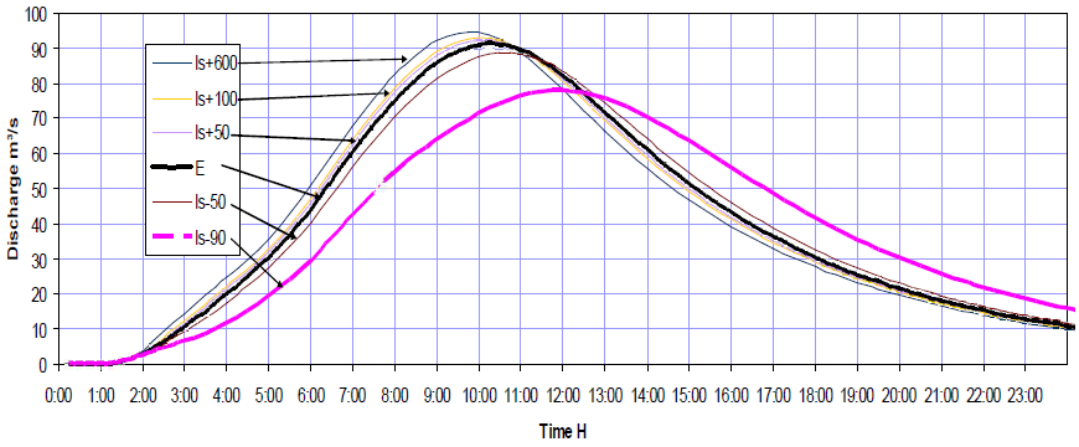


Figure 2.2: Hydrographs for different topographic slope simulations (Masoudian and Theobald, 2011)

Figure 2.2, from Masoudian and Theobald (2011), shows that the catchments with steeper slopes' hydrographs moved upwards (higher flow rate) and to the left (quicker in time), as an indication that the flow rate increased and also reached its peak at an earlier stage. The opposite can be seen for flatter slopes, the flow rate has decreased, and only reached its peak at a later stage.

2.3 THE IMPACT OF COVERAGE ON RUNOFF

2.3.1 Vegetation as coverage

Vegetation cover, considered as a 'surface roughness', as represented by the Manning n value, has an effect on the speed of runoff of rainfall. The Manning's n is a coefficient which represents the roughness or friction applied to the flow by the channel bed. Vegetation presence on a terrain is an effective way to reduce runoff. Zhang et al. (2015) has done a field experiment on various experimental plots, varying from flat platforms to steep slopes, in an opencast mine in Shanxi. He investigated the effect of vegetation on runoff. The volumes of runoff were recorded during each rainfall. According to the results the flat platforms generated larger runoff, while the steep slopes generated larger erosion (Zhang et al., 2015).

Different types of vegetation have different impacts on runoff, Zhang et al. (2015) concluded that plots covered with 5-year old mixed legume shrubs and mixed grass-shrub forests were most effective with regards to decreasing runoff.

Studies on historical data from four Texas metropolitan areas also showed that healthier green spaces decrease the amount of runoff. The study was conducted using records from October 2007 and October 2012. Both these years' assessments showed that more vegetation resulted in less runoff (Kim et al., 2017).

Vegetable raingardens can also be very effective as storm water-runoff reduction systems. A control garden was studied, where the one half of the garden was lined at the bottom, and the other half was unlined. Results showed that the unlined (infiltration-type) raingarden reduced both the volume and frequency of runoff by more than 90%. Thus, it is possible to produce an adequate amount of vegetables in a raingarden and reduce urban runoff, simultaneously (Richards et al., 2015).

Figure 2.3 demonstrates the flow in the unlined (infiltration-type) raingarden. The light arrows represent the upward seepage of the storm water introduced into the raingarden, flowing via the slotted pipe underneath the soil, through capillary rise. The dark arrows represent the infiltration of the storm water into the underlying soil (Richards et al., 2015).

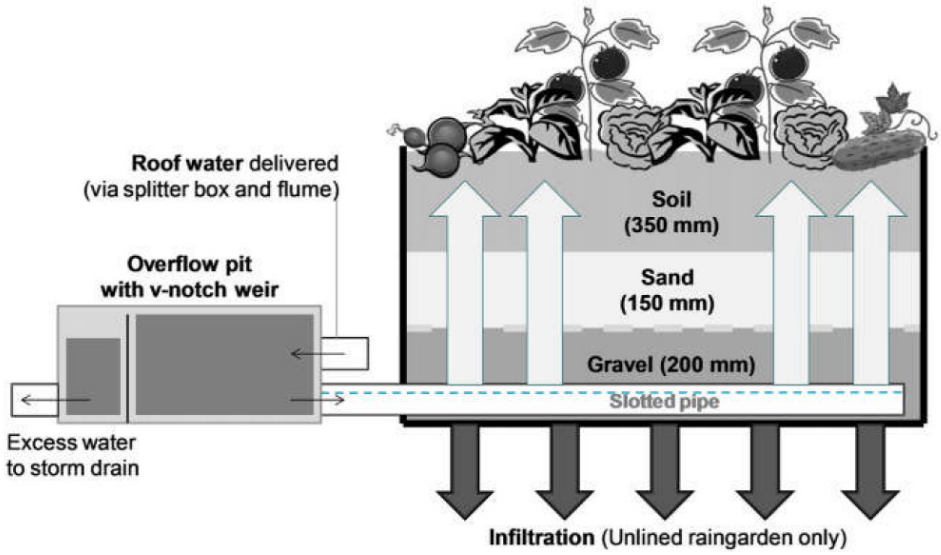


Figure 2.3: Visual representation of the unlined vegetable raingarden (Richards et al., 2015)

2.3.2 Development as coverage

It has been widely argued that urban and suburban development patterns have an influence on environmental, social and economic conditions. Aspects such as building density, neighbouring connectivity and development or land use proximity affect environmental conditions (Brody et al., 2013) .

Miller et al. (2014) investigated the changes in storm water-runoff resulting from the transformation of previously rural landscapes into peri-urban areas (the landscape interface between town and country). Two adjacent catchments were monitored for rainfall, runoff and evaporation. One catchment was highly urbanized and the other recently developed as a peri-urban area. The peri-urban area comprised of two distinct areas of drainage: one with mixed natural and storm drainage pathways, the other entirely storm drainage (Miller et al., 2014).

The hydrographs in Figure 2.4 from the storm runoff study show that the area serviced by storm drainage had a stronger determinant of storm runoff response than either impervious area (not allowing water to pass through) or development type. It is evident that as urban development increased, the flow rate increased, and the time of the peak flow shifted earlier. Thus, little distinction in hydrological response exists between urban and peri-urban developments of similar impervious cover when no significant hydraulic alteration is present. Results from the peri-urban catchment showed an increase in impervious cover from 11% to 44% in 40 years. The introduction of a large-scale storm drainage reduced the flood duration by over 50% while increasing peak flow by over 400%. (Miller et al., 2014)

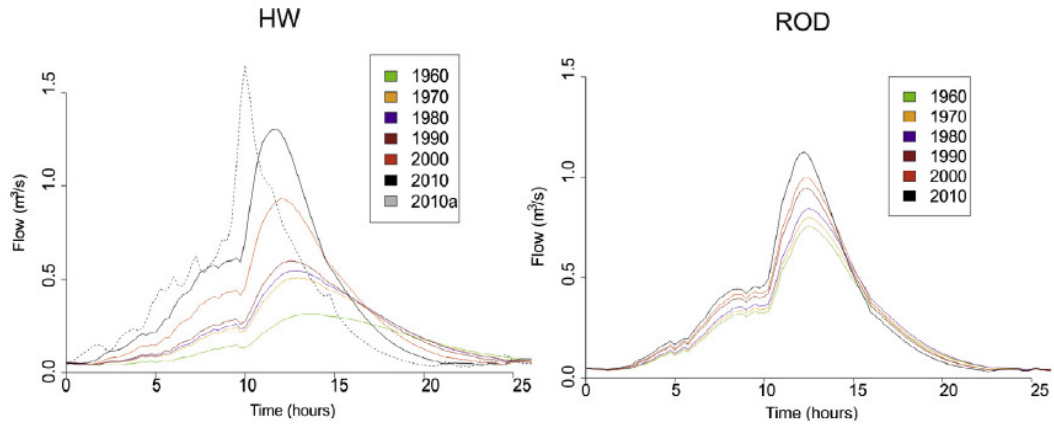


Figure 2.4: Annual averaged hydrographs from both urban (left) and peri-urban (right) developments for several years (Miller et al., 2014)

In comparing the urban and peri-urban catchments, it showed that the increase in peak flows and reduction in flood duration and response time of a catchment is greatest at low levels of urbanization. Thus, *the introduction of storm water conveyance systems significantly increases the damage of storm water-runoff.* This study demonstrates that careful consideration is required when using impervious cover data within hydrological models (Miller et al., 2014).

Wang et al. (2014) showed in a study that there is a relationship between land use changes and corresponding hydrological responses. This study was done using the Soil and Water Assessment Tool (SWAT) to simulate the runoff generation process in two separate river basins. The results showed that over a 10-year period (2000 – 2010) the total area of forest and pasture decreased, while paddy fields and upland increased, in both basins (Wang et al., 2014).

In this study it was found that land use change dramatically affected the hydrological processes. Evapotranspiration decreased, while quickflow, infiltration and flow increased

(Wang et al., 2014). The empirical regression model confirmed that upland had a negative effect on runoff while in contrast, the forest runoff generation was positive.

2.4 THE IMPACT OF INFILTRATION ON RUNOFF

Soil infiltration is a complex field element, and is dependent on various aspects, such as the type of soil, rainfall properties and initial conditions. Various mathematicians and engineers have developed their own theories and understandings regarding the science of infiltration (Assouline, 2013).

Systems, such as infiltration pond systems, have been widely researched as cost-effective local 'source control' drainage solutions. These systems reduce peak flow inside storm water systems, and are useful because of the use of infiltration to reduce the water flow rate. As there are still many technical constraints connected to sustainable drainage systems, various experiments are still being done regarding the efficiency of infiltration to reduce flow (Scholz, 2006).

Studies have been done on planted and unplanted infiltration ponds, and the rainfall and runoff monitored daily. In a study by Scholz (2006) the infiltration rate of the runoff into the ponds as well as water depth variation were monitored. The mean infiltration rate applied to the ponds was 1.17 m/h. This was unrealistically high and was found to be closer to 0.21 m/h during testing, which might have been due to the base of the pond (Scholz, 2006).

Figure 2.5 shows the experimnetal water-depth variations during periods of no precipitation for all seasons. It is visible that infiltration is relatively low at water depths smaller than 0.8m. As can be seen in the graph, infiltration is an element that requires time to be efficient. In a flat pond, it took about 25 hours for a meter to infiltrate into the pond in the Winter time.

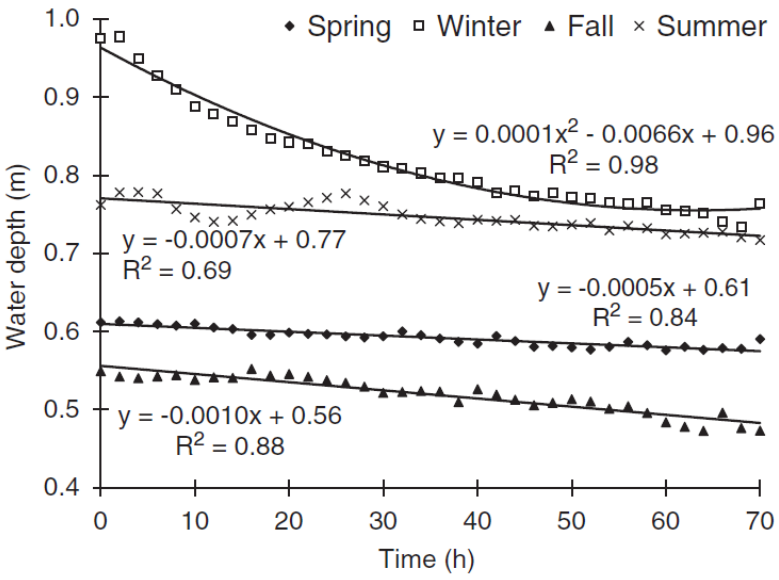


Figure 2.5: Water depth changes as a function of infiltration rates (Scholz, 2006)

2.5 LANDLAB AS SIMULATION PROGRAM

Researchers across a wide variety of disciplines use numerical models to study processes that operate on and across the Earth's surface. Many scientists often build their own Earth model, re-coding the basic building blocks of such a model. The Landlab simulation model aims to create a user- and developer-friendly modelling environment that provides the basic building blocks required for modelling the Earth surface dynamics (Hobley, 2017). This modelling framework is an open-source, highly flexible and interdisciplinary Python-language library. The development environment is flexible, extensible, highly reusable and well documented.

The Landlab framework provides a more accessible modelling environment than other complex runoff models. A large number of 'components' are available to model various aspects of Earth surface dynamics over a range of temporal and spatial scales (Adams et al., 2017).

The Landlab modelling framework takes advantage of the fact that nearly all surface-dynamics component models share a set of common software elements. It has a triplex structure comprising the core grid, a library of process components and a set of supporting utilities, as depicted in Figure 2.6 (Hobley, 2017).

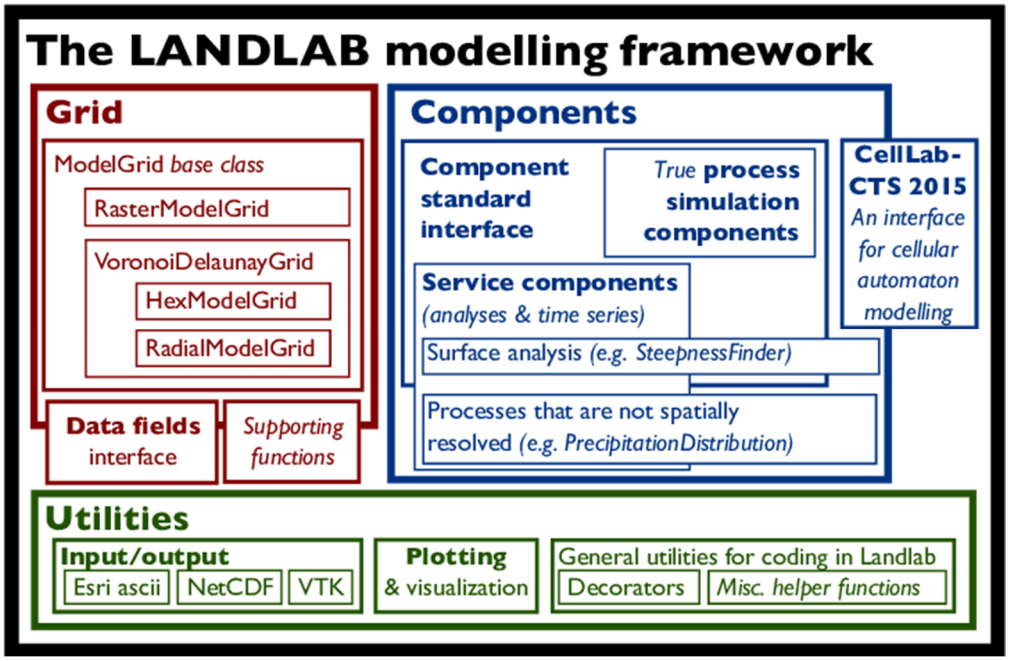


Figure 2.6: The Landlab modelling framework (Hobley, 2017)

2.5.1 The grid

The terrain topography is represented in Landlab by the *ModelGrid* class. It enables the creation of a 2-D grid with size and shape defined by the user. As depicted in Figure 2.7, each grid is defined by a set of elements: nodes, links, cells, corners, faces and patches. These elements define the connectivity between the two dimensional data fields of the grid (Hobley, 2017).

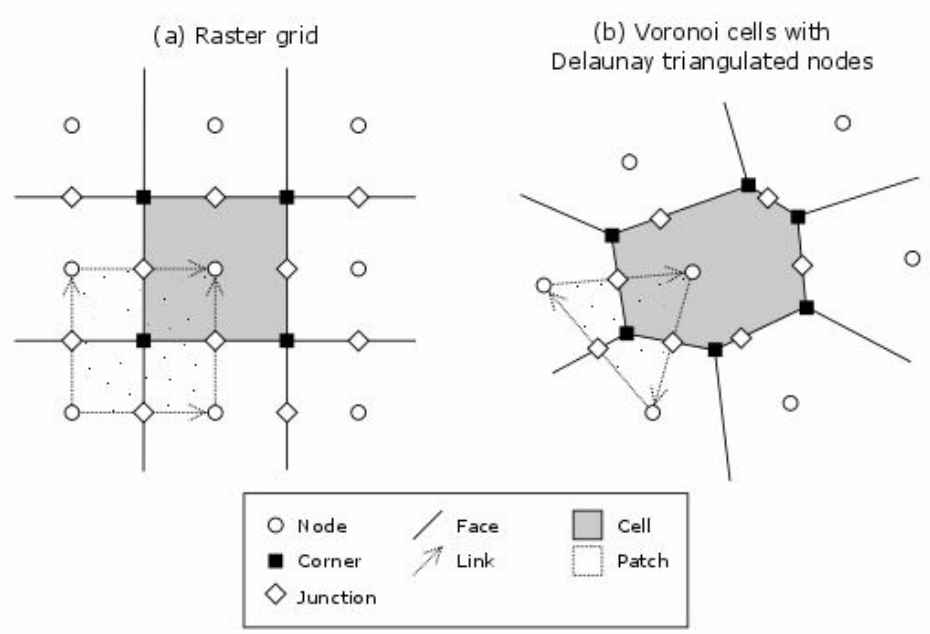


Figure 2.7: Representation of the grid layout used in Landlab (Hobley, 2017)

The grid representation can be summarised as follows. The grid contains a set of (x,y) points called *nodes*. Nodes are the locations at which one tracks scalar state variables, such as water depth, land elevation, sea-surface elevation, or temperature. Each adjacent pair of nodes is connected by a line segment known as a *link*. Every node in the grid interior is associated with a polygon known as a *cell*. Each cell is bounded by a set of line segments known as *faces*, which it shares with its neighbouring cells. In the simple case of a regular (raster) grid, the cells are square, the nodes are the centre points of the cells, and the links and faces have identical length (equal to the node spacing) (Hobley, 2017).

2.5.2 The grid boundary conditions

The *ModelGrid* class handles boundary conditions by tagging nodes that are treated as boundaries (boundary nodes) and those that are treated as regular nodes belonging to the interior computational domain (core nodes). The model also allows the de-activation of portions of the grid perimeter, so that they effectively act as walls (Hobley, 2017).

A closed boundary is one at which no flux is permitted to enter or leave. By definition, all links coming into or out of a closed boundary node must be inactive. There is effectively no value assigned to a closed boundary. An open boundary is one at which flux can enter or leave, but whose value is controlled by some boundary condition rule, updated at the end of each time step. Figure 2.8 is an illustration of a small grid with a combination of open and closed boundary nodes (Hobley, 2017).

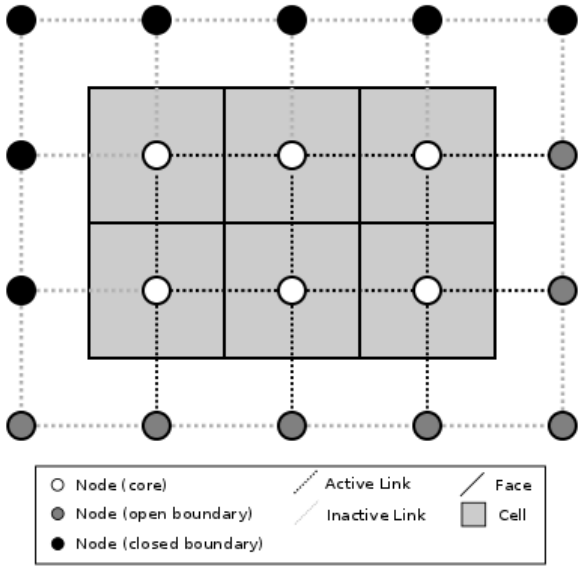


Figure 2.8: Boundary conditions representation (The Landlab Team 2013)

2.5.3 Components

Landlab offers an ever-growing library of process simulators. Components aim to describe individual or closely associated suites of surface processes. The components are designed to be plug-and-play and to interact with each other with the minimum of technical difficulties. For this investigation, two components were identified that describe and incorporate rainfall events over topographies with different aspects (margauxmouchene 2018).

Overland flow component

In the Landlab *OverlandFlow* component a 2-D solution of the shallow water equations derived from the algorithm of Almeida et al. (2012) is implemented. The component simulates a flood wave moving across a gridded terrain, capturing hydrodynamics throughout the system. At each point within the grid, surface water discharge is calculated based on physical properties. This component also has a non-steady flow routing method as an alternative to the steady-state flow routing regimes found in many geomorphic or landscape evolution models (such as the Landlab *FlowRouter* component) (Adams 2017).

User-defined input to this model includes a weight on the adaptive time step (*alpha*, ranging from 0.2 – 0.7), an empirical value describing surface roughness (*Manning n*) and a weighting factor in the Almeida equations (*theta*, ranging from 0.8 – 0.9) (Hobley, 2017).

The output data available for visualisation and processing are surface *water discharge* (m^2/s) at each link in the grid, and *surface water depth* (m) at each node (Adams 2017).

Soil infiltration component

The Landlab *SoilInfiltrationGreenAmpt* component is based on the Green-Ampt model, for surface water infiltration into soil (refer to Figure 2.9). Green and Ampt based their model on fundamental physics, which matches empirical observations (Spring, 2011).

The Green-Ampt function is a function comprising of the soil suction head, porosity, hydraulic conductivity and time, as shown in Equation 2.1:

$$F(t) - |\psi|(\Delta\theta) ln\left(\left|1 + \frac{F(t)}{|\psi|(\Delta\theta)}\right|\right) = Kt \qquad \text{(Equation 2.1)}$$

where, ψ is the wetting front soil suction head (m); θ is the water content; K is the hydraulic conductivity (m/h); $F(t)$ the cumulative depth of infiltration (m), and t is time (h).

This formula can be used to solve for either volume of infiltration, or instantaneous infiltration rate. (CE 551, 2016)

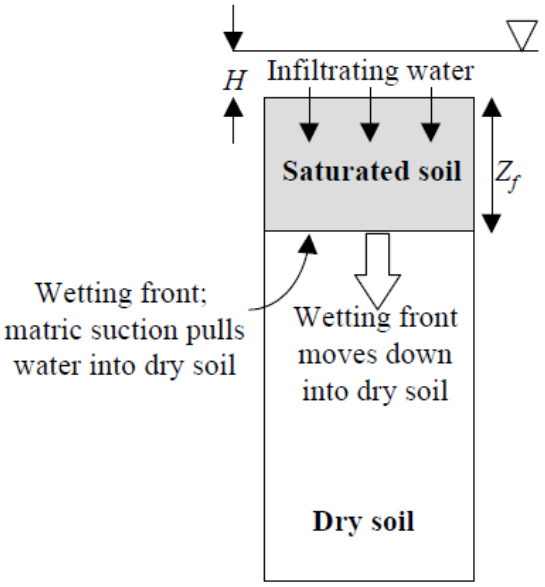


Figure 2.9: GreenAmpt infiltration model (Spring, 2011)

The Landlab *SoilInfiltrationGreenAmpt* component contains several input parameters. The soil's effective *hydraulic conductivity* (m/s), dry *soil bulk density* (kg/m^3), the density of the soil constituent material, also called r*ock density* (kg/m^3) and the *initial soil moisture content* (m^3/m^3) are input parameters to the model. The *soil type* must also be passed as a parameter, i.e. one of 'sand', loamy sand', 'sandy loam', 'loam', 'silt loam', 'sandy clay loam', 'clay loam', 'silty clay loam', 'sandy clay', 'silty clay' or 'clay'. Other parameters available on the input variable list are the *volume fraction coarse fragments* (m^3/m^3), the *surface water minimum depth* (m), the *soil pore size distribution index*, the *soil bubbling pressure* (m) and the *wetting front capillary pressure head* (m) (Hobley, 2017).

The output variables provided by this component are the *surface water depth* (m) and the *surface water infiltration depth* (m) across the grid.

2.5.4 User interface

Landlab offers a straight-forward and standardised input and output interface, including the ability to import from and export to common spatially distributed data formats. Hydrographs and 2-D output across the topographical grid can be visualised using the standard Python *Matplotlib*-module. Figure 2.10 and Figure 2.11 show examples of typical output data visualisation. The image represented in Figure 2.10 shows terrain topography in a top view, with the elevation as colour variation on the graph (see the colour bar). Figure 2.11 shows a hydrograph, the grid topography, as well as the water depth on the topography at different time steps (Hobley, 2017).

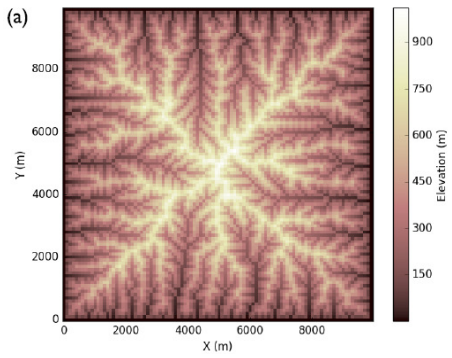


Figure 2.10: Terrain topography representation (Hobley, 2017)

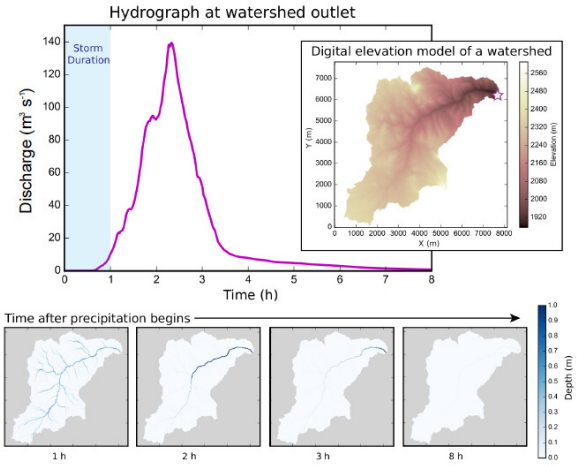


Figure 2.11: Hydrograph, topography and water depth on the topography (Hobley, 2017)

2.5.5 Summary

With all the information gathered in Chapter 2, the impact of each element and its experimental application is clear. Each element could have been experimented on its own, but it was also realised that the elements were co-dependant. One element can't be changed without affecting the other. Thus, an integrated simulated model was generated, during which the components influence each other and will return adapted variables. In this study, the *OverlandFlow* and *SoilInfiltrationGreenAmpt* components were used to simulate rainfall-runoff, and its elements. By combining these components one could simulate the effect of slope, coverage and soil infiltration on the measured rainfall runoff.

3 EXPERIMENTAL DESIGN

3.1 INTRODUCTION

The Landlab *OverlandFlow* and *SoilInfiltrationGreenAmpt* component simulation environment was calibrated against real world data, by comparing simulation rainfall flow rates against measured data. The calibration established confidence in the Landlab simulation environment and it was accepted as a valid environment for further experimentation.

This chapter describes the assumptions made for the simulations and the setup procedure, as well as the calibration procedure and results. The chapter is concluded with a discussion on the design of the simulation experiments.

At the beginning of the project considerable time was spent to understand the Landlab software; it is quite an involved simulation tool. There is documentation available, but at times, no relevant documentation was found to answer some questions. Many experiments leading to dead-ends helped to develop an understanding of how *not* to use the Landlab software. Some parameters are sensitive to initial conditions, e.g., soil infiltration depth should not be zero, it leads to non-sense answers. Landlab is a powerful simulation tool, but requires very careful setup with valid input parameters.

3.2 ASSUMPTIONS AND LIMITATIONS

The following assumptions were made in simulating the overland flow and soil infiltration: The effect of soil erosion, and storm water drainage were not taken into consideration. The simulated catchment area is within closed boundaries, thus no other water than the rainfall flows into the area. For each experiment the boundaries determine the flow out of the catchment area and flow is measured at a specified outlet point. Rainfall is simulated as an average value on the catchment area for the specified duration. The rainfall is the same and uniform over the entire catchment area.

It was determined that the model performs best under certain conditions, thus the following limitations were identified while conducting the experiment. The range of coverage (Manning n values) was limited to 0.03 to 0.07, slope was limited to 2˚ to 25˚, and hydraulic conductivity was limited to 1.6×10^{-5} m/s to 1.7×10^{-7} m/s.

3.3 SIMULATION PROCEDURE

The Landlab simulation toolset introduced in Section 2.5 was used in all experimentation. It is an already developed foundation, comprising of a terrain grid, components and ready to use utilities and user interface. A Jupyter Notebook experimentation environment was set up to use as a virtual laboratory. The workflow in all experimentation was the same: only terrain

topography and a set of input parameters were varied. The basic simulation setup and workflow followed in the Notebooks are discussed in this section. Screen dumps are used form the WatershedTutorial.ipynb Notebook at URL

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/landlabExplained

3.3.1 Import Python modules

Specific Python modules needed to be loaded at the start of the simulation to be able to perform specific tasks. The following code loads Python system modules, the fundamental package for scientific computing (numpy), the Matplotlib 2-D plotting library and Landlab specific modules to set up the raster model grid, set up and run the required components and assist in visualisation.

```
# system python modules
import sys, os.path

# numpy modules
import numpy as np

# landlab modules
from landlab import RasterModelGrid, CLOSED_BOUNDARY, FIXED_VALUE_BOUNDARY
from landlab.io import read_esri_ascii
from landlab import imshow_grid
from landlab.components import SoilInfiltrationGreenAmpt
from landlab.components import OverlandFlow

# Plotting modules
from matplotlib import pyplot as plt
import matplotlib as mpl

# path with topography dem data
topopath=os.path.join("../../","create-topo")

%matplotlib inline
```

3.3.2 Define storm conditions

For the land overflow component, the rainfall intensity (m/s) must be specified for each moment in time (s). At the start of the simulation the precipitation is defined in m/s, as well as the storm duration in s. This data is used during run time to simulate the storm intensity and duration.

```
# rainfall in mm/hour
starting_precip_mmhr = 20.6

# convert to m/s
starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)

# duration of the storm in s
storm_duration = 0.25 * 3600.
```

3.3.3 Set up the run time and logging

The simulation time is defined as a multiple of the storm duration time in order to observe the storm, as well as the dissipation of the runoff water at the outlet node, for a time period after the storm.

```
# Simulation run time and report time as function of storm duration
runHours = 2.5 * storm_duration/3600.
model_run_time = runHours * 3600.
reportInterval = model_run_time / 3.

# graphs and data during run time?
showProgress = True
showPlots = True
```

3.3.4 Grid topography definition

Simple raster grids can be manually created with Landlab using the *RasterModelGrid* function, as shown here:

```
# Creating grid of 80 by 80 with 5 as spacing
mg = RasterModelGrid((80, 80), 5.)

# Assume constant elevation of 500m
z = mg.add_zeros('node', 'topographic__elevation')
z = z + 500
```

Real world topographies can be loaded into the raster model grid from an ESRI ASCII file. This type of file contains the catchment size, and elevations at grid nodes. For the initial calibration study a real-world topography was loaded from file (see Appendix A for the conversion procedure).

Well-defined Landlab grid topographies (tilted flat slopes and gully slopes) were generated from simple geometry principles for experimentation purposes. Details of this process and utilities used are given in Appendix B. Using these utility functions, a topography can easily be created specifying the grid size and tilt of the plane. Figure 3.1 shows the flat terrain topography grid of 30 rows and 30 columns with grid size of 20 m x 20 m. The grid was tilted by 10˚ along zero azimuthal angle. The elevation was specified as 1000 m at the lower-left corner. The following code was used to generate this grid:

```
x,y,grid,filename = createFlatTopo('slope',ncols=30,nrows=30,tilt=10.,
                                   azim=0.0,cellsize=20,llcorner=[0.,0.,1000.])
plotTopo(x,y,grid,filename)
```

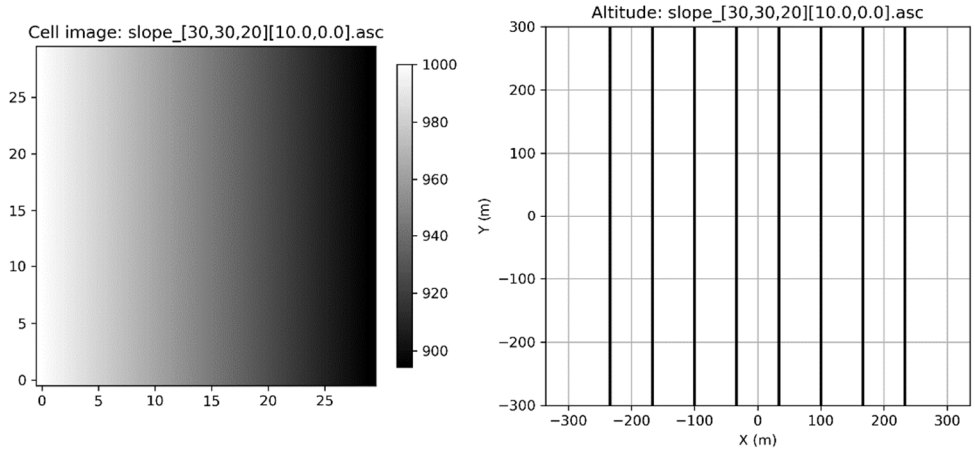


Figure 3.1: Example of simple topography generated with the utility functions provided in Appendix B

The topography of an example real-world gully is loaded from file into the simulation environment using the code:

```
(rmg, z) = read_esri_ascii(os.path.join(topopath,'west_bijou_gully_PF.asc'), name='↵
    topographic__elevation')
```

The Landlab imshow_grid function can be used to visualise the loaded topography (see Figure 3.2).

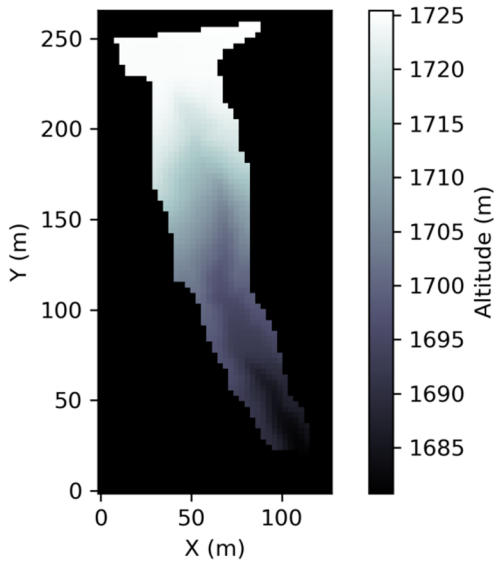


Figure 3.2: Gully topography loaded from ESRI ASCII file

3.3.5 Set starting conditions

Starting conditions, such as the surface water depth, surface water infiltration depth, terrain roughness, soil type and hydraulic conductivity must be allocated to the nodes of the grid. These initial variables are used in the relevant components, i.e. *OverlandFlow* and *SoilInfiltrationGreenAmpt*. During experimentation, the soil hydraulic conductivity was assigned a random value between a minimum and maximum to induce some variation on the grid properties.

```
# Depth of water above the surface [m]
h = rmg.add_ones('node', 'surface_water__depth')
h *= 0.02

# Water column height above the surface previously absorbed into the soil [m]
# This is NOT the actual depth of the wetted front, which also depends on the porosity
d = rmg.add_ones('node', 'soil_water_infiltration__depth', dtype=float)
d *= 0.15

# Terrain roughness
mannings_n = 0.05

# Soil type and hydraulic conductivity [m/s]
```

```
# The ease with which a fluid (usually water) can move through pore spaces or fractures
# Specify range and initialise the field with random values in this range
soil_type = 'sandy loam'
min_h = 5e-6
max_h = 5.5e-6
hydraulic_conductivity = rmg.ones('node')*min_h
hydraulic_conductivity += np.random.rand(z.size)*(max_h - min_h)
```

Figure 3.3 shows the randomised soil hydraulic conductivity values on the grid as visualised with the following code:

```
# Visualise the randomness of the hydraulic conductivity
plt.imshow(hydraulic_conductivity.reshape((rmg.shape[0],rmg.shape[1])))
plt.colorbar()
plt.title('Hydraulic conductivity [m/s] on raster model grid')
plt.axis('off')
plt.show()
```

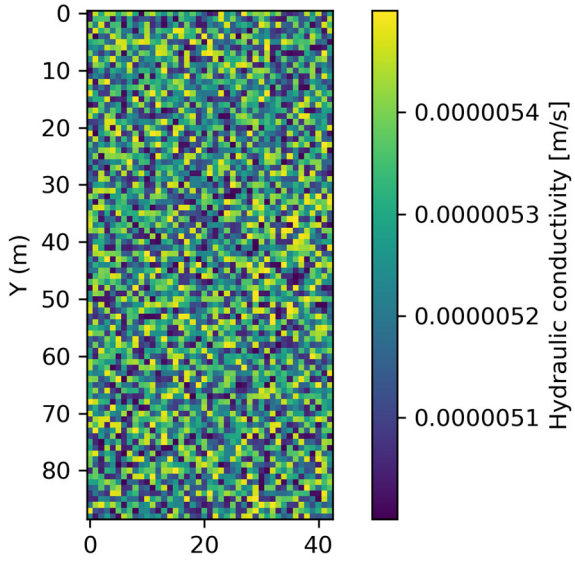


Figure 3.3: Randomised hydraulic conductivity on grid

3.3.6 Set boundary conditions and outlet node

Boundary conditions must be set to maintain control over where on the boundary the runoff will flow out of the topography. The boundaries of the catchment area can either be set to open or closed.

Setting each of the four boundaries simply as a closed or fixed boundary, the following code is used:

```
# set all boundaries closed, except right open:
for edge in (rmg.nodes_at_right_edge):
    rmg.status_at_node[edge] = FIXED_VALUE_BOUNDARY
for edge in (rmg.nodes_at_left_edge, rmg.nodes_at_top_edge, rmg.nodes_at_bottom_edge):
    rmg.status_at_node[edge] = CLOSED_BOUNDARY
```

The option is also available to set all boundaries of a catchment area to closed, except the lowest node on the boundary. This node is then also defined as the outlet node. This method was used in this example, as well as in the calibration process and in experimentation.

The Landlab imshow_grid function is used below in visualising the catchment area boundary conditions (see Figure 3.4). The blue area in the graph is the outside the catchment area and defines the closed boundaries. The outlet node, which is the point with lowest altitude in the topography grid, is shown in white.

```
# Visualise the boundary conditions and the outlet point
imshow_grid(rmg, rmg.status_at_node,color_for_closed='blue',plot_name='Boundary ↵
    conditions, showing outlet node',
            grid_units=['m','m'],var_name='Boundary value')
```

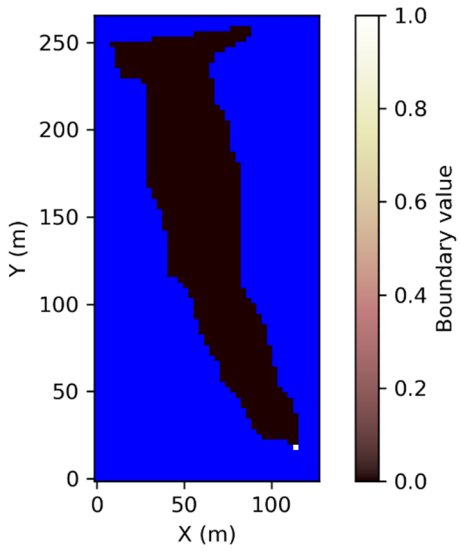


Figure 3.4: Visual representation of boundary conditions

The number of the links at the outlet node is determined and visualised (Figure 3.5) with the code:

```
# Set the outlet node value
outlet_node = np.where(rmg.status_at_node==1)
print('The outlet node is at {}'.format(outlet_node[0][0]))

# Get information about the links at the outlet node
linksAtNode = rmg.links_at_node[outlet_node][0]
print('The links at the outlet node are: {}'.format(linksAtNode))

# Status of links at the outlet node:
# 0: active
# 2: fixed
# 4: inactive
linkStatus = rmg.status_at_link[rmg.links_at_node[outlet_node]][0]
print('The status at the links are: {}'.format(linkStatus))

# Link flux directions at each node:
#  1: incoming flux
# -1: outgoing flux
#  0: no flux
fluxAtLink = rmg.active_link_dirs_at_node[outlet_node][0]
print('The flux directions at the links are: {}'.format(fluxAtLink))

# set the link to the avtive one
outlet_link = linksAtNode[np.where(linkStatus == 0)[0] and np.where(fluxAtLink != 0)↵
    [0]][0]
print('Outlet link to monitor is set to {}'.format(outlet_link))

# Visualise the outlet node
imshow_grid(rmg, z, plot_name='Outlet node on topography',
            grid_units=['m','m'],var_name='Altitude',var_units='m')
plt.plot(rmg.node_x[outlet_node],rmg.node_y[outlet_node],'go')
plt.show()
```

```
The outlet node is at 296
The links at the outlet node are: [548 590 547 505]
The status at the links are: [4 0 4 4]
The flux directions at the links are: [ 0 -1  0  0]
Outlet link to monitor is set to 590
```

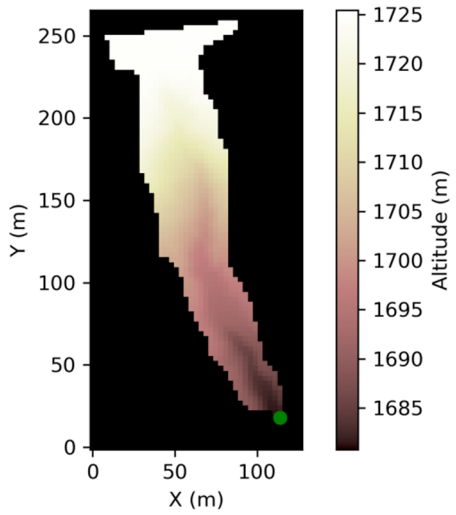


Figure 3.5: Visualisation of outlet node

3.3.7 Set up the Components

The *OverlandFlow* Landlab component simulates the runoff on topographies during a rainfall event. The *OverlandFlow* component can be set up with different values, but shown here for *Manning n=0.05* (empirical value describing surface roughness) as defined in the starting conditions.

The *SoilInfiltrationGreenAmpt* Landlab component simulates the soil conditions on which water infiltrates and run off. The component has default values which can be used as supplied, or the user can change these to different values. Only the soil type and hydraulic conductivity were specified for this experiment, typical default values were used for the rest of the variables.

```
# Set up the OverlandFlow component
of = OverlandFlow(rmg, mannings_n = mannings_n, steep_slopes = True)

# Set up the SoilInfiltrationGreenAmpt component
SI = SoilInfiltrationGreenAmpt(rmg, soil_type = soil_type, hydraulic_conductivity=↵
    hydraulic_conductivity)
```

Table 3.1 and 3.2 list typical Manning n (Chow, 1959) and typical hydraulic conductivity values (Gowdish and Muñoz-Carpena, 2009, Nie et al., 2017) consulted for this experiment.

Table 3.1: Extraction of Manning n values from Chow (1959)

Type of channel and Description	Minimum	Normal	Maximum
Main Channels			
clean, straight, full stage, no rifts or deep pools	0.025	0.03	0.033
same as above, but more stones and weeds	0.03	0.035	0.04
clean, winding, slime pools and shoals	0.033	0.04	0.045
same as above, but some weeds and stones	0.035	0.045	0.05
same as above, lower stages, more ineffective slopes and sections	0.04	0.048	0.055
same as above with more stones	0.045	0.05	0.06
sluggish reaches, weedy, deep pools	0.05	0.07	0.08
very weedy reaches, deep pools, or floodways with heavy stand of timber and underbush	0.075	0.1	0.15
Mountain streams, no vegetation in channel, banks usually steep, trees and brush along banks submerged at high stages			
bottom: gravels, cobbles, and few boulders	0.03	0.04	0.05
bottom: cobbles with large boulders	0.04	0.05	0.07

Floodplains			
Pasture, no bush			
short grass	0.025	0.03	0.035
high grass	0.03	0.035	0.05
Cultivated areas			
no crop	0.02	0.03	0.04
mature row crops	0.025	0.035	0.045
mature field crops	0.03	0.04	0.05
Brush			
scattered brush, heavy weeds	0.035	0.05	0.07
light brush and trees, in winter	0.035	0.05	0.06
light brush and trees, in summer	0.04	0.06	0.08
medium to dense brush, in winter	0.045	0.07	0.11
medium to dense brush, in summer	0.07	0.1	0.16
Trees			
dense willows, summer, straight	0.11	0.15	0.2
cleared land with tree stumps, no sprouts	0.03	0.04	0.05
same as above, but with heavy growth of sprouts	0.05	0.06	0.08
heavy stand of timber, a few down trees, little undergrowth, flood stage below branches	0.08	0.1	0.12
same as above, with flood stage reaching branches	0.1	0.12	0.16
Excavated or Dredged Channels			
Earth, straight and uniform			
clean, recently completed	0.016	0.018	0.02
clean, after weathering	0.018	0.022	0.025
gravel, uniform section, clean	0.022	0.025	0.03
with short grass, few weeds	0.022	0.027	0.033
Earth winding and sluggish			
no vegetation	0.023	0.025	0.03
grass, some weeds	0.025	0.03	0.033
dense weeds or aquatic plants in deep channels	0.03	0.035	0.04
earth bottom and rubble sides	0.028	0.03	0.035
stony bottom and weedy banks	0.025	0.035	0.04
cobble bottom and clean sides	0.03	0.04	0.05
Dragline-excavated or dredged			
no vegetation	0.025	0.028	0.033
light brush on banks	0.035	0.05	0.06
Rock cuts			
smooth and uniform	0.025	0.035	0.04
jagged and irregular	0.035	0.04	0.05

Channels not maintained, weeds and brush uncut			
dense weeds, high as flow depth	0.05	0.08	0.12
clean bottom, brush on sides	0.04	0.05	0.08
same as above, highest stage of flow	0.045	0.07	0.11
dense brush, high stage	0.08	0.1	0.14
Lined or Constructed Channels			
Cement			
neat surface	0.01	0.011	0.013
mortar	0.011	0.013	0.015
Wood			
planed, untreated	0.01	0.012	0.014
planed, creosoted	0.011	0.012	0.015
unplaned	0.011	0.013	0.015
plank with battens	0.012	0.015	0.018
lined with roofing paper	0.01	0.014	0.017
Concrete			
trowel finish	0.011	0.013	0.015
float finish	0.013	0.015	0.016
finished, with gravel on bottom	0.015	0.017	0.02
unfinished	0.014	0.017	0.02
Gunite, good section	0.016	0.019	0.023
Gunite, wavy section	0.018	0.022	0.025
on good excavated rock	0.017	0.02	
on irregular excavated rock	0.022	0.027	
Concrete bottom float finish with sides of:			
dressed stone in mortar	0.015	0.017	0.02
random stone in mortar	0.017	0.02	0.024
cement rubble masonry, plastered	0.016	0.2	0.024
cement rubble masonry	0.02	0.25	0.03
dry rubble or riprap	0.02	0.03	0.035
gravel bottom with sides of:			
formed concrete	0.017	0.02	0.025
random stone mortar	0.02	0.023	0.026
dry rubble or riprap	0.023	0.033	0.036
Brick			
glazed	0.011	0.013	0.015
in cement	0.012	0.015	0.018
Masonry			
cemented rubble	0.017	0.025	0.03
dry rubble	0.023	0.032	0.035
Dressed ashlar/stone paving	0.013	0.015	0.017

Asphalt			
smooth	0.013	0.013	
rough	0.016	0.016	
Vegetal lining	0.03		0.5

Table 3.2: Typical values for hydraulic conductivity from Gowdish and Muñoz-Carpena (2009) and Nie et al. (2017)

Soil Type	Hydraulic Conductivity (m/s)
Clay	1.7×10^{-7}
Silty Clay	2.7×10^{-7}
Sandy Clay	3.3×10^{-7}
Silty Clay Loam	5.5×10^{-7}
Clay Loam	7.2×10^{-7}
Sandy Clay Loam	8.3×10^{-7}
Silt Loam	1.8×10^{-6}
Loam	3.6×10^{-6}
Sandy Loam	6.1×10^{-6}
Loamy Sand	1.6×10^{-5}
Sand	6.5×10^{-5}

3.3.8 Run simulation and save data

Once all the required initial conditions have been set, the simulation can be executed covering the specified time duration. The storm starts at the beginning of the time period. While the elapsed time is less than the storm duration, water is being added to the system as rainfall. The *OverlandFlow* and *SoilInfiltrationGreenAmpt* models are executed for each step in time and data saved for plotting after completion of the run.

```
# Simulation loop
elapsed_time = 0.
next_report = 0.
fig = 1

# Lists for saving data
discharge_at_outlet = []
hydrograph_time = []

# Simulation Loop
while elapsed_time < model_run_time:

    # Setting the adaptive time step
    of.dt = of.calc_time_step()

    # The storm starts when the model starts
    # While the elapsed time is less than the storm duration, add water to the system as rainfall
    if elapsed_time < (storm_duration):
        of.rainfall_intensity =  starting_precip_ms

    # elapsed time exceeds the storm duration, rainfall ceases
    else:
        of.rainfall_intensity = 0.0

    # Generating overland flow component
    of.overland_flow()

    # Append time and discharge to lists to save data for plotting
    hydrograph_time.append(elapsed_time/3600.)                    # hours
    discharge_at_outlet.append(np.abs(of.q[outlet_link]) * rmg.dx) # m^3/s

    # Run soil infiltration component
    SI.run_one_step(of.dt)

    # Report if requested
    if elapsed_time >= next_report:
        if showPlots:
            plt.figure(fig)
            fig = fig + 1
            imshow_grid(rmg, 'soil_water_infiltration__depth', cmap='Blues',
                        grid_units=['m','m'],var_name='Soil water infiltration depth',
                            var_units='m')
            plt.title('Elapsed time {:.2f} hours'.format(elapsed_time/3600.))

        elif showProgress:
            print('elapsed time = {:.2f} s [{:.2f} hours]'.format(elapsed_time,
                elapsed_time/3600.))

        next_report += reportInterval

    # Update elapsed_time
    elapsed_time += of.dt
```

3.3.9 Visualise output data

The experimental outcome of the simulation was visualised by plotting the hydrograph (Figure 3.6) and cumulative water flow graph (Figure 3.7) from the data captured at the outlet node.

```
# Hydrograph at the outlet node
plt.plot(hydrograph_time, discharge_at_outlet, 'g')
plt.xlabel('Time (hours)')
plt.ylabel('Discharge, (m$^3$/s)')
plt.title('Outlet node hydrograph, {} mm/hr for duration of {} hr'.format(
    starting_precip_mmhr, storm_duration/3600.))
plt.show()
print('Maximum discharge at the outlet node = {:.3f} m3/s'.format(max(
    discharge_at_outlet)))
```

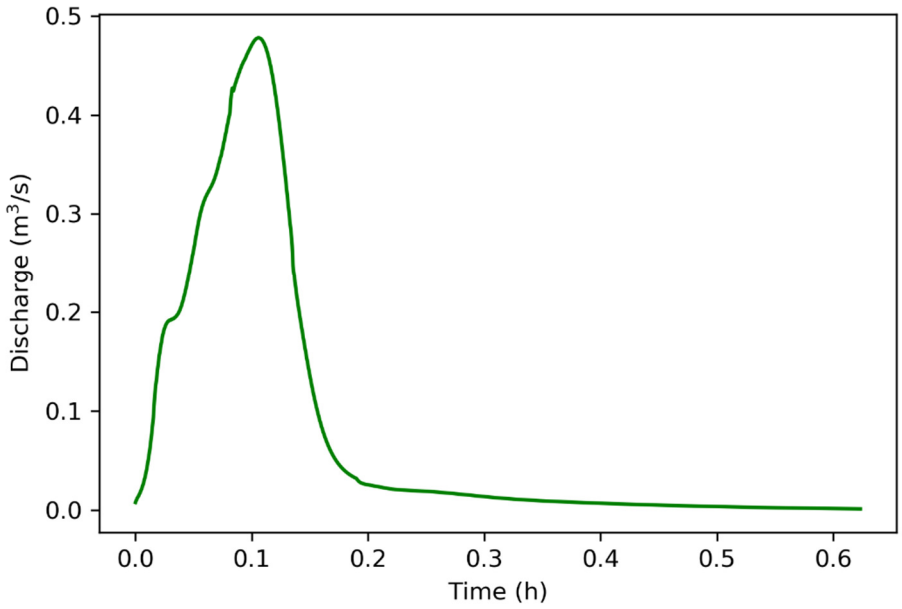


Figure 3.6: Hydrograph for gully topography

```
# Cumulative water flow at outlet node
tot_volume_mid = np.cumsum(np.array(discharge_at_outlet)[1:]*np.diff(np.array(
    hydrograph_time)))*3600.

plt.plot(hydrograph_time[1:], tot_volume_mid)
plt.title('Cumulative water flow')
plt.ylabel('m$^3$')
plt.xlabel('Time (hours)')
print('Cumulative discharge at the outlet node = {:.3f} m3/s'.format(sum(np.abs(
    discharge_at_outlet)[1:]*np.diff(np.array(hydrograph_time))*3600.)))
```

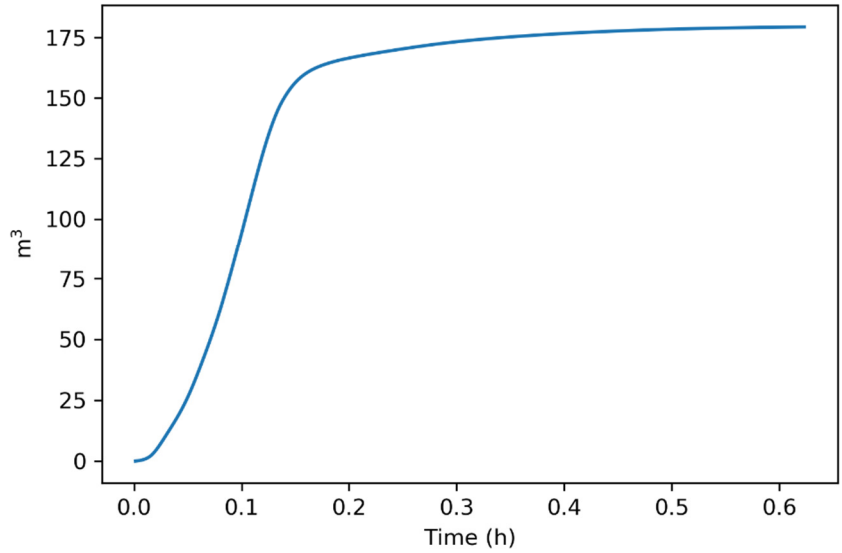


Figure 3.7: Cumulative discharge for gully topography

3.4 CALIBRATION

Actual historical rainfall (UNISA and Bolepi rain stations) and runoff flow data from a catchment area (Brooklyn and Waterkloof), as seen in Figure 3.8, was used to calibrate the simulation. The figure shows the boundary of the catchment area, the longest water course and the outlet, where the flow was measured.

Figure 3.8: Catchment area (Brooklyn and Waterkloof) used for calibration

The calibration process attempts to find values for Manning n and hydraulic conductivity (i.e. soil type) that provide a good match between the measured flow and simulated flow results. The simulation is considered valid and calibrated if the simulation parameters (Manning n and hydraulic conductivity) compared reasonably (order of magnitude correct) against published data. Note that the catchment area contains a mixture of urban and natural terrain cover – the parameters are therefore a gross simplification of the real terrain cover: a mix of built-up area, gardens, asphalt roads and storm water pipes. The range of 'reasonable values' can be easiest defined by identifying the 'unreasonable values' thereby setting the bounds for reasonable

values. In Table 3.3 and Figure 3.9 the unreasonable values, which are clearly not possible are shown in red, thereby defining the reasonable values as the parts not in red.

Considering the values in Table 3.3, a reasonable estimate of the Manning n parameter would be approximately 0.05: neither a clean channel, nor a weedy overgrown area, but a mix of these.

Table 3.3: Manning n values (Chow, 1959)

1. Main Channels			
a. clean, straight, full stage, no rifts or deep pools	0.025	0.030	0.033
b. same as above, but more stones and weeds	0.030	0.035	0.040
c. clean, winding, some pools and shoals	0.033	0.040	0.045
d. same as above, but some weeds and stones	0.035	0.045	0.050
e. same as above, lower stages, more ineffective slopes and sections	0.040	0.048	0.055
f. same as "d" with more stones	0.045	0.050	0.060
g. sluggish reaches, weedy, deep pools	0.050	0.070	0.080
h. very weedy reaches, deep pools, or floodways with heavy stand of timber and underbrush	0.075	0.100	0.150

Figure 3.9 shows typical values of hydraulic conductivity. Given the nature of the built-up area with parks, developed housing, asphalt roads and storm water drains, it is highly unlikely that the conductivity will correspond with clay or even silt soil, neither with coarse gravel. A reasonable average value over all of the catchment area would be the equivalent of sandy loam, being reasonably conducting but not overly high.

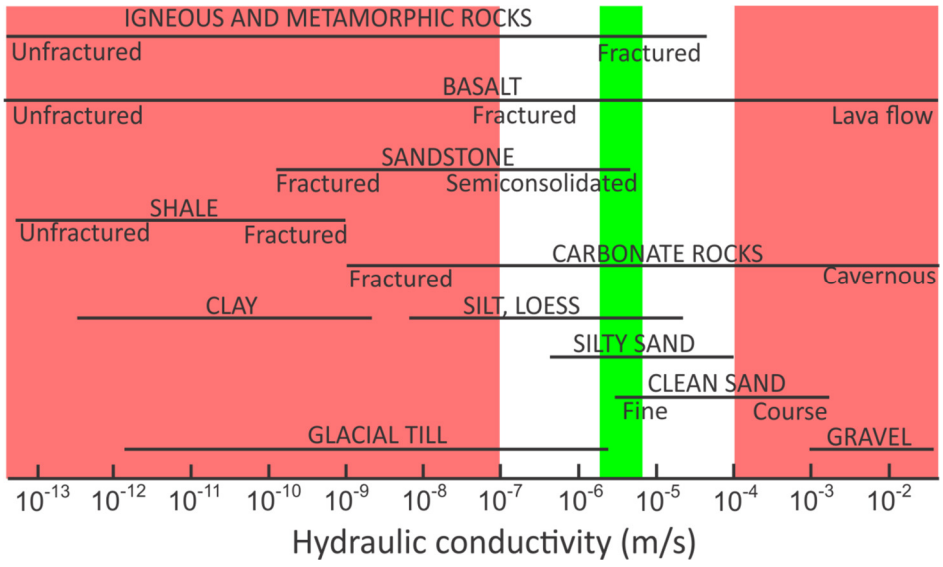


Figure 3.9: Hydraulic conductivity (AQTESOLV, 2016)

The following process was followed to calibrate the model. The contours of the topographical basin were exported to an ESRI ASCII file (see Appendix A). As can be seen in Figure 3.10 the shape and topographical elements are maintained in the conversion of the file. The areas in black fall outside of the catchment area and define the boundary of the catchment for Landlab usage.

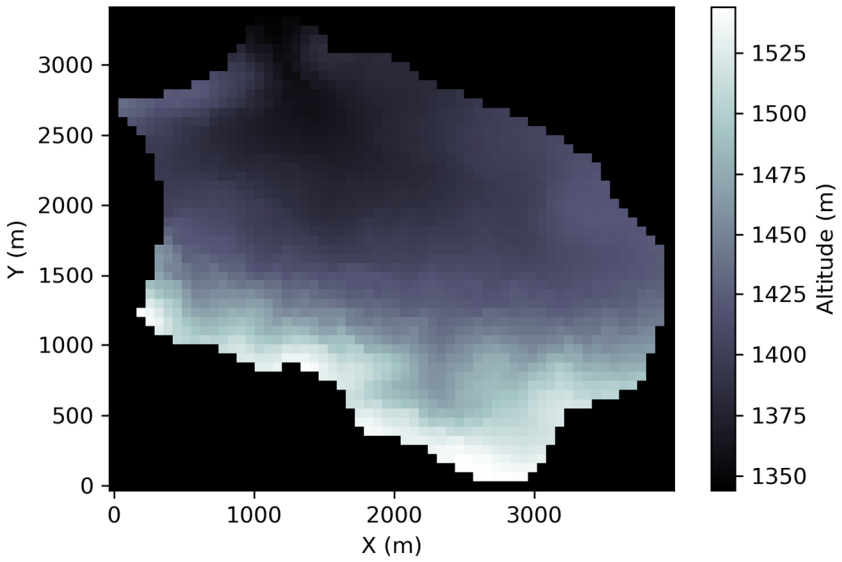


Figure 3.10: Visualisation of calibration catchment topography grid

Historical rainfall and runoff data was analysed. Five rainfall sessions with the highest measured runoff flow (Figure 3.11) and the corresponding best-fit rainfall data from UNISA or Bolepi were identified.

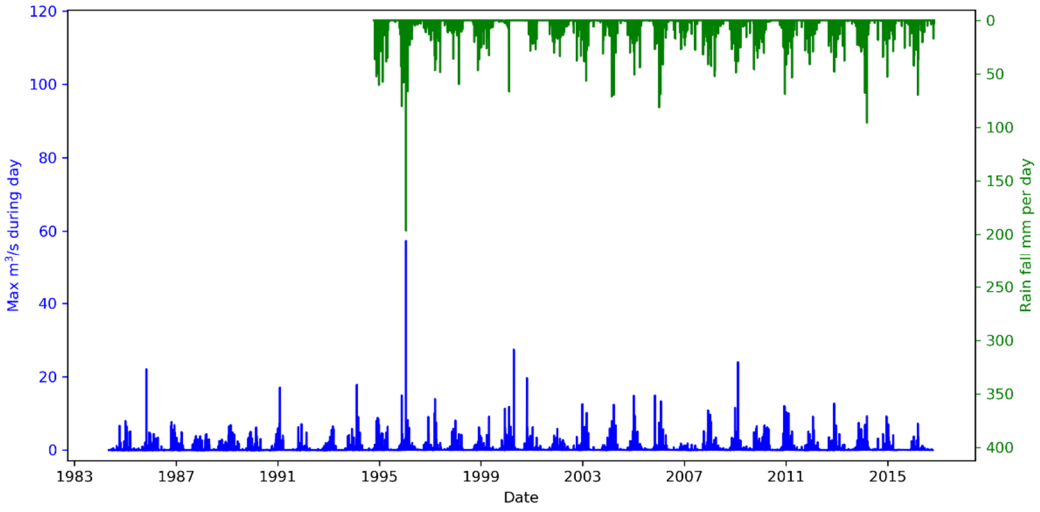


Figure 3.11: Measured flow and rainfall in the catchment for 1983 to 2016

Outliers in measured flow have not been used, as they can be measurement errors due to faulty machinery or from pipe bursts, rather than rainfall measurements. An average rainfall for the specific day was calculated and used in the simulation, on the relevant catchment area.

As all the above data was given, and can be assumed to be accurate, the only unknowns to be adjusted were the soil initial conditions, Manning n and hydraulic conductivity. An iterative process was followed to determine the appropriate values to match the measured flow. The specific code used to execute the calibration tests can be found in Appendix A.

The Landlab *SoilInfiltrationGreenAmpt* component requires the specification of soil type as well as hydraulic conductivity. The soil type was chosen to be a sandy loam, with hydraulic conductivity ranging from $4x10^{-6}$ m/s to $6x10^{-6}$ m/s. A better investigation could be to use Google Earth pictures to identify the land cover (and hence conductivity) in smaller areas, but this is outside the scope of the current study. An average Manning n of 0.05 was assumed and the water column height previously absorbed into the soil was set so 0.15 m. As the initial conditions before the storm are unknown, the model initial surface water depth was adjusted for each storm to best fit the required match on the measured flow.

As can be seen from Table 3.4 the measured and simulated flows are close to each other for hydraulic conductivity values of $5.0x10^{-6}$ m/s to $5.5x10^{-6}$ m/s. The Landlab overland flow and soil infiltration simulation components give realistic results for the real world terrain data, and are therefore considered fit for use.

Table 3.4: Measured and simulated flow for the calibration catchment area

Date	Measured flow (m^3/s)	Simulated flow (m^3/s)	Hydraulic conductivity range (m/s)	Initial surface water depth (m)
	Required match		**Best-fit parameters**	
1995-11-18	14.887	19.949	$4.0x10^{-6}$ to $4.5x10^{-6}$	0.020
		17.140	$4.5x10^{-6}$ to $5.0x10^{-6}$	
		14.401	$5.0x10^{-6}$ to $5.5x10^{-6}$	
		11.834	$5.5x10^{-6}$ to $6.0x10^{-6}$	
1997-03-11	13.891	23.702	$4.0x10^{-6}$ to $4.5x10^{-6}$	0.003
		18.819	$4.5x10^{-6}$ to $5.0x10^{-6}$	
		13.840	$5.0x10^{-6}$ to $5.5x10^{-6}$	
		7.290	$5.5x10^{-6}$ to $6.0x10^{-6}$	
2009-02-10	23.926	28.036	$4.0x10^{-6}$ to $4.5x10^{-6}$	0.022
		25.012	$4.5x10^{-6}$ to $5.0x10^{-6}$	
		22.005	$5.0x10^{-6}$ to $5.5x10^{-6}$	
		19.129	$5.5x10^{-6}$ to $6.0x10^{-6}$	
2010-12-08	11.981	17.575	$4.0x10^{-6}$ to $4.5x10^{-6}$	0.018
		13.914	$4.5x10^{-6}$ to $5.0x10^{-6}$	
		11.276	$5.0x10^{-6}$ to $5.5x10^{-6}$	
		8.838	$5.5x10^{-6}$ to $6.0x10^{-6}$	
2012-11-24	12.643	19.264	$4.0x10^{-6}$ to $4.5x10^{-6}$	0.019
		16.273	$4.5x10^{-6}$ to $5.0x10^{-6}$	
		13.290	$5.0x10^{-6}$ to $5.5x10^{-6}$	
		10.520	$5.5x10^{-6}$ to $6.0x10^{-6}$	

The hydrograph and cumulative flow from the calibration simulation for the rainfall on 2009-02-10 are shown in Figure 3.12 and Figure 3.13. As can be seen from the table and figures, the simulated flow is well inside the approved area of calibration. Thus, the calibration was considered successful.

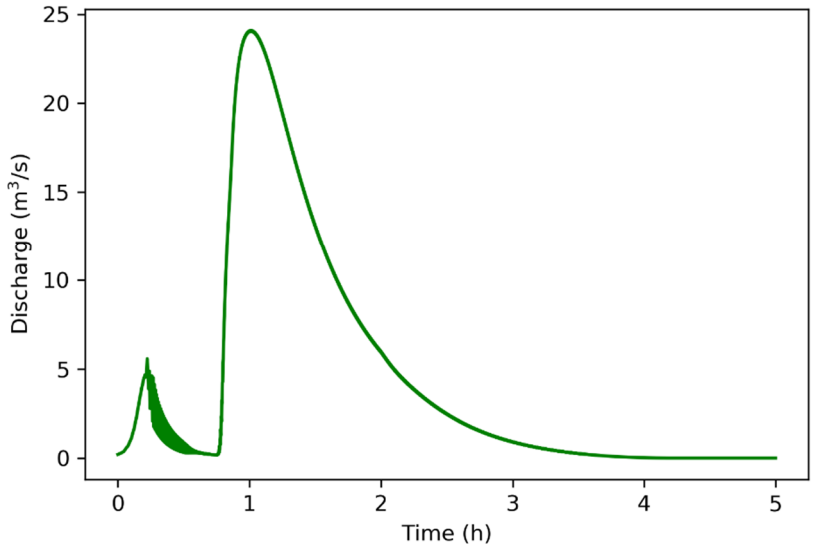


Figure 3.12: Hydrograph of simulated rainfall for 2009-02-10

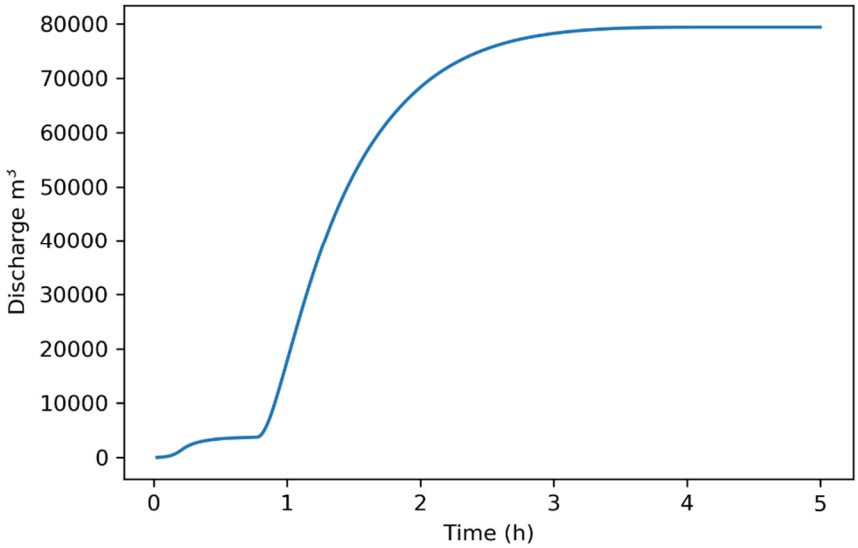


Figure 3.13: Cumulative discharge at the outlet node of simulated rainfall for 2009-02-10

3.5 SIMULATION PARAMETER SETS

For experimentation purpose the rainfall runoff produced by a 50 mm/h storm, applied for one hour, was measured at the lowest discharge node point of a synthetic valley topography. Details of the experimental procedure is shown in Appendix B. The simulation input topography file was generated using the topography generation utility provided in Appendix A. Slopes between 2° and 25° were considered to be in a realistic range. Figure 3.14 shows the valley topography with the 5° slopes used in experimentation. The topography boundary conditions represent a watershed at all edges except at the single discharge node point at the top-right corner, i.e., all discharge leave the topography only at the top-right discharge node.

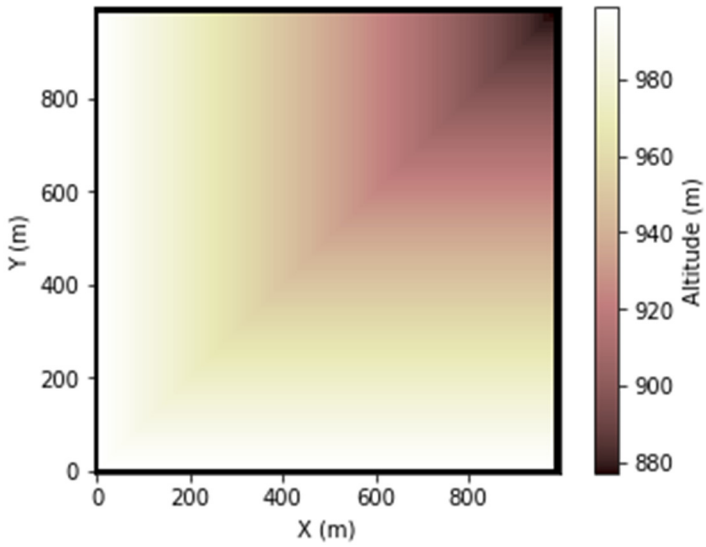


Figure 3.14: Valley topography with 5° slopes, discharge measured at lowest altitude at the top-right corner

For each of the defined topographies, the soil hydraulic conductivity and Manning n values were varied and the flow measured at the lowest point on the topography. Manning n was varied in the range from 0.03 to 0.07 in steps of 0.01, while the soil hydraulic conductivity was varied from 1.6×10^{-5} m/s to 8.3×10^{-7} m/s, covering a range of 10 soil types. These combinations of parameters resulted in 300 simulation experiments (6 slopes x 5 Manning n values x 10 soil infiltration/type values). See Appendix B for the Python code used to execute the experiments.

The hydrograph data was saved to a disk for each combination of the three experimental parameters. Figure 3.15 shows a typical hydrograph from the simulation output, comparing the discharge for the various soil infiltration/types used in the experiment (see Table 3.2).

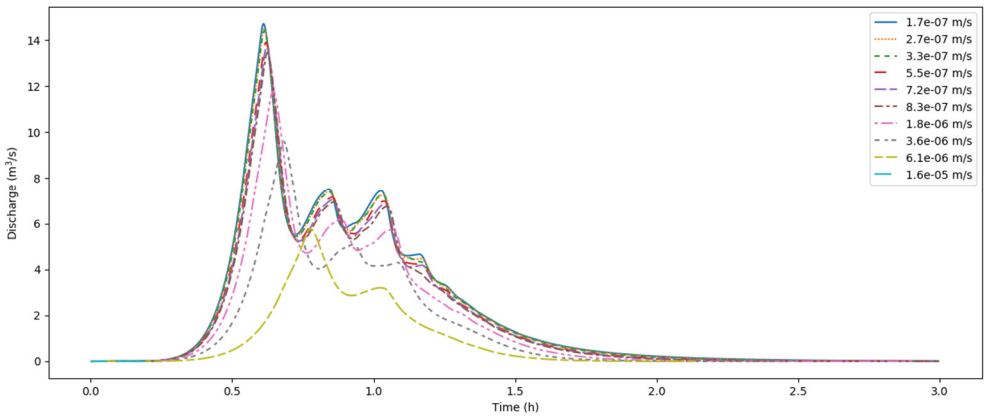


Figure 3.15: Hydrograph for various hydraulic conductivity values. Storm of 50 mm/h, duration one hour on a valley with 5˚ slope, initial infiltration depth of 0.1 m and a Manning n of 0.03

The hydrograph data was also used to visualise the cumulative and maximum discharge measured. Once again these graphs were generated for all combinations of the three free parameters in the experiment. The Python script used for visualisation of the simulation output data is discussed in Appendix B. Figure 3.16 shows an example of the maximum and cumulative discharge measured on all topographies with the Manning n constant at 0.03 and the soil type and terrain slope varied within the defined ranges.

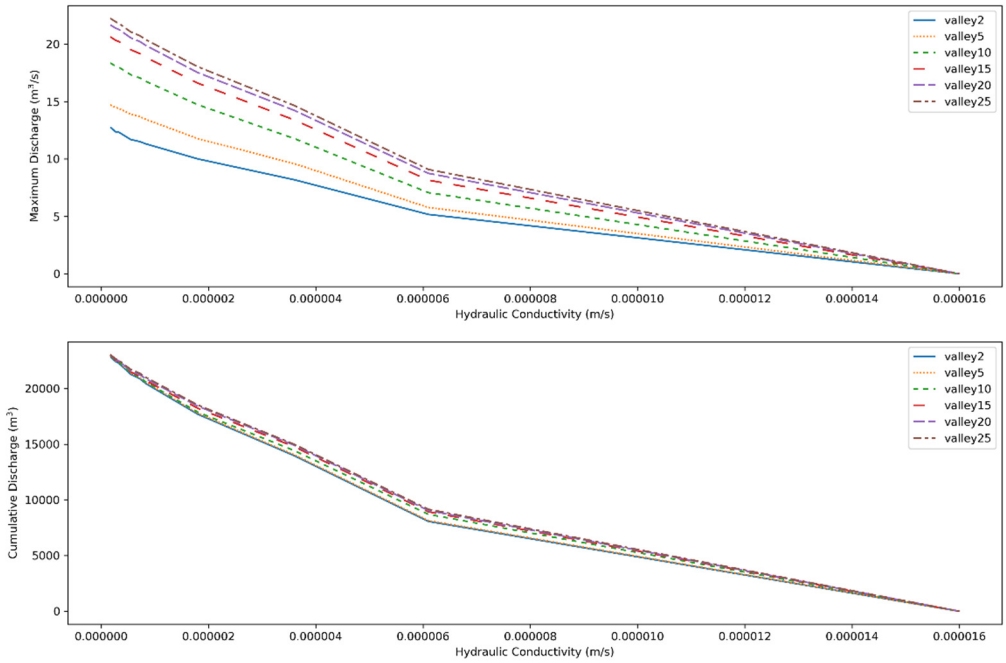


Figure 3.16: Maximum and cumulative discharge for various soil hydraulic conductivities and terrain slopes. Storm of 50.0 mm/h, duration one-hour initial infiltration depth of 0.1 m and a Manning n of 0.03

These graphs visually show the mutual influence of the combinations of parameters on the measured flow.

4 DISCUSSION OF RESULTS

4.1 INTRODUCTION

This Chapter summarises the results from the 300 experiments executed with the Landlab rainfall runoff model. In the data analysis process, one of the three variables used in simulation was kept constant and graphs drawn showing the change in the discharge at the outlet point as a function of the other two free variables.

4.2 THE IMPACT OF SLOPE ON RUNOFF

4.2.1 Hydrographs for varying slope

The hydrographs calculated in simulation for varying topographical slope were plotted for an increasing hydraulic conductivity (increase in infiltration), constant Manning n of 0.05 and infiltration depth of 0.1 m. Figure 4.1, Figure 4.2 and Figure 4.3 respectively show the hydrographs for three of the soil types: clay, clay loam and sandy loam.

In all three graphs, the maximum value of the hydrograph decreases as the terrain slope decreases. For topographies with a small gradient, more of the rainfall is infiltrated into the soil and the measurement at the outlet point is lower. The steeper the valley, the higher the measurement of flow at the outlet point. Note that in general, the time of the maximum measurement at the outlet point is longer for valleys with lower slopes.

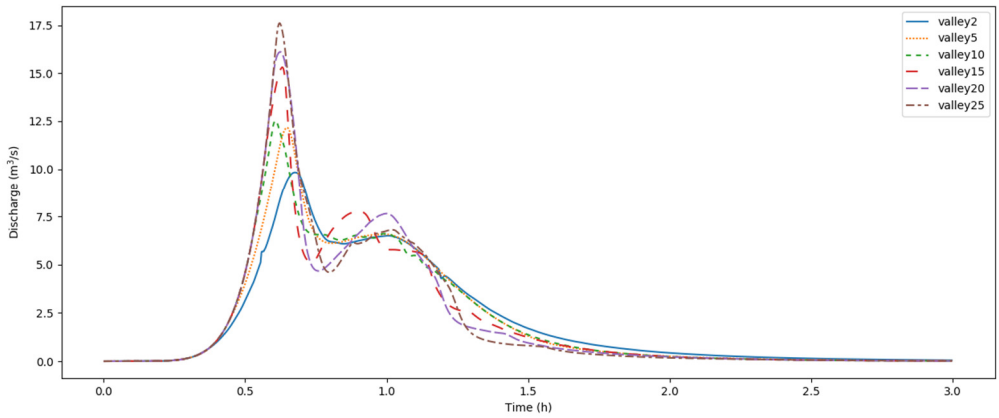


Figure 4.1: Hydrograph for various terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, clay soil hydraulic conductivity of 1.7×10^{-7} m/s and a Manning n of 0.05

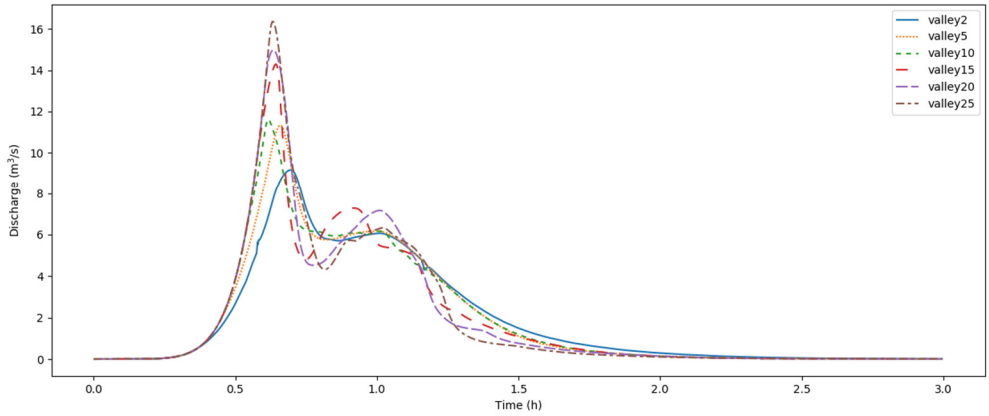


Figure 4.2: Hydrograph for various terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, clay loam soil hydraulic conductivity of 7.2×10^{-7} m/s and a Manning n of 0.05

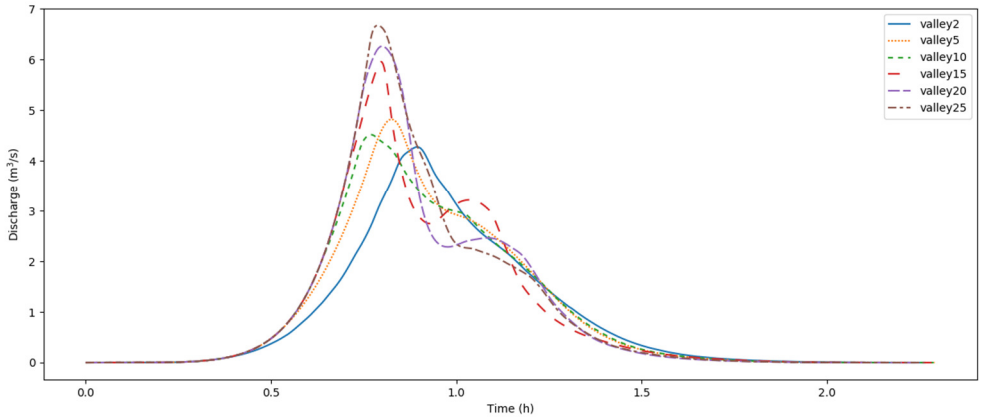


Figure 4.3: Hydrograph of various terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, sandy loam soil hydraulic conductivity of 6.1×10^{-6} m/s and a Manning n of 0.05

From these hydrographs it is also evident that as the hydraulic conductivity increases, the maximum amount of discharge decreases. It is interesting to note that the time of maximum discharge is in the same order for the clay and clay loam soils but shifts to longer times for the sandy loam. This observation is as expected because more water infiltrates into the soil and it

will therefore take longer for the discharge at the outlet point to reach the maximum value. Thus, as the soil hydraulic conductivity increases, the runoff infiltrates faster into the soil, thus the effect of slope has a smaller impact on the discharge.

4.2.2 Maximum and cumulative discharge graphs for varying slope

The maximum and cumulative discharge values measured at the outlet point were plotted for various terrain slopes and Manning n values. Figure 4.4, Figure 4.5 and Figure 4.6 present the graphs for three different soil hydraulic conductivities, i.e. clay, clay loam and sandy loam. As shown in these graphs, the higher terrain slopes generate a higher discharge, as was expected. For a specific soil hydraulic conductivity, the discharge increases as the Manning n value decrease, i.e. as the surface roughness decreases.

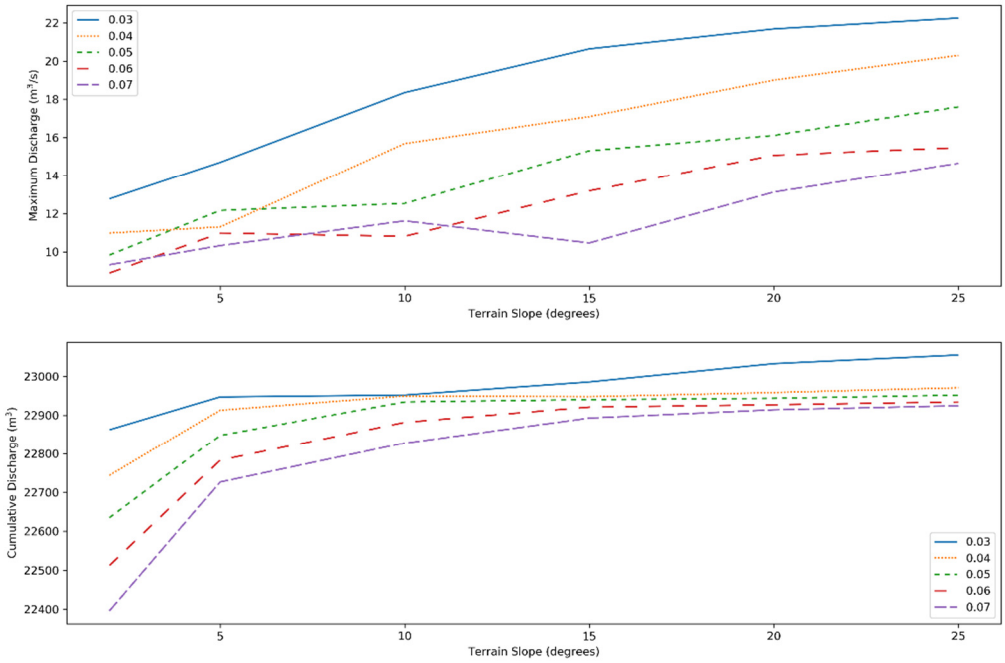


Figure 4.4: Maximum and cumulative discharge for various terrain slopes and Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a clay soil hydraulic conductivity of 1.7×10^{-7} m/s

Observing all three figures, it is noted that as the hydraulic conductivity increases, the water infiltration into the soil becomes faster and the flow measured at the outlet point is lower.

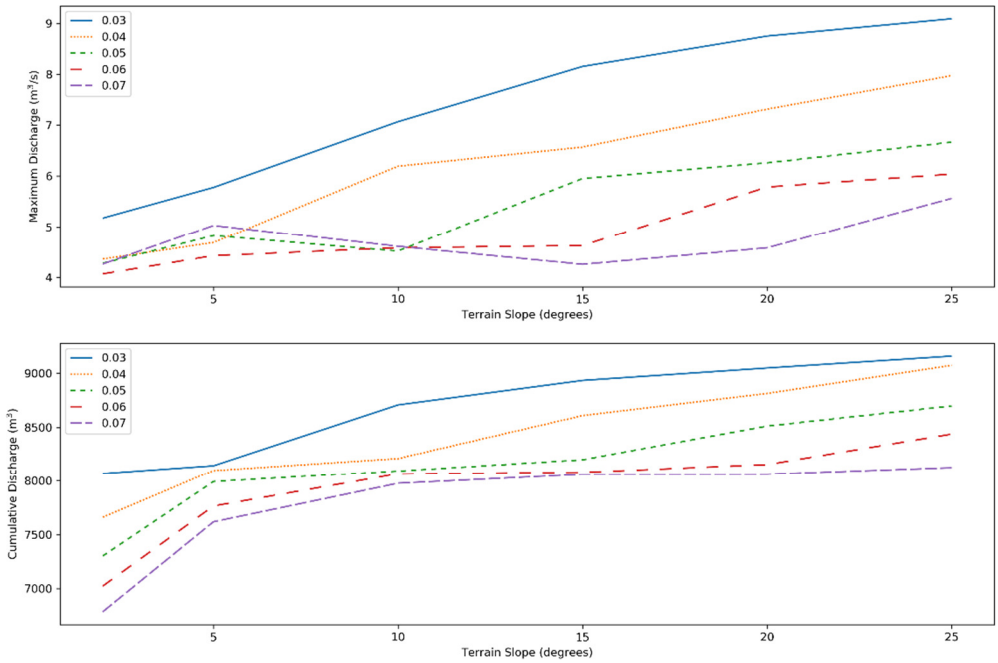


Figure 4.5: Maximum and cumulative discharge for various terrain slopes and Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a clay loam soil hydraulic conductivity of 7.2x10-7 m/s

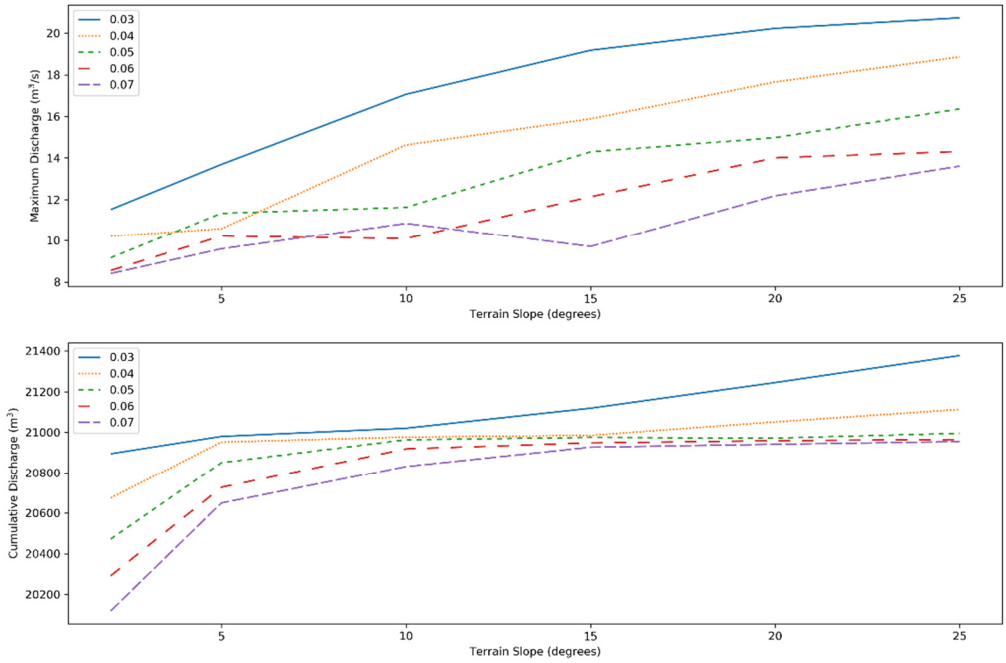


Figure 4.6: Maximum and cumulative discharge for various terrain slopes and Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a sandy loam soil hydraulic conductivity of $6.1x10^{-6}$ m/s

4.3 THE IMPACT OF VEGETATION AND DEVELOPMENT ON RUNOFF

4.3.1 Hydrographs for varying Manning n

The hydrographs for simulation experiments on the 5° sloped valley and varying Manning n values are presented in Figure 4.7, Figure 4.8 and Figure 4.9 for soil hydraulic conductivity of clay, clay loam and sandy loam. It is expected that as the surface roughness increases, i.e. the Manning n value larger, the flow of water on the terrain will be slower and the measured discharge at the outlet point will be lower. This is indeed observed in these graphs. The lowest Manning n value of 0.03 has the largest discharge and the highest value the lowest discharge.

Comparing the three hydrographs, it is noted that as the hydraulic conductivity increases, the maximum discharge decreases. The sandy loam soil with the highest hydraulic conductivity has the largest impact on the measured discharge. It is once again noted that the time of maximum discharge measurement shifts to a larger value for the sandy loam soil. For a terrain with a high soil hydraulic conductivity and terrain coverage, the water will infiltrate faster, thus lessening the measured water discharge at the outlet point.

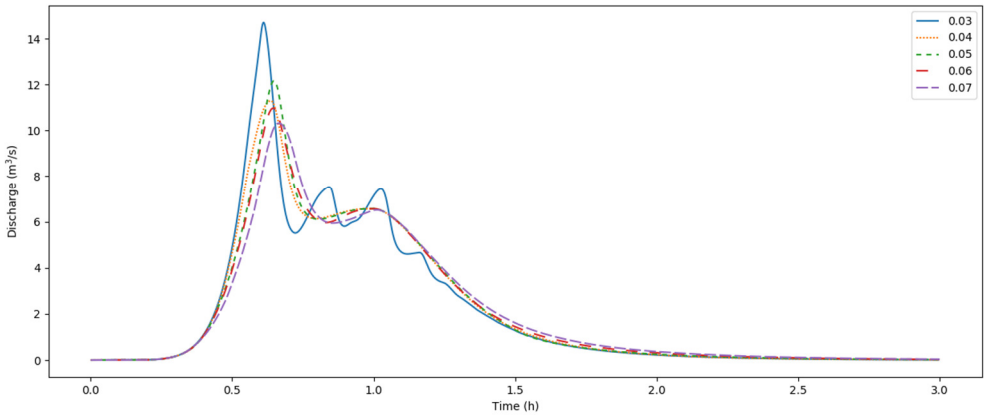


Figure 4.7: Hydrograph for various Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, clay soil hydraulic conductivity of 1.7×10^{-7} m/s and a terrain slope of 5 degrees

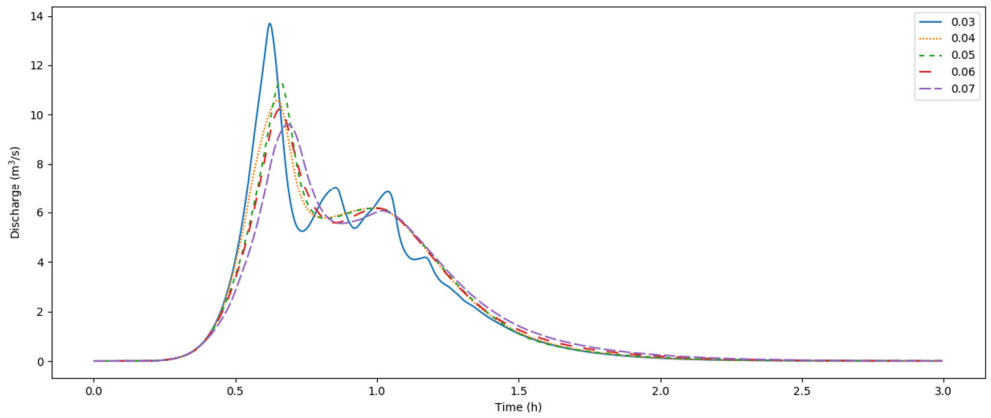


Figure 4.8: Hydrograph for various Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, clay loam soil hydraulic conductivity of $7.2x10^{-7}$ m/s and a terrain slope of 5 degrees

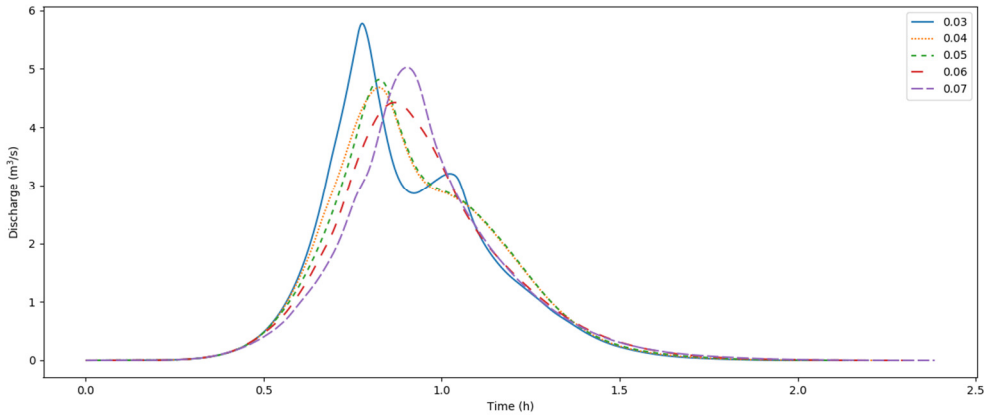


Figure 4.9: Hydrograph for various Manning n values. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, sandy loam soil hydraulic conductivity of $6.1x10^{-6}$ m/s and a terrain slope of 5 degrees

4.3.2 Maximum and cumulative discharge graphs for varying Manning n

The maximum and cumulative discharge values were plotted for experiments with varying Manning n and varying terrain slope. Three graphs are presented in Figure 4.10, Figure 4.11 and Figure 4.12 for clay, clay loam and sandy loam soils. In all three sets of graphs, the maximum and cumulative discharge decrease with increase in Manning n. This statement is also true for the terrain slope, the steeper the slope, the higher the measured discharge.

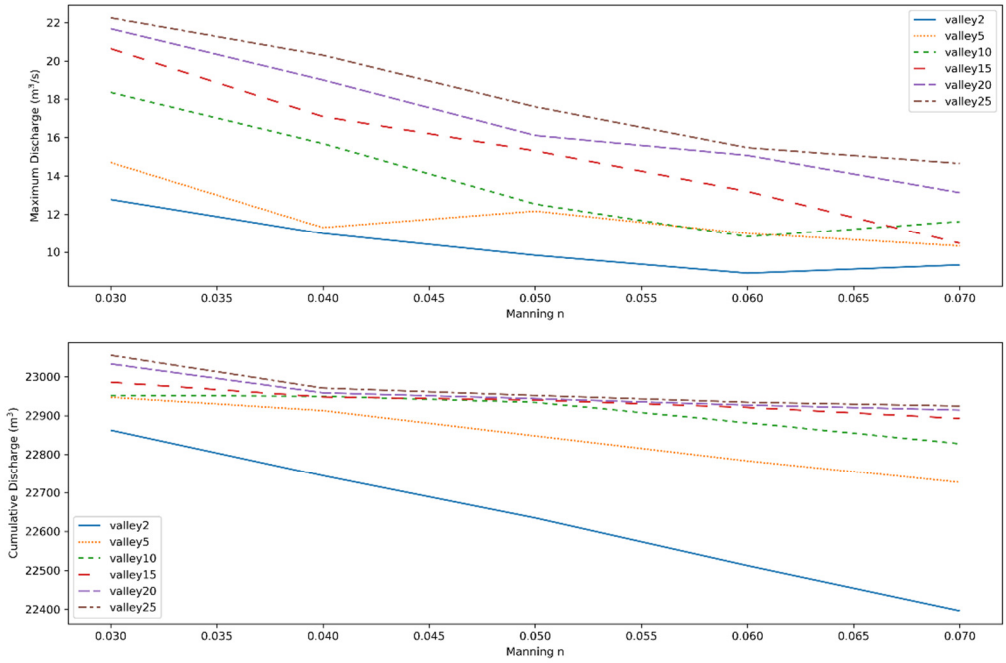


Figure 4.10: Maximum and cumulative discharge for various Manning n values and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a clay soil hydraulic conductivity of 1.7×10^{-7} m/s

For the high hydraulic conductivity sandy loam soil, the water infiltrates quicker, and the measured maximum, as well as cumulative discharge, are much lower than for the lower hydraulic conductivity soils.

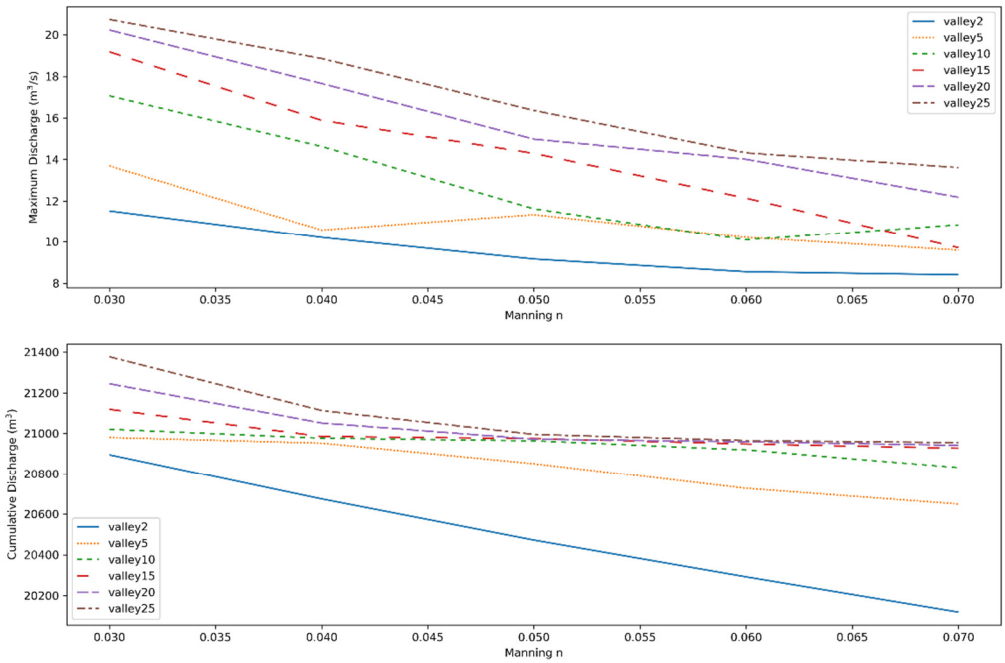


Figure 4.11: Maximum and cumulative discharge for various Manning n values and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a clay loam soil hydraulic conductivity of 7.2×10^{-7} m/s

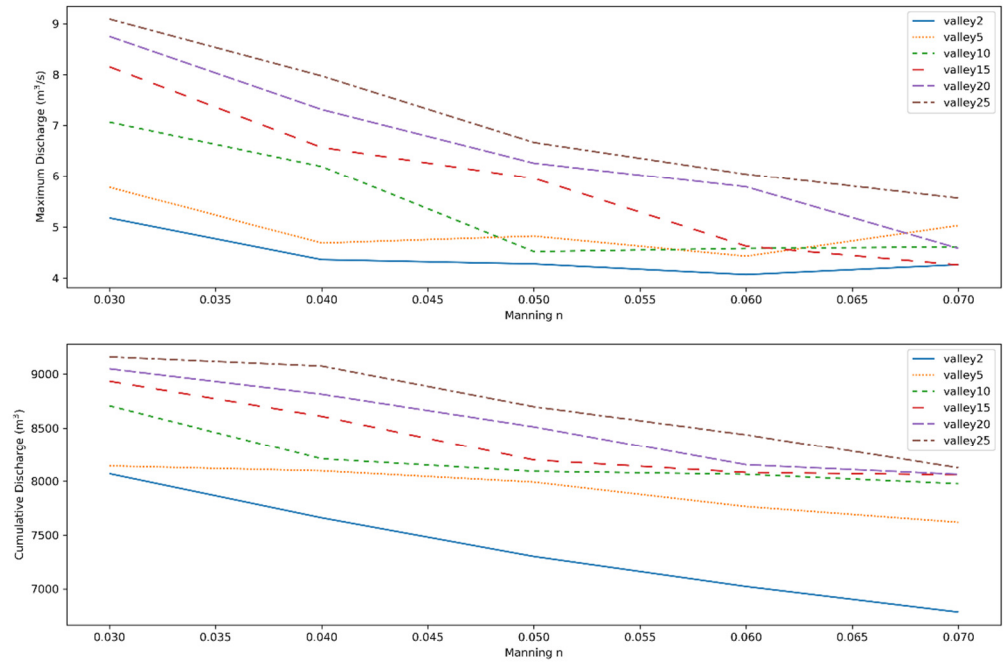


Figure 4.12: Maximum and cumulative discharge for various Manning n values and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a sandy loam soil hydraulic conductivity of 6.1×10^{-6} m/s

4.4 THE IMPACT OF INFILTRATION ON RUNOFF

4.4.1 Hydrographs for varying hydraulic conductivity

The hydrographs for varying soil hydraulic conductivity was plotted for a valley with a constant slope of 5˚. Figure 4.13, Figure 4.14 and Figure 4.15 respectively shows the hydrograph for Manning n values of 0.03, 0.05 and 0.07. The hydrographs show that as the water infiltrates the soil faster with increasing soil hydraulic conductivity, the measured discharge at the outlet point on the terrain decreases. Comparing the peak values in the hydrographs in the three figures, it is evident that as the Manning n value increases, i.e. as the vegetation increases, the discharge decreases.

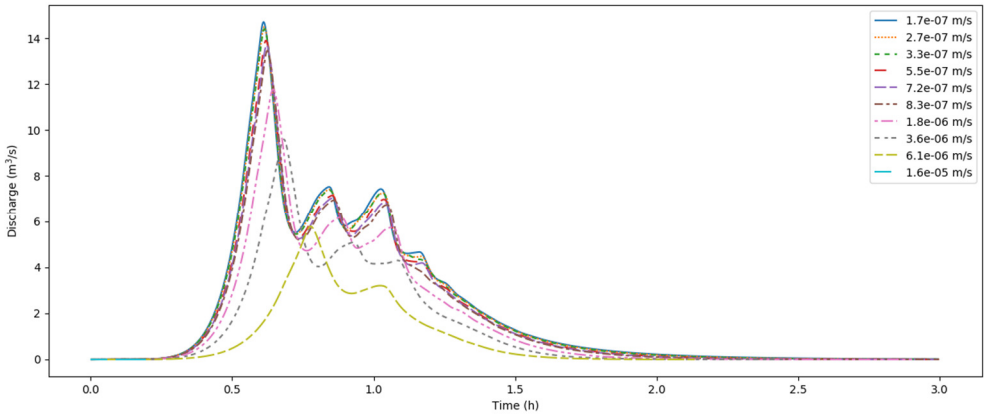


Figure 4.13: Hydrograph of various hydraulic conductivities. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, terrain slope of 5 degrees and Manning n of 0.03

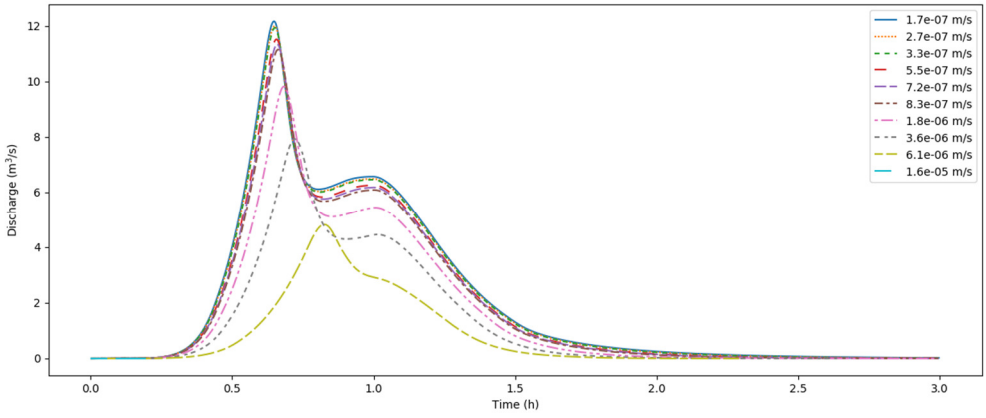


Figure 4.14: Hydrograph of various hydraulic conductivities. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, terrain slope of 5 degrees and Manning n of 0.05

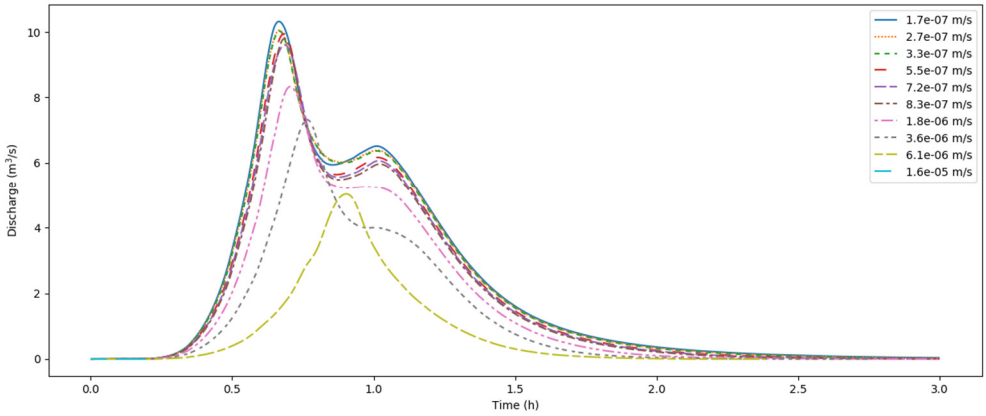


Figure 4.15: Hydrograph of various hydraulic conductivities. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m, terrain slope of 5 degrees and Manning n of 0.07

4.4.2 Maximum and cumulative discharge for varying soil hydraulic conductivity

The maximum and cumulative discharge measured at the outlet point was plotted for all valley slopes and the range of soil hydraulic conductivity values used in the experiments. Figure 4.16, Figure 4.17 and Figure 4.18 show the graphs for three values of Manning n. The graphs show that the discharge measured decreases as the Manning n value increases, thus as the vegetation increases. The steeper slope in the graphs confirms that soil hydraulic conductivity smaller than $6x10^{-6}$ m/s leads to a larger measured discharge, i.e. loam and clay soil types.

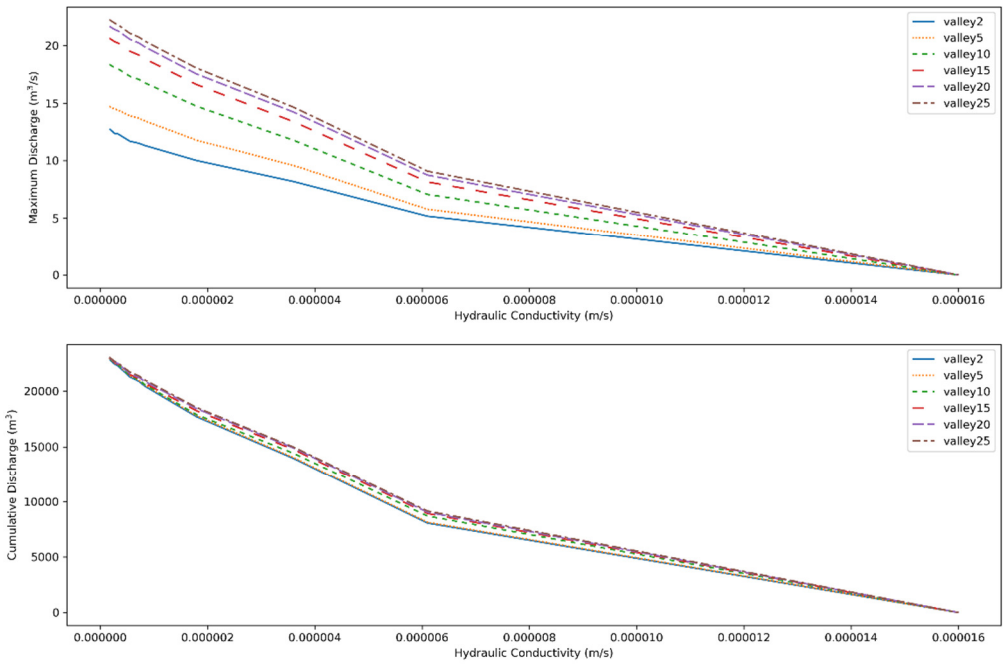


Figure 4.16: Maximum and cumulative discharge for various soil hydraulic conductivities and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a Manning n of 0.03

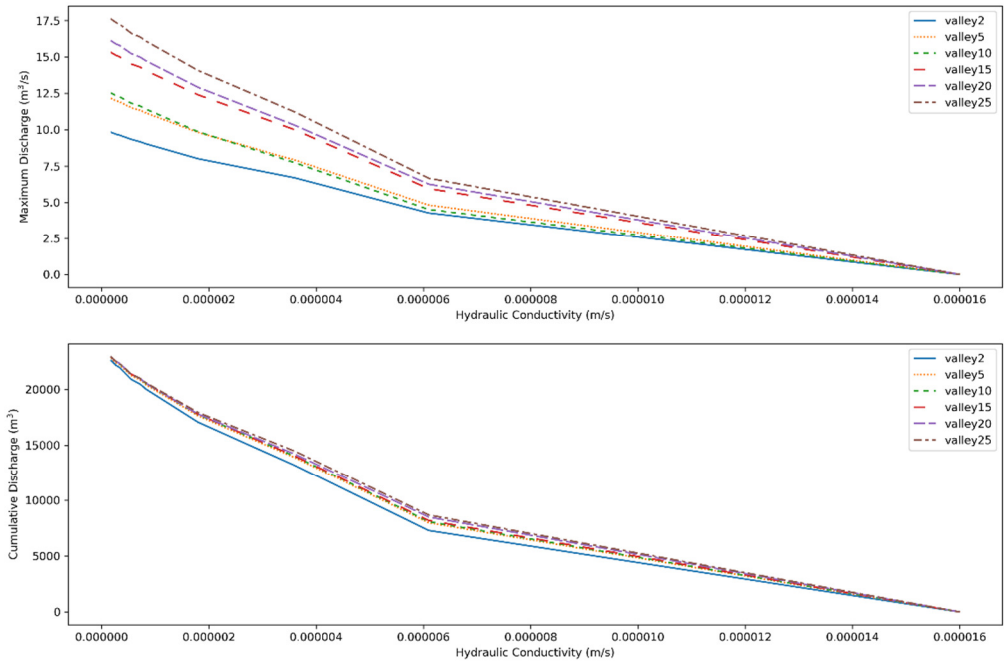


Figure 4.17: Maximum and cumulative discharge for various soil hydraulic conductivities and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a Manning n of 0.07

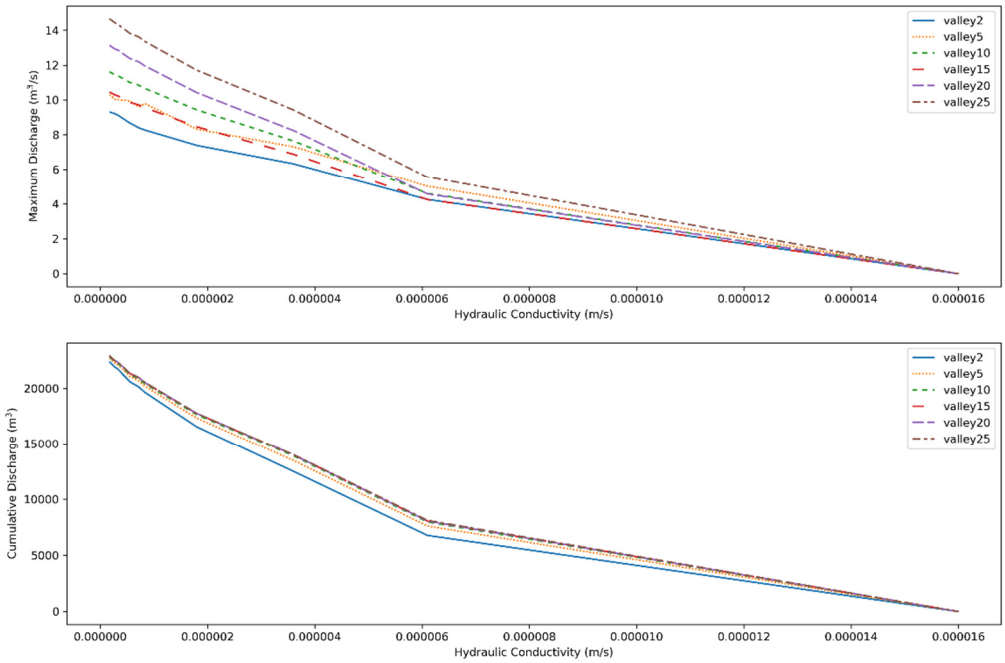


Figure 4.18: Maximum and cumulative discharge for various soil hydraulic conductivities and terrain slopes. Storm of 50.0 mm/h, duration one hour, initial infiltration depth of 0.1 m and a Manning n of 0.05

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The water runoff -flow was measured at the lowest point on a valley catchment area following a simulated one-hour storm with 50 mm/h intensity. The conclusions in this chapter are only valid for this storm, other storm intensities and/or durations will lead to different results, and therefore, different conclusions. Three parameters were varied during experimentation: valley slope, Manning n (terrain coverage) and soil hydraulic conductivity. The 300 simulation experiments were analysed by studying the hydrographs generated using the Landlab simulation model.

The simulation results confirmed that the terrain slope, coverage and soil type directly affect the water discharge measured after a rainfall storm and are indeed all interlinked. The soil hydraulic conductivity was found to have the largest impact on water runoff.

The heatmap graphs presented in this chapter contain summary information of the hydrographs presented in Chapter 4. Each block or pixel in the heatmap represent a result from a different hydrograph. Each heatmap therefore contains the summary information from a very large number of hydrographs. However, the heatmaps do not display the time-dependent discharge shown in the hydrographs; it only displays the peak discharge, cumulative discharge or time-to-maximum discharge.

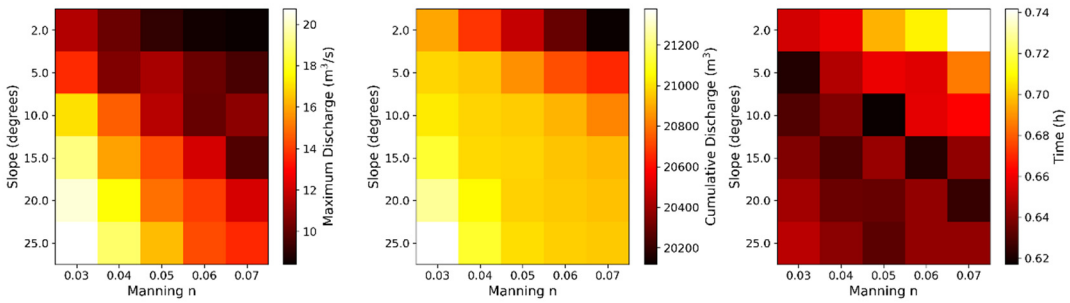


Figure 5.1: Maximum discharge (left), cumulative discharge (middle) and time of maximum discharge (right) measured at the outlet point for a clay loam soil with hydraulic conductivity of 7.2×10^{-7} m/s

Figure 5.1 depicts the maximum and cumulative discharge as well as the time of the maximum discharge measured at the outlet point for a clay loam soil with hydraulic conductivity of 7.2×10^{-7} m/s. The discharge values are shown for all Manning n and slope values covered in the experiments. This presentation of the data shows that as the slope

decreases and the Manning n value increases, the discharge decreases. Thus, as the terrain topography becomes less steep, and the catchment area coverage denser, more water infiltrates into the soil and the discharge at the lowest point on the terrain decreases. This behaviour was noted for all soil hydraulic conductivity values used in the experiment. As can be expected, as the terrain became rougher and the slope flatter, the time it took to reach its maximum discharge increased. The heatmaps in Figure 5.1 show that the relationship between terrain slope and Manning n is almost linear and well-behaved within the bounds of this experiment.

Comparing the measured discharge for either Manning n (Figure 5.2), or terrain slope (Figure 5.3), with soil hydraulic conductivity, the impact of the soil type on the water runoff is highlighted clearly. As the hydraulic conductivity increases, the infiltration of the water into the soil becomes easier and the discharge at the outlet point lower. In both comparisons the water infiltration for the sandy soil type totally overshadows the variation in terrain coverage as well as terrain slope.

For a nearly flat terrain (left column in the heatmaps in Figure 5.2), or densely covered (right column in the heatmaps in Figure 5.3), the maximum measured discharge is low for all soil types. The flat topography and high Manning n dominates the maximum flow of the water in these conditions.

On the other hand, for a terrain with a steep slope (right column in Figure 5.2), or a sparsely covered terrain (left column in Figure 5.3), the maximum water runoff measured is high for soils with hydraulic conductivity larger than 3.6×10^{-6} m/s. It is only the sandy soil types (higher hydraulic conductivity) that lower the maximum discharge measurement.

By comparing the cumulative flow and time of maximum discharge graphs, it is evident that the soil type is the dominant factor, overshadowing the terrain slope as well as coverage.

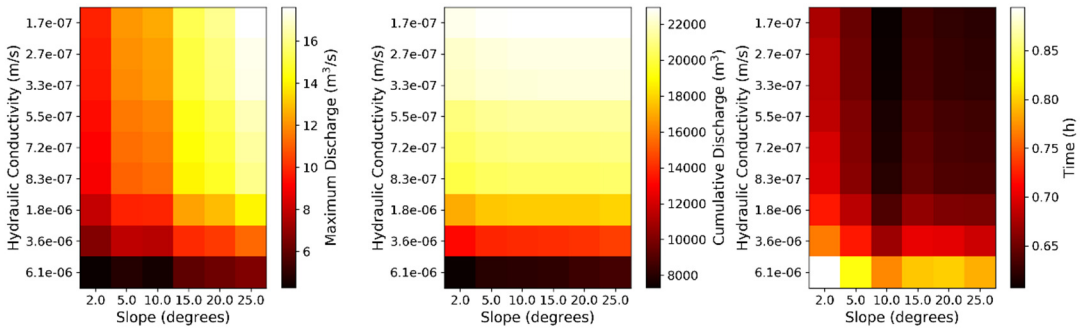


Figure 5.2: Maximum discharge (left) cumulative discharge (middle) and time of maximum discharge (right) measured at the outlet point for terrain coverage with Manning n of 0.05

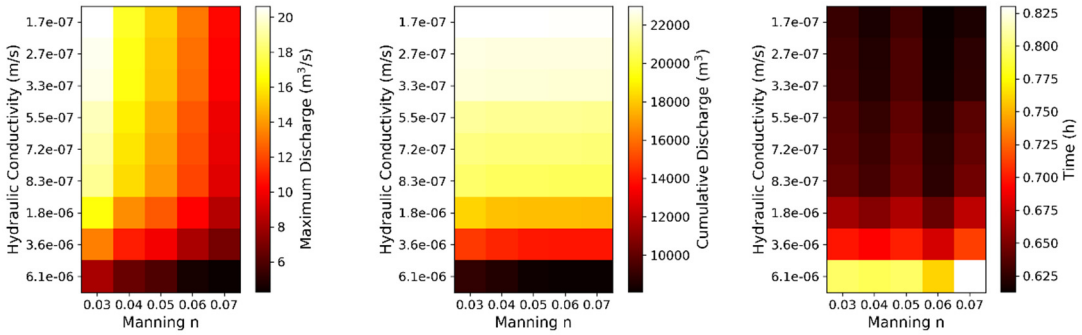


Figure 5.3: Maximum discharge (left) cumulative discharge (middle) and time of maximum discharge (right) measured at the outlet point for a valley with flat slopes of 15 degrees

5.2 RECOMMENDATIONS

This study focused on the rainfall runoff measurement of a specific simulated storm on a valley-type of catchment area. The terrain slope, coverage and soil types were limited to well-defined ranges. In a subsequent study, a wider range of free variables can be considered. The Landlab simulation environment can also be expanded and improved by implementing more advanced land overflow and soil infiltration models. As confidence in the simulation grows, the model can be used in practise as an investigation tool to aid in better understanding of the infiltration and speed of runoff of water within a certain calibrated catchment area. Practical application will require a reasonably good spatial model of a real terrain, in terms of Manning n, slope and soil hydraulic conductivity. In simulating these elements, the application of a design will be more efficient and trustworthy.

6 REFERENCES

2016, *Geen-Ampt Equation,* lecture notes, Geosynthetics and their applications CE 551 Indian Institute of Technology Guwahati

ADAMS, J. M., GASPARINI, N. M., HOBLEY, D. E. J., TUCKER, G. E., HUTTON, E. W. H., NUDURUPATI, S. S. & ISTANBULLUOGLU, E. 2017. The Landlab v1.0 OverlandFlow component: a Python tool for computing shallow-water flow across watersheds. *geoscientific model development* [Online], 10.

ALMEIDA, G. A. M., BATES, P., FREER, J. E. & SOUVIGNET, M. 2012. Improving the stability of a simple formulation of the shallow water equations for 2-D flood modeling. *Water Resources Research,* 48.

AQTESOLV, 2016. *HydroSOLVE, Inc.,* 09/04/2018, <http://www.aqtesolv.com/aquifer-tests/aquifer_properties.htm>

ASSOULINE, S. 2013. Infiltration into soils: Conceptual approaches and solutions. *Water Resources Research,* 49**,** 1755-1772.

BRODY, S., KIM, H. & GUNN, J. 2013. Examining the Impacts of Development Patterns on Flooding on the Gulf of Mexico Coast. *Urban Studies,* 50**,** 789-806.

CHOW, V. T. 1959. *Open-channel hydraulics,* New York :, McGraw-Hill.

CJ Willers & MS Willers, *OSSIM: Optronics System Simulator*, White Paper Technical Report 6700-PG-103400-01 RPT Rev 5, CSIR, 2014.

GOWDISH, L. & MUÑOZ-CARPENA, R. 2009. An improved Green–Ampt infiltration and redistribution method for uneven multistorm series. *Vadose Zone Journal,* 8**,** 470-479.

HOBLEY, D. E. J., ADAMS, J.M., SAI, S.N., HUTTON, E.W.H., GASPARINI, N.M., ISTANBULLUOGLU, E. & TUCKER, G.E. 2017. "Creative computing with Landlab: an open-source toolkit for building, coupling, and exploring two-dimensional numerical models of Earth-surface dynamics". *Earth Surface Dynamics,* vol. 5**,** pp. 21-46.

Jordan Adams, 2017. *Github, Inc,* 22/03/2018, <https://github.com/landlab/landlab/wiki/OverlandFlow-Component-Users-Manual>

KIM, H. W., KIM, J.-H., LI, W., YANG, P. & CAO, Y. 2017. Exploring the impact of green space health on runoff reduction using NDVI. *Urban Forestry & Urban Greening,* 28**,** 81-87.

margauxmouchene, 2018. *Github, Inc*, 22/03/2018, <https://github.com/landlab/landlab/wiki/Components>

MASOUDIAN, M. & THEOBALD, S. 2011. Influence of land surface topography on flood hydrograph. *Journal of American Science,* 7.

MILLER, J. D., KIM, H., KJELDSEN, T. R., PACKMAN, J., GREBBY, S. & DEARDEN, R. 2014. Assessing the impact of urbanization on storm runoff in a peri-urban catchment using historical change in impervious cover. *Journal of Hydrology,* 515**,** 59-70.

NIE, W.-B., LI, Y.-B., FEI, L.-J. & MA, X.-Y. 2017. Approximate Explicit Solution to the Green-Ampt Infiltration Model for Estimating Wetting Front Depth. *Water,* 9**,** 609.

RICHARDS, P. J., FARRELL, C., TOM, M., WILLIAMS, N. S. G. & FLETCHER, T. D. 2015. Vegetable raingardens can produce food and reduce stormwater runoff. *Urban Forestry & Urban Greening,* 14**,** 646-654.

SCHOLZ, M. 2006. Wetland systems to control urban runoff. 1st ed. ed. Amsterdam ;: Elsevier.

Spring, 2011, *Green and Ampt Infiltration*, lecture notes, Physical Hydrology for Ecosystems BEE 3710 Cornell University

The Landlab Team, 2013. *Sphinx*, 22/03/2018, <http://landlab-mcflugen.readthedocs.io/en/latest/model_grid.html>

The Landlab Team, 2013. *Sphinx*, 22/03/2018, <http://landlab.readthedocs.io/en/latest/landlab.components.greenampt.html>

WANG, G., YANG, H., WANG, L., XU, Z. & XUE, B. 2014. Using the SWAT model to assess impacts of land use changes on runoff generation in headwaters. *Hydrological Processes,* 28**,** 1032-1042.

ZHANG, L., WANG, J., BAI, Z. & LV, C. 2015. Effects of vegetation on runoff and soil erosion on reclaimed land in an opencast coal-mine dump in a loess area. *CATENA,* 128**,** 44-53.

ZHANG, X., HU, M., GUO, X., YANG, H., ZHANG, Z. & ZHANG, K. 2018. Effects of topographic factors on runoff and soil loss in Southwest China. *Catena,* 160**,** 394-402.

APPENDIX A

CALIBRATION CATCHMENT PROCEDURES AND SCRIPTS

A.1 INTRODUCTION

This Appendix discusses the code used to calibrate the simulation model used in the experiment of the dissertation. The input data received for the calibration catchment area can be located at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/calibrationData

The Python script used in the calibration process is at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/calibration

A.2 TOPOGRAPHY DATA CONVERSION TO ESRII FILE

The calibration area topography was received as a contour map in the shp file format. This file had to be converted to a *RasterModelGrid* ESRI ASCII file for loading into Landlab. The details of the conversion process (done by CJ Willers) is beyond the scope of this study, and is documented in *shape-to-asc.md,* located at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/calibrationData/Reworked

A.3 LANDLAB RUNOFF CALCULATION PYTHON SCRIPT

A Jupyter Notebook *InfiltrationAndStormCalibrationDemExample.ipynb* was used to experiment with the calibration topography and set of storms identified for the calibration process. The final simulation procedure, with no visualisation and plotting, was then used in an iterative batch process to run the model for a number of input parameter combinations (see procedure in the Notebook *InfiltrationAndStormCalibrationDemLoop.ipynb*). Screen dumps of the code, images and graphs from the notebook for one set of example input parameters are presented in this section. The complete code set can be found at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/calibration

Load python modules

```
# system python modules
import sys, os.path

# numpy modules
import numpy as np

# landlab modules
from landlab import RasterModelGrid, CLOSED_BOUNDARY, FIXED_VALUE_BOUNDARY
from landlab.io import read_esri_ascii
from landlab import imshow_grid
from landlab.components import SoilInfiltrationGreenAmpt
from landlab.components import OverlandFlow

# Plotting modules
from matplotlib import pyplot as plt
import matplotlib as mpl

# path with topography dem data
topopath=os.path.join("../../","create-topo")

%matplotlib inline
```

Setup data for the run

Define the storm conditions (mm/hr)

```
storm_flag = '2009-02-10'

if storm_flag == '1995-11-18':
    starting_precip_mmhr = 15.6
    starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)
    storm_duration = 17100.
elif storm_flag == '1997-03-11':
    starting_precip_mmhr = 34.8
    starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)
    storm_duration = 4500.
elif storm_flag == '2009-02-10':
    starting_precip_mmhr = 18.
    starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)
    storm_duration = 7200.
elif storm_flag == '2010-12-08':
    starting_precip_mmhr = 16.8
    starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)
    storm_duration = 5400.
elif storm_flag == '2012-11-24':
    starting_precip_mmhr = 16.8
    starting_precip_ms = starting_precip_mmhr * (2.77778 * 10 ** -7)
    storm_duration = 3900.
```

Setup the run time and logging

```
# Simulation run time and report time as function of storm duration
runHours = 2.5 * storm_duration/3600.
model_run_time = runHours * 3600.
reportInterval = model_run_time / 3.

# graphs and data during run time?
showProgress = True
showPlots = False
```

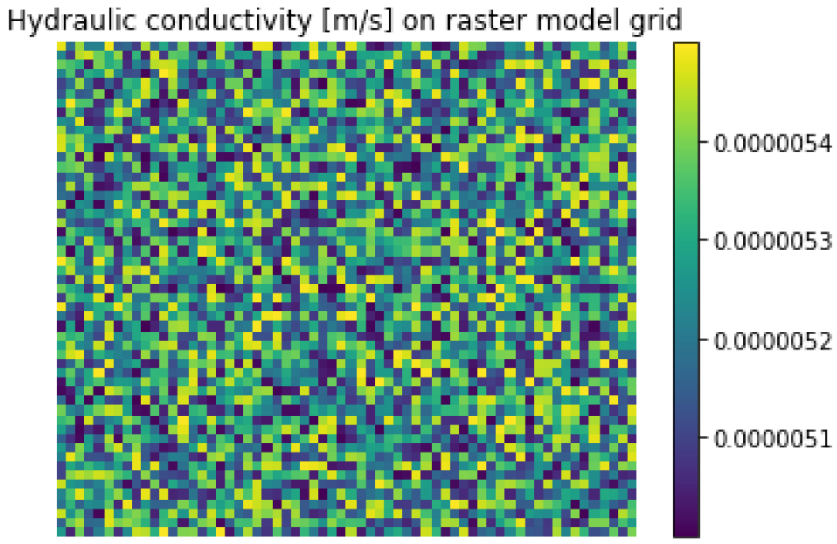
Load topography and create the raster model grid

```
(rmg, z) = read_esri_ascii(os.path.join(topopath,'clipped2cropped2.asc'), name='↵
    topographic__elevation')
```

Initialise variables

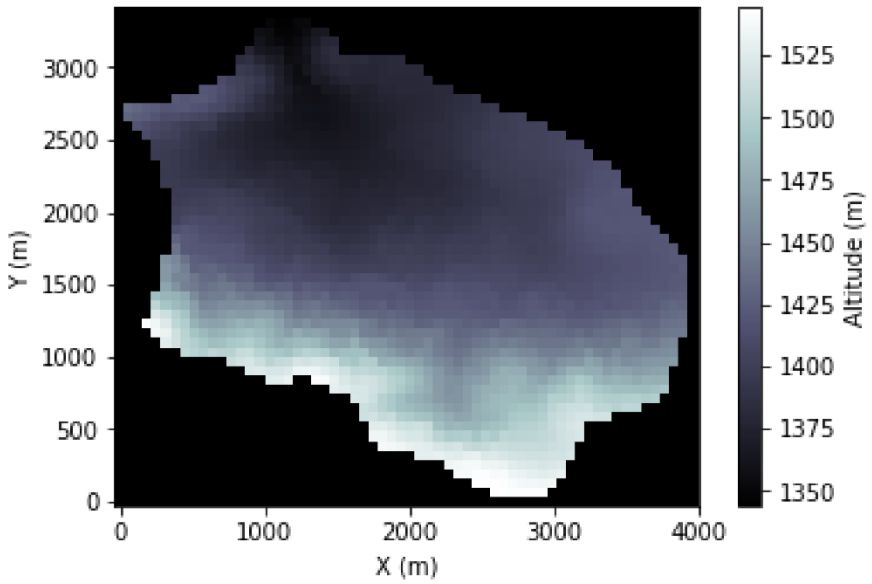
```
# Depth of water above the surface [m]
h = rmg.add_ones('node', 'surface_water__depth')
h *= 0.023

# Water column height above the surface previously absorbed into the soil [m]
# This is NOT the actual depth of the wetted front, which also depends on the porosity
d = rmg.add_ones('node', 'soil_water_infiltration__depth', dtype=float)
d *= 0.15

# Terrain roughness
mannings_n = 0.05

# Soil type and hydraulic conductivity [m/s]
# The ease with which a fluid (usually water) can move through pore spaces or fractures
# Specify range and initialise the field with random values in this range
soil_type = 'sandy loam'
min_h = 5e-6
max_h = 5.5e-6
hydraulic_conductivity = rmg.ones('node')*min_h
hydraulic_conductivity += np.random.rand(z.size)*(max_h - min_h)
```

```
# Visualise the randomness of the hydraulic conductivity
plt.imshow(hydraulic_conductivity.reshape((rmg.shape[0],rmg.shape[1])))
plt.colorbar()
plt.title('Hydraulic conductivity [m/s] on raster model grid')
plt.axis('off')
plt.show()
```

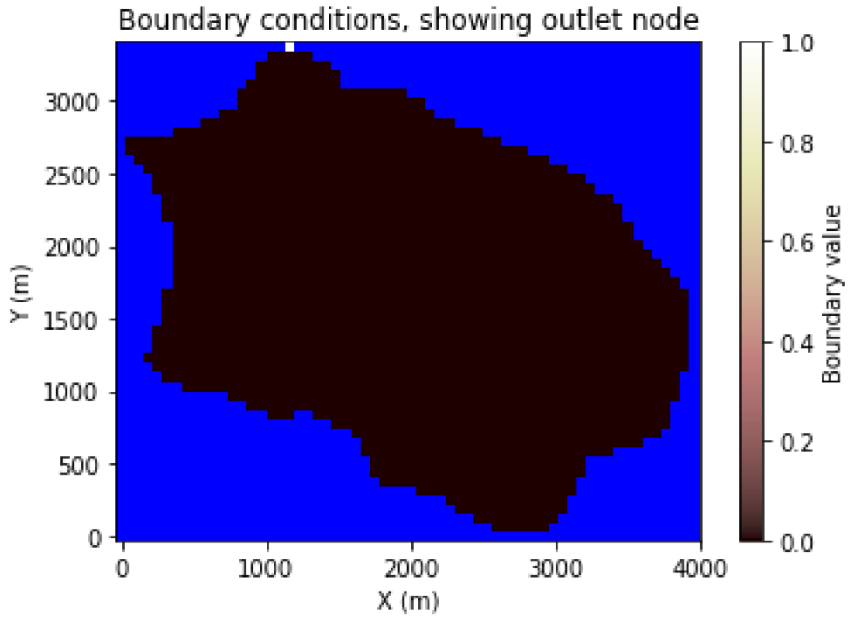


Set boundary conditions and outlet node

```
# Boundary conditions
# The dem has values of -99999 outside the watershed area with only lowest point set as ↲
    open
noDataValue = -99999
rmg.set_watershed_boundary_condition(z, nodata_value=noDataValue)

imshow_grid(rmg, 'topographic__elevation',
            cmap='bone',
#             plot_name='Watershed topography',
            grid_units=['m','m'],var_name='Altitude',var_units='m')
# plt.savefig('topography.eps')
plt.savefig('topography.png')
```



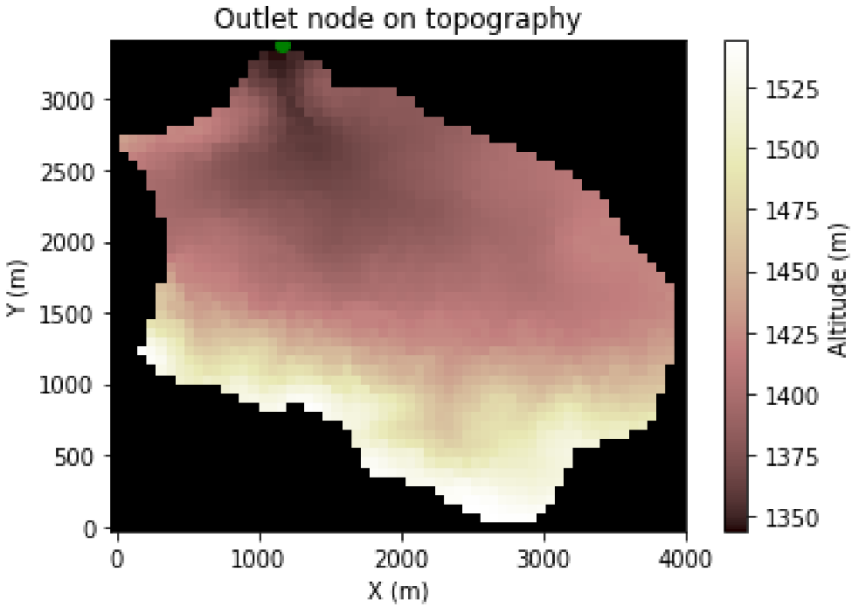

```
# Visualise the boundary conditions and the outlet point
imshow_grid(rmg, rmg.status_at_node,color_for_closed='blue',plot_name='Boundary ↵
    conditions, showing outlet node',
            grid_units=['m','m'],var_name='Boundary value')
```




```
# Set the outlet node value
outlet_node = np.where(rmg.status_at_node==1)
print('The outlet node is at {}'.format(outlet_node[0][0]))

# Get information about the links at the outlet node
linksAtNode = rmg.links_at_node[outlet_node][0]
print('The links at the outlet node are: {}'.format(linksAtNode))

# Status of links at the outlet node:
# 0: active
```

```
# 2: fixed
# 4: inactive
linkStatus = rmg.status_at_link[rmg.links_at_node[outlet_node]][0]
print('The status at the links are: {}'.format(linkStatus))

# Link flux directions at each node:
#  1: incoming flux
# -1: outgoing flux
#  0: no flux
fluxAtLink = rmg.active_link_dirs_at_node[outlet_node][0]
print('The flux directions at the links are: {}'.format(fluxAtLink))

# set the link to the avtive one
outlet_link = linksAtNode[np.where(linkStatus == 0)[0] and np.where(fluxAtLink != 0)↵
    [0]][0]
print('Outlet link to monitor is set to {}'.format(outlet_link))

# Visualise the outlet node
imshow_grid(rmg, z, plot_name='Outlet node on topography',
            grid_units=['m','m'],var_name='Altitude',var_units='m')
plt.plot(rmg.node_x[outlet_node],rmg.node_y[outlet_node],'go')
plt.show()
```

```
The outlet node is at 3242
The links at the outlet node are: [6414   -1 6413 6352]
The status at the links are: [4 4 4 0]
The flux directions at the links are: [0 0 0 1]
Outlet link to monitor is set to 6352
```



Set up the components

```
# Set up the OverlandFlow component
of = OverlandFlow(rmg, mannings_n = mannings_n, steep_slopes = True)

# Set up the SoilInfiltrationGreenAmpt component
SI = SoilInfiltrationGreenAmpt(rmg, soil_type = soil_type, hydraulic_conductivity=↵
    hydraulic_conductivity)
```

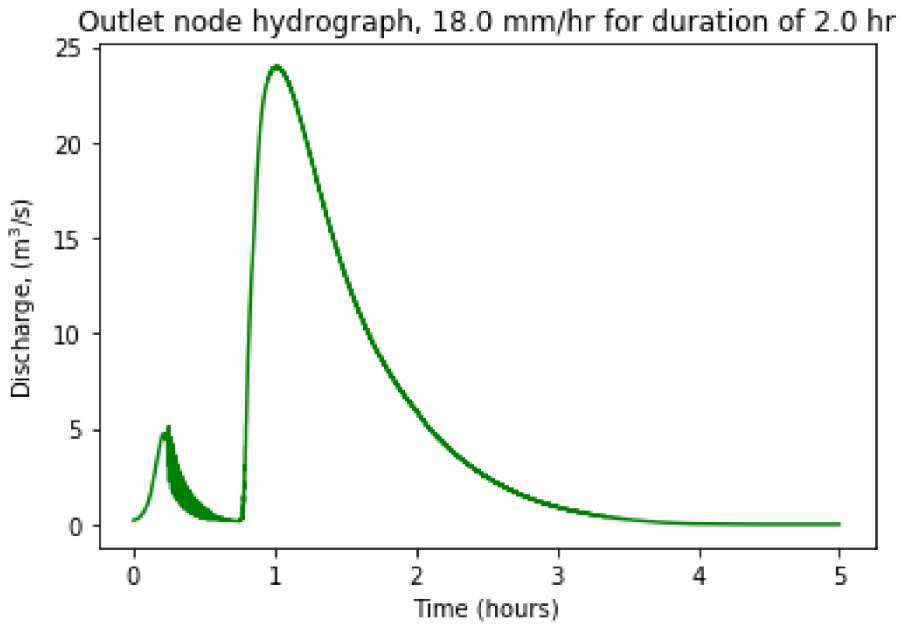
Run simulation

```
# Simulation loop
elapsed_time = 0.
next_report = 0.
fig = 1

# Lists for saving data
discharge_at_outlet = []
hydrograph_time = []

# Simulation Loop
while elapsed_time < model_run_time:

    # Setting the adaptive time step
    of.dt = of.calc_time_step()

    # The storm starts when the model starts
    # While the elapsed time is less than the storm duration, add water to the system as rainfall
    if elapsed_time < (storm_duration):
        of.rainfall_intensity =  starting_precip_ms

    # elapsed time exceeds the storm duration, rainfall ceases
    else:
        of.rainfall_intensity = 0.0

    # Generating overland flow component
    of.overland_flow()

    # Append time and discharge to lists to save data for plotting
    hydrograph_time.append(elapsed_time/3600.)                    # hours
    discharge_at_outlet.append(np.abs(of.q[outlet_link]) * rmg.dx) # m^3/s

    # Run soil infiltration component
    SI.run_one_step(of.dt)

    # Report if requested
    if elapsed_time >= next_report:
        if showPlots:
            plt.figure(fig)
            fig = fig + 1
            imshow_grid(rmg, 'soil_water_infiltration__depth', cmap='Blues',
                        grid_units=['m','m'],var_name='Soil water infiltration depth',
                            var_units='m')
            plt.title('Elapsed time {:.2f} hours'.format(elapsed_time/3600.))

        elif showProgress:
            print('elapsed time = {:.2f} s [{:.2f} hours]'.format(elapsed_time,
                elapsed_time/3600.))

        next_report += reportInterval

    # Update elapsed_time
    elapsed_time += of.dt
```

```
elapsed time = 0.00 s [0.00 hours]
elapsed time = 6006.99 s [1.67 hours]
elapsed time = 12006.75 s [3.34 hours]
```

Simulation output

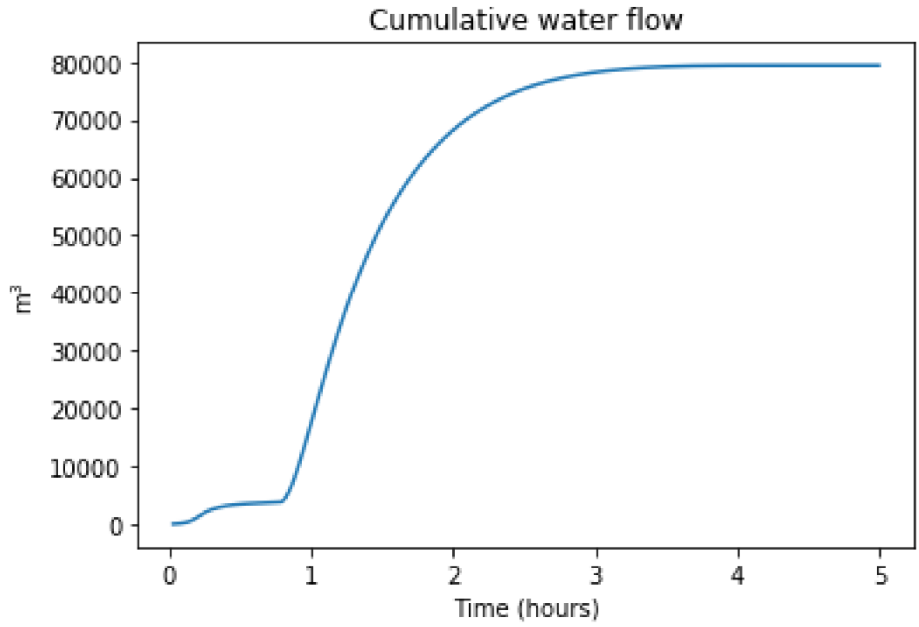
```
# Hydrograph at the outlet node
plt.plot(hydrograph_time, discharge_at_outlet, 'g')
plt.xlabel('Time (hours)')
plt.ylabel('Discharge, (m$^3$/s)')
plt.title('Outlet node hydrograph, {} mm/hr for duration of {} hr'.format(
    starting_precip_mmhr, storm_duration/3600.))
plt.show()
print('Maximum discharge at the outlet node = {:.3f} m3/s'.format(max(
    discharge_at_outlet)))
```




```
Maximum discharge at the outlet node = 23.949 m3/s
```

```
# Cumulative water flow at outlet node
tot_volume_mid = np.cumsum(np.array(discharge_at_outlet)[1:]*np.diff(np.array(↵
    hydrograph_time)))*3600.

plt.plot(hydrograph_time[1:], tot_volume_mid)
plt.title('Cumulative water flow')
plt.ylabel('m$^3$')
plt.xlabel('Time (hours)')
print('Cumulative discharge at the outlet node = {:.3f} m3/s'.format(sum(np.abs(↵
    discharge_at_outlet)[1:]*np.diff(np.array(hydrograph_time))*3600.)))
```

```
Cumulative discharge at the outlet node = 79399.297 m3/s
```



APPENDIX B

EXPERIMENT PROCEDURES AND SCRIPTS

B.1 INTRODUCTION

This Appendix discusses the Python scripts used in the execution of the simulation experiments.

B.2 TOPOGRAPHY GENERATION

The Python code used to generate the valley topographies used in the simulation experiments was developed by CJ Willers and is located at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/topographies

The Jupyter Notebook *Create-simple-topo.ipynb* is presented in this Appendix. In initial experimentation for the dissertation, the simple flat tilted topography was used. This type of topography has an open boundary along one side of the terrain and did not produce hydrographs matching real-world behaviour. In order to simulate more realistic real-world scenarios, the valley topography was used in the final experiment. These valley topographies are also simplified terrains in the sense that the two slopes that form the valley are flat tilted surfaces. In order to simplify the experimental problem and measure water runoff as a function of slope, these simplified valley terrain were preferred above terrain topographies with variation in height on the slopes, e.g. the fractal topography (last example in this notebook). The rest of this section is a verbatim inclusion of the Jupyter Notebook.

Create simple topograpy maps for LandLab

This notebook provides a tool to create three types of terrain topography files for experimentation in LandLab.

The functions included here allows the user to calculate different terrains, based on the different input parameters to the generating functions. The different types are:

1. A tilted plane. The plane can be tilted in any direction by any magnitude.
2. A valley defined by two tilted planes
3. A simple fractal terrain.

```
import numpy as np
import random as rnd
import matplotlib
import matplotlib.cm as cm
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.ndimage.filters

matplotlib.rcParams['xtick.direction'] = 'out'
matplotlib.rcParams['ytick.direction'] = 'out'

%matplotlib inline
```

Landlab documentation

1. https://github.com/landlab/landlab/wiki
2. https://github.com/landlab/landlab/wiki/Grid
3. https://github.com/landlab/landlab/wiki/User-Guide
4. https://media.readthedocs.org/pdf/landlab/latest/landlab.pdf

ASC file format

The Esri ASCII raster format can be used to transfer information to or from other cell-based or raster systems. When an existing raster is output to an Esri ASCII-format raster, the file will begin with header information that defines the properties of the raster such as the cell size, the number of rows and columns, and the coordinates of the origin of the raster. The header information is followed by cell value information specified in space-delimited row-major order, with each row separated by a carriage return.

To convert an ASCII file to a raster, the data must be in this same format. The parameters in the header part of the file must match correctly with the structure of the data values.

The basic structure of the Esri ASCII raster has the header information at the beginning of the file followed by the cell value data. The spatial location of the raster is specified by the location of the lower left cell, and either by:

The center of the lower left cell

```
NCOLS xxx
NROWS xxx
XLLCENTER xxx
YLLCENTER xxx
CELLSIZE xxx
NODATA\_VALUE xxx
row 1
row 2
...
row n
```

The lower left corner of the lower left cell

```
NCOLS xxx
NROWS xxx
XLLCORNER xxx
YLLCORNER xxx
CELLSIZE xxx
NODATA\_VALUE xxx
row 1
row 2
...
row n
```

Row 1 of the data is at the top of the raster, row 2 is just under row 1, and so on.

Header format

The syntax of the header information is a keyword paired with the value of that keyword. These are the definitions of the keywords:

Parameter	Description	Requirements
NCOLS	Number of cell columns	Integer greater than 0.
NROWS	Number of cell rows	Integer greater than 0.
XLLCENTER or XLLCORNER	X-coordinate of the origin	Match with y-coordinate type.
	(by center or lower left corner of the cell)	
YLLCENTER or YLLCORNER	Y-coordinate of the origin	Match with x-coordinate type.
	(by center or lower left corner of the cell)	
CELLSIZE	Cell size	Greater than 0.
NODATA_VALUE	The input values to be NoData in the output raster	Optional. Default is -9999.

Data format

The data component of the Esri ASCII raster follows the header information.

1. Cell values should be delimited by spaces.
2. No carriage returns are necessary at the end of each row in the raster. The number of columns in the header determines when a new row begins.
3. Row 1 of the data is at the top of the raster, row 2 is just under row 1, and so on.

Example ASCII raster

```
ncols 480
nrows 450
xllcorner 378923
yllcorner 4072345
cellsize 30
nodata\_value -32768
43 2 45 7 3 56 2 5 23 65 34 6 32 54 57 34 2 2 54 6
35 45 65 34 2 6 78 4 2 6 89 3 2 7 45 23 5 8 4 1 62 ...
```

References

1. http://resources.esri.com/help/9.3/arcgisdesktop/com/gp_toolref/spatial_analyst_tools/esri_ascii_raster_format.htm
2. https://resources.arcgis.com/en/help/main/10.1/index.html
3. https://gis.stackexchange.com/questions/71867/understanding-esris-asc-file

The ASC file is an ASCII file for raster GIS data. It is not necessarily only for DEM data, presumably any raster data can be stored in this format?

LandLab ASC functions

https://landlab.readthedocs.io/en/release/landlab.io.esri_ascii.html

ESRI ASCII functions (link above has example python code).

1. read_asc_header(asc_file) Read header information from an ESRI ASCII raster file.
2. read_esri_ascii(asc_file[, grid, reshape, ...]) Read RasterModelGrid from an ESRI ASCII file.
3. write_esri_ascii(path, fields[, names, clobber]) Write landlab fields to ESRI ASCII.

Writing ASC DEM files

`writeASCDEM` takes a two-dimensional grid (numpy array) and the required parameters to write the grid information to an ASC file. The function parameters are explained in the file docstring.

```
def writeASCDEM(filename, grid, cellsize, xllcorner=0., yllcorner=0., nodata_value=0):
    """ Writes a pre-calculated grid to an ESRI ASC format

    Args:
        | fileName (string): output file name with full path
        | grid (np.array): two-dimensional numpy array with grid values
        | cellsize (int or float): size of one cell in the grid (assumed isotropic)
        | xllcorner (int or float): x coordinate of the lower left corner
        | yllcorner (int or float): y coordinate of the lower left corner
        | nodata_value (int or float): data to be used if none is present in ASC file

    Returns:
        | Nothing, as side effect the file is written to disk

    Raises:
        | No exception is raised.
    """

    # get number of rows and cols from the grid itself
    ncols = grid.shape[1]
    nrows = grid.shape[0]

    with open(filename,'wb') as fout:
        fout.write('ncols {}\n'.format(ncols).encode()) # to give byte string
        fout.write('nrows {}\n'.format(nrows).encode())
        fout.write('xllcorner {}\n'.format(xllcorner).encode())
        fout.write('yllcorner {}\n'.format(yllcorner).encode())
        fout.write('cellsize {}\n'.format(cellsize).encode())
        fout.write('nodata_value {}\n'.format(nodata_value).encode())
        np.savetxt(fout, grid.astype(float), fmt='%.6e', delimiter=' ', newline='\n')
```

The following example writes a random field out as an ASC file. This is not a valid topography, it only serves to demonstrate writing the file.

```
grid = np.random.rand(4,6)
writeASCDEM('random.asc', grid, cellsize=30)
```

Flat, tilted plane

The topography is modelled by a plane defined by the normal vector to the plane and the altitude of the lower left corner.

The general equation for a plane is given by $Ax + By + Cz + D = 0$ where A, B,and C are the x, y, z components of a vector normal to the plane. It normal vector need not be normalised to unity value. If the plane passes through the origin, $D = 0$, then $z = (Ax + By)/C$. If the plane is perfectly level $(A, B, C) = (0, 0, 1)$.

```
def createPlane(ncols,nrows,tilt=10.,azim=0.,cellsize=1):
    """ Creates a flat plane in prescribed tilt and azimuth angles

    The plane is described by the direction of its normal vector.
    The normal vector tilts from the vertical, pointing along the azimuth direction.
    Azimuth of zero or 360 deg points towards +x or the East.
    Tilt of zero points vertically up, plan is horisontal.
```

```
    The number of rows/cols and cell size are used to calculate x/y coordinates.

    The plane passes through the origin (0,0,0), the user must lift or lower as ↵
        necessary.

    Args:
        | ncols (int): number of grid columns (along x)
        | nrows (int): number of grid rows (along y)
        | tilt (float): tilt angle [degrees] of the plane normal vector
        | azim (float): azimuth angle [degrees] of the plane normal vector
        | cellsize (int or float):  size of one cell in the grid (assumed isotropic)

    Returns:
        | x (np.array): two-dimensional array, values varying in x direction
        | y (np.array): two-dimensional array, values varying in y direction
        | grid (np.array): two-dimensional array, plane values

    Raises:
        | No exception is raised.
    """

    C = np.cos(tilt*np.pi/180.)
    zd =  - np.abs(np.sin(tilt*np.pi/180.))
    A = zd * np.cos(azim*np.pi/180.)
    B = zd * np.sin(azim*np.pi/180.)

    varx = np.linspace(-cellsize * ncols/2., cellsize * ncols/2., ncols)
    vary = np.linspace(-cellsize * nrows/2., cellsize * nrows/2., nrows)
    x, y = np.meshgrid(varx, vary)

    grid = (A * x + B * y  ) / C

    return x,y,grid
```

```
def plotTopo(x,y,grid,filename):
    """ Plot the topography, given x,y, and grid values.

    An image graph and contour plot are created.
    If the topography is horisontal, the contour plot is not drawn.

    Args:
        | x (np.array): two-dimensional array, values varying in x direction
        | y (np.array): two-dimensional array, values varying in y direction
        | grid (np.array): two-dimensional array, plane values
        | filename (string): filename used in the plot header

    Returns:
        | Nothing

    Raises:
        | No exception is raised.
    """
    plt.figure(num=None, figsize=(12,5),dpi=150)
    plt.subplot(1,2,1)
    im = plt.imshow(grid, interpolation='bilinear', origin='lower',cmap=cm.gray)
    CBI = plt.colorbar(im,  shrink=0.8)
    plt.title('Cell image: {}'.format(filename))

    if np.unique(grid).shape[0] > 1:
        plt.subplot(1,2,2)
        levels = np.linspace(np.min(grid),np.max(grid),10)
        CS = plt.contour(x,y,grid, levels, origin='lower', linewidths=2,colors='k')#,↵
            extent=(-3, 3, -2, 2))
#         plt.clabel(CS, levels,inline=1,fmt='%1.1f',fontsize=14)
        plt.axis('equal')
        plt.xlabel('x [m]')
        plt.ylabel('y [m]')
        plt.title('Altitude: {}'.format(filename))
        plt.grid(True)
    else:
        print('Topography is horizontal with altitude {} m'.format(np.unique(grid)[0]))
```

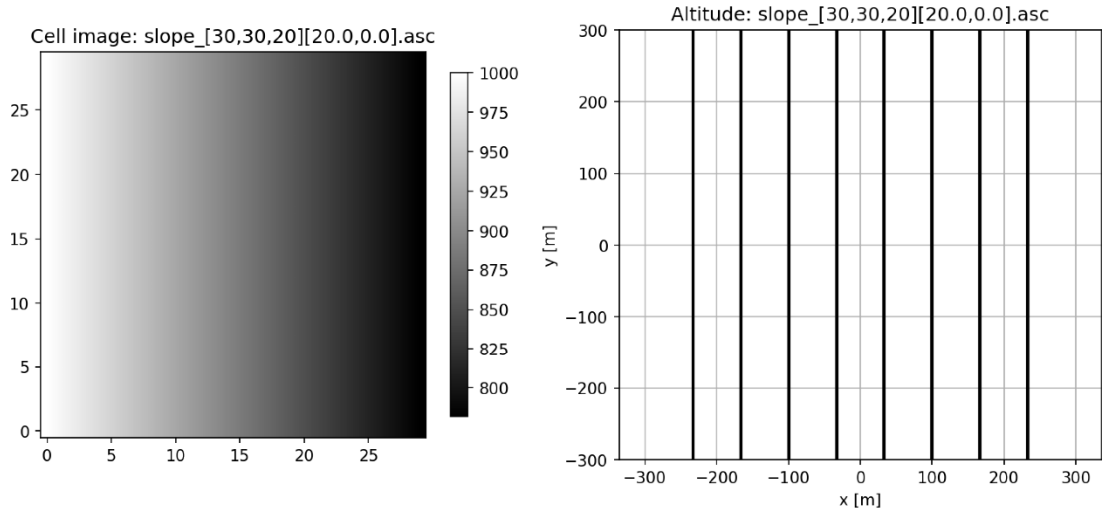
Flat tilted topography

The topography created by this function is a perfectly flat tilted surface. It is therefore the most simple watershed possible, with a guaranteed exit (all water will leave the terrain).

```
def createFlatTopo(filecore,ncols,nrows,tilt=10.,azim=0.,cellsize=1,llcorner=[0.,0.,0.]):
    """ Creates a plane or flat topography, tilted along azimuth angle

    The topography is described by the direction of its normal vector.
    The normal vector tilts from the vertical, pointing along the azimuth direction.
    Azimuth of zero or 360 deg points towards +x or the East.
    Tilt of zero points vertically up, plan is horisontal.

    The number of rows/cols and cell size are used to calculate x/y coordinates.

    The plane lower left corner is set to the values in llcorner.


    Args:
        | filecore (string): first part of the filename
        | ncols (int): number of grid columns (along x)
        | nrows (int): number of grid rows (along y)
        | tilt (float): tilt angle [degrees] of the plane normal vector
        | azim (float): azimuth angle [degrees] of the plane normal vector
        | cellsize (int or float):  size of one cell in the grid (assumed isotropic)
        | llcorner ([int or float]): [x,y,z] coordinates of the lower left corner

    Returns:
        | x (np.array): two-dimensional array, values varying in x direction
        | y (np.array): two-dimensional array, values varying in y direction
        | grid (np.array): two-dimensional array, plane values
        | filename (string): filename used in the plot header
        | as a side effect, the file is also written to disk

    Raises:
        | No exception is raised.
    """

    x,y,grid = createPlane(ncols,nrows,tilt,azim,cellsize)
    gridll = grid[0,0]
    grid = grid - gridll + llcorner[2]

    filename = '{}_[{},{},{}][{},{}].asc'.format(filecore,ncols,nrows,cellsize,tilt,azim)
    print('Writing file {}'.format(filename))
    writeASCDEM(filename,grid,cellsize,xllcorner=llcorner[0], yllcorner=llcorner[1])

    return x,y,grid,filename
```

```
x,y,grid,filename = createFlatTopo('slope',ncols=30,nrows=30,tilt=20.,azim=0.0,cellsize=20,llcorner=[0.,0.,1000.])
plotTopo(x,y,grid,filename)
```

```
Writing file slope_[30,30,20][20.0,0.0].asc
```



Flat valley topography

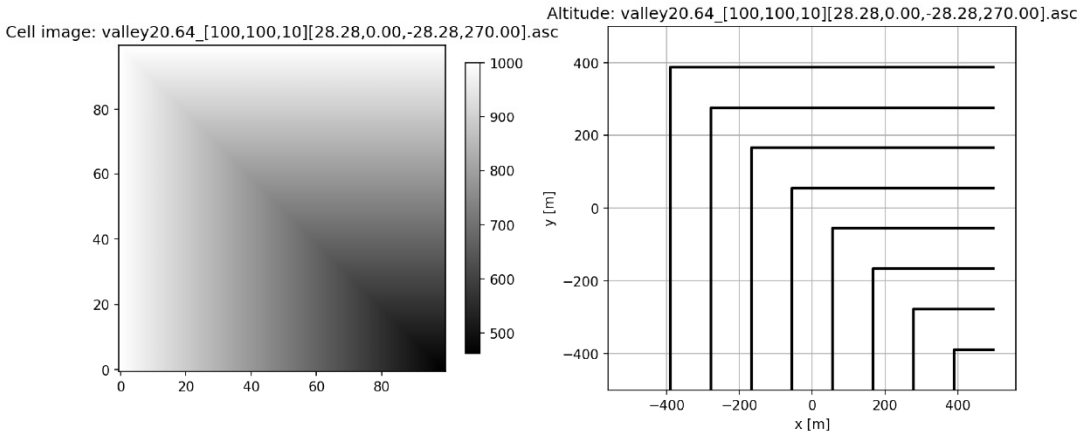
The topography created by this function is a valley formed by two perfectly flat tilted surfaces. There is one watershed, the valley with a guaranteed exit (all water will leave the terrain).

```
def createValleyTopo(filecore,ncols,nrows,tilt1=0.,azim1=0.,tilt2=0.,azim2=0.,cellsize↵
    =1,llcorner=[0.,0.,0.]):
    """ Creates a valley topography defined by two tilted planes

    The topography is described by the directions of its two normal vectors.
    The normal vector tilts from the vertical, pointing along the azimuth direction.
    Azimuth of zero or 360 deg points towards +x or the East.
    Tilt of zero points vertically up, plan is horisontal.

    The topography is taken to be the highest of the two planes

    The number of rows/cols and cell size are used to calculate x/y coordinates.

    The plane lower left corner is set to the values in llcorner.


    Args:
        | filecore (string): first part of the filename
        | ncols (int): number of grid columns (along x)
        | nrows (int): number of grid rows (along y)
        | tilt1 (float): tilt angle [degrees] of the first plane normal vector
        | azim1 (float): azimuth angle [degrees] of the first plane normal vector
        | tilt2 (float): tilt angle [degrees] of the second plane normal vector
        | azim2 (float): azimuth angle [degrees] of the second plane normal vector
        | cellsize (int or float):  size of one cell in the grid (assumed isotropic)
        | llcorner ([int or float]): [x,y,z] coordinates of the lower left corner

    Returns:
        | x (np.array): two-dimensional array, values varying in x direction
        | y (np.array): two-dimensional array, values varying in y direction
        | grid (np.array): two-dimensional array, plane values
        | filename (string): filename used in the plot header
        | as a side effect, the file is also written to disk

    Raises:
        | No exception is raised.
    """
    x1,y1,grid1 = createPlane(ncols,nrows,tilt1,azim1,cellsize)
    x2,y2,grid2 = createPlane(ncols,nrows,tilt2,azim2,cellsize)
    grid = np.maximum(grid1,grid2)

    gridll = grid[0,0]
    grid = grid - gridll + llcorner[2]

    row2 = int(nrows/2.)
    col2 = int(ncols/2.)
```

```
    r = np.sqrt((cellsize*row2)*(cellsize*row2) + (cellsize*col2)*(cellsize*col2))
    flatslope = np.arctan((grid[0,0]-grid[row2-1,col2-1])/(r))*180/np.pi
    valleyslope = np.arctan((grid[nrows-1,0]-grid[row2-1,col2-1])/(r))*180/np.pi

    print('flatslope={:.3f} deg'.format(flatslope))
    print('valleyslope={:.3f} deg'.format(valleyslope))


    filename = '{}{:.2f}_[{},{},{}][{:.2f},{:.2f},{:.2f},{:.2f}].asc'.format(filecore,↵
        flatslope,ncols,nrows,cellsize,tilt1,azim1,tilt2,azim2)
    print('Writing file {}'.format(filename))
    writeASCDEM(filename,grid,cellsize,xllcorner=llcorner[0], yllcorner=llcorner[1])

    return x1,y1,grid,filename
```

```
down = 20*np.sqrt(2)
valg = 45
x,y,grid,filename = createValleyTopo('valley',ncols=100,nrows=100,
                                     tilt1=down,azim1=45.-valg,tilt2=-down,azim2↵
                                         =180+45.+valg,
                                     cellsize=10,llcorner=[0.,0.,1000.])

plotTopo(x,y,grid,filename)
```

```
flatslope=20.639 deg
valleyslope=20.639 deg
Writing file valley20.64_[100,100,10][28.28,0.00,-28.28,270.00].asc
```



Fractal terrain topography

The fractal topography generator is taken from here:

https://github.com/dafarry/python-fractal-landscape/blob/master/fractal-mountain.py

There is an error in this code, because the terrains have some correlation between x and y, the terrains are not fully random. It is however a short code and the results may be useful, even if all mathematical requirements are not met in the implementation.

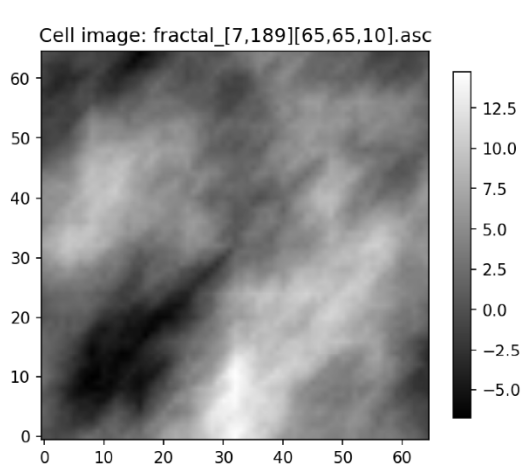
The topography created by this function is a random fractal surface. It is a complex watershed model, without a guaranteed exit (not all water will leave the terrain, some water will be captured in local lakes).

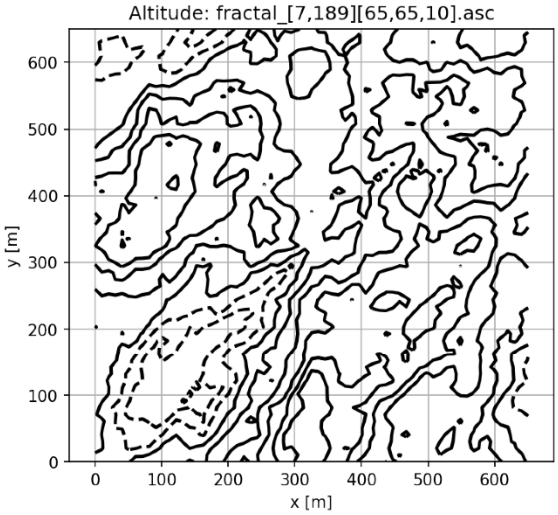
```
def createFractalTopo(filecore,levels=5,seed=10,cellsize=1,llcorner=[0.,0.,0.]):
    """ Creates a fractal topography

    https://github.com/dafarry/python-fractal-landscape/blob/master/fractal-mountain.py

    The terrain size is 2**(levels-1)
```

```
    Use different seed values to create different terrains.

    The plane lower left corner is set to the values in llcorner.


    Args:
        | filecore (string): first part of the filename
        | levels (int): recursion depth, determines the terrain size
        | seed (int): the number used to seed the random number generator
        | cellsize (int or float):  size of one cell in the grid (assumed isotropic)
        | llcorner ([int or float]): [x,y,z] coordinates of the lower left corner

    Returns:
        | x (np.array): two-dimensional array, values varying in x direction
        | y (np.array): two-dimensional array, values varying in y direction
        | grid (np.array): two-dimensional array, plane values
        | filename (string): filename used in the plot header
        | as a side effect, the file is also written to disk

    Raises:
        | No exception is raised.
    """

    size = 2 ** (levels - 1)
    grid  = np.zeros((size + 1, size + 1))
    rnd.seed(seed)
    for lev in range(levels):
        step = size // 2 ** lev
        for y in range(0, size + 1, step):
            jumpover = 1 - (y // step) % 2 if lev > 0 else 0
            for x in range(step * jumpover, size + 1, step * (1 + jumpover)):
                pointer = 1 - (x // step) % 2 + 2 * jumpover if lev > 0 else 3
                yref, xref = step * (1 - pointer // 2), step * (1 - pointer % 2)
                corner1 = grid[y - yref, x - xref]
                corner2 = grid[y + yref, x + xref]
                average = (corner1 + corner2) / 2.0
                variation = step * (rnd.random() - 0.5)
                grid[y,x] = average + variation if lev > 0 else 0

    grid = grid - grid[0,0] + llcorner[2]

    varx = np.linspace(0, cellsize * grid.shape[1], grid.shape[1])
    vary = np.linspace(0, cellsize * grid.shape[0], grid.shape[0])
    xx, yy = np.meshgrid(varx, vary)

    ncols = grid.shape[1]
    nrows = grid.shape[0]
    filename = '{}_[{},{}][{},{},{}].asc'.format(filecore,levels,seed,ncols,nrows,↵
        cellsize)
    print('Writing file {}'.format(filename))
    writeASCDEM(filename,grid,cellsize,xllcorner=llcorner[0], yllcorner=llcorner[1])

    return xx,yy,grid,filename
```

```
x,y,grid,filename = createFractalTopo('fractal',levels=7,seed=189,cellsize=10,llcorner↵
    =[0.,0.,0.])
plotTopo(x,y,grid,filename)
```

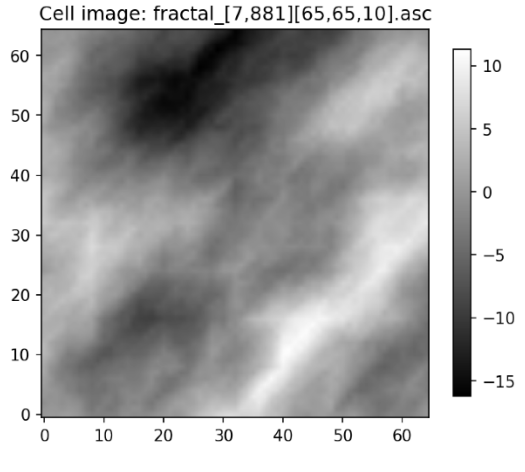
```
Writing file fractal_[7,189][65,65,10].asc
```

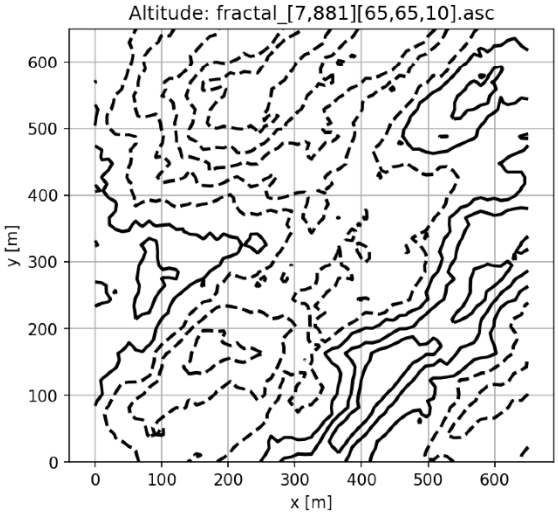





```
x,y,grid,filename = createFractalTopo('fractal',levels=7,seed=881,cellsize=10,llcorner
    =[0.,0.,0.])
plotTopo(x,y,grid,filename)
```

```
Writing file fractal_[7,881][65,65,10].asc
```





B.3 RAINFALL RUNOFF SIMULATION CODE

The final set of Python code used to execute the 300 experiments and to visualise the output can be found at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/experimental

The same Landlab simulation procedure used for the calibration process was used in the execution of the simulation experiments. The code common to running the Landlab model for each set of input parameters was identified and encapsulated in a utility Python function *runModel*. Utility functions were developed for writing and reading the relevant output from a simulation run to and from a file on disk. These utility functions can be found in *myLandlabTools.py* at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/util

The Jupyter Notebook *calcFlowLoop.ipynb* (screen dumps presented below) simply sets up the ranges of the free variables, iterates over all the combinations and write the simulation output data to disk for analysis.

```
import sys
import os.path
import numpy as np
from landlab import imshow_grid
from landlab import RasterModelGrid
from matplotlib import pyplot as plt

#use the utility functions
utilspath=os.path.join("../../","util")
sys.path = [utilspath]+sys.path
import myLandlabTools as tools

topopath=os.path.join("../../","create-topo")

%matplotlib inline
```

```
demDict = {
    'valley2' : ['valley1.98_[100,100,10][2.83,0.00,-2.83,270.00].asc'],
    'valley5' : ['valley4.96_[100,100,10][7.07,0.00,-7.07,270.00].asc'],
    'valley10' : ['valley10.00_[100,100,10][14.14,0.00,-14.14,270.00].asc'],
    'valley15' : ['valley15.20_[100,100,10][21.21,0.00,-21.21,270.00].asc'],
    'valley20' : ['valley20.64_[100,100,10][28.28,0.00,-28.28,270.00].asc'],
    'valley25' : ['valley26.41_[100,100,10][35.36,0.00,-35.36,270.00].asc'],
 }

# soil infiltration
# from Gowdish and Munoz & Carsel and Parrish
hydCondDict = {
            'cond1' :[1.7e-7, 'clay'],
            'cond2' :[2.7e-7, 'silty clay'],
            'cond3' :[3.3e-7, 'sandy clay'],
            'cond4' :[5.5e-7, 'silty clay loam'],
            'cond5' :[7.2e-7, 'clay loam'],
            'cond6' :[8.3e-7, 'sandy clay loam'],
            'cond7' :[1.8e-6, 'silt loam' ],
            'cond8' :[3.6e-6, 'loam'],
            'cond9' :[6.1e-6, 'sandy loam' ],
            'cond10':[1.6e-5, 'loamy sand'],
}


roughnessDict = {
    'clean': [0.03],
    'mediummix': [0.05],
    'overgrown': [0.07],
}

# Storm [starting_precip_mmhr, storm duration s]
#
# A constant precipitation rate can be passed to the OverlandFlow class, where ↵
    precipitation persists for the
# entire model run. Alternatively, a single event can be set within the time loop, and ↵
    then water can drain
# from the system when the precipitation event is over.
stormDict = {
    'high':[30., 17100.],
    'low':[15., 17100.],
    'intense': [50., 900.],
    'intense1': [50., 3600.],
    'intense2': [100., 900.],
    '1995-11-18':[15.6, 17100.],
    '1997-03-11':[34.8, 4500.],
    '2009-02-10':[18.0, 7200.],
    '2010-12-08':[16.8, 5400.],
    '2012-11-24':[16.8, 3900],
}
```

```
# set simulation parameters - use the dictionaries in the previous block to set up the ↵
    scenario

# storm
storm_flag = 'intense1'

# Water column height above the surface previously absorbed into the soil.
# Do NOT set to zero
soilWaterInfiltrationDepth = 0.1
```

```
# Depth of water above the surface - a value close to 0 is recommended here
surfaceWaterDepth = 1e-8

# simulation run time [hours]
runHours = 3.
# interval to report progress [hours]
reportInterval = 1.

# Storm
starting_precip_mmhr = stormDict[storm_flag][0]
storm_duration = stormDict[storm_flag][1]

# Manning n
deltaM = 0.01
manningRange = np.arange(roughnessDict['clean'][0],roughnessDict['overgrown'][0]+deltaM,↵
     deltaM)

# Hydraulic conductivity
hydcondRange = []
soilType = []
for key in hydCondDict:
    hydcondRange.append(hydCondDict[key][0])
    soilType.append(hydCondDict[key][1])

soilInf = True
```

```
# Loop over all combinations of input parameters

# For each dem in the list
for key in demDict:
    dem = demDict[key][0]
    print(dem)

    # set boundary and monitor link

    if 'gully' in key:
        boundary = 'noData'
        outletNode = 296
        monitorLink = 590
    elif 'valley' in key:
        boundary = 'setSelf'
        outletNode = 9898
        monitorLink = 19500      # links for node 9898 : 19600 19699 19599 19500
    else:
        if 'flat' in key:
            boundary = 'allOpen'
            outletNode = 465
            monitorLink = 870      # links for node 465 : 900 929 899 870
        else:
            boundary = 'rightOpen'
            outletNode = 449
            monitorLink = 854

    # for each hydraulic conductivity
    for min_h, soil_type in zip(hydcondRange,soilType):

        # for each roughness value
        for mannings_n in manningRange:

            rmg, hydraulic_conductivity, discharge_at_outlet, hydrograph_time, h, d = ↵
                tools.runModel(
                                    topopath, dem,
                                    soil_type, min_h, min_h,
                                    starting_precip_mmhr, storm_duration,
                                    soilWaterInfiltrationDepth, surfaceWaterDepth,
                                    mannings_n,
                                    boundary, monitorLink,
                                    runHours, reportInterval,
                                    soilInf = soilInf, sinkFiller = False,
                                    showProgress = False, showPlots = False,
                                    outlet_node = outletNode)

            fileName = tools.getLLFilename(key,mannings_n,min_h,↵
                soilWaterInfiltrationDepth)
            tools.writeLLData(fileName,hydrograph_time,discharge_at_outlet,h,d)
```

B.4 DATA ANALYSIS CODE

The Python script *doAllPlots.py* was used to plot hydrographs, maximum discharge and cumulative discharge for the ranges of two of the input parameters, keeping one parameter fixed. Heat maps visualising the maximum flow and sum of the discharge were generated using the *plotHeatMaps.ipynb* notebook. The final code can be found at URL:

https://github.com/ismari92/LandlabSurfaceRunOffModel/tree/master/experimental

APPENDIX C

EVALUATION FORM

UNIVERSITY OF PRETORIA
DEPARTMENT OF CIVIL ENGINEERING
MARKING SHEET FOR UNDERGRADUATE RESEARCH PROJECT REPORTS

Student:

Minimum requirements – the project report will be referred back if it does not meet the following requirements:

The student identified, assessed, formulated and solved a problem (ECSA ELO1)		The report is of a professional quality and appearance (ECSA ELO6)	
The student applied fundamental knowledge to solve an engineering problem (ECSA ELO2)		The student is aware of the impact of engineering activities on the environment and the community (ECSA ELO7)	
The student solved the problem systematically (ECSA ELO3)		The student worked independently to submit a unique research report (ECSA ELO9)	
The student analysed, interpreted and derived information from data (ECSA ELO4)		Project is largely the students own work. Student should clearly indicate his/her own work. (ECSA ELO10)	
The student used appropriate methods and computer technology to solve the problem (ECSA ELO5)		The report is submitted by the specified date. (ECSA ELO10)	

❑

❑ **Marks**

❑ 30% Very Bad	❑ 45% Bad	❑ 55% Acceptable	❑ 65% Good	❑ 75% Distinction	❑ 90% Exceptional

	Factors taken into account during evaluation:	**MAXIMUM MARKS**	**MARKS**
1	Problem definition		
2	Literature review (relevance, completeness, critical evaluation)		
3	Design of experiment		
4	Execution of experiment		
5	Presentation of results		
6	Analysis of results		
7	Evaluation of results		
8	Conclusions		
9	Recommendations		
10	Technical content		
11	Layout of report		
12	Style of writing		
13	Originality		
14	Level of difficulty		
15	Neatness of report		
16	General impression		
Final mark		100	

COMMENTS

Exainer: Date: