CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

FACULTY OF ENVIRONMENTAL SCIENCES

MASTER THESIS

Development of a tool for digital terrain analysis

Supervisor: Ing. VojtěchBarták, Ph.D.
Diplomant: Daria Rapoport

Prague 2016

DIPLOMA THESIS ASSIGNMENT

Daria Rapoport

Land and Water Management

Thesis title

Development of a tool for digital terrain analysis

Objectives of thesis

Create ArcGIS toolset for the calculation of flow accumulation by Multiple Flow Direction algorithm.

Methodology

First step will be to learn basics of programming in Python language as well as using it for writing scripts in ArcGIS. Next step will be to learn how to work with raster data types in Python: how to convert rasters to an array data structure, how to access values in the array and so on. It will be necessary to get accustomed to binary heap data structure and its application in work with raster data sets.

After that, a chosen flow routing algorithm will be implemented in ArcGIS by writing Python scripts and integrating it into newly created Toolbox.

Last step will be to demonstrate the new toolbox on real hydrologically correct digital terrain models and to compare the results with those derived by using built-in ArcGIS tools.

The proposed extent of the thesis

40 – 60 s.

Keywords

ArcGIS, Digital terrain analysis, Multiple Flow Direction, GIS, Python

Recommended information sources

Barták, V. (2008). Algoritmy pro zpracování digitálních modelů terénu s aplikacemi v hydrologickém modelování. Diploma thesis. pp. 202

Freeman, T. G. (1991). Calculating catchment area with divergent flow based on a regular grid. Computers & Geosciences, 17(3), 413-422

Garbrecht, J., & Martz, L. W. (2000). Digital elevation model issues in water resources modeling. Hydrologic and hydraulic modeling support with geographic information systems, 1-28.

O'Callaghan, J. F., & Mark, D. M. (1984). The extraction of drainage networks from digital elevation data. Computer vision, graphics, and image processing,28(3), 323-344.

Quinn, P. F. B. J., Beven, K., Chevallier, P., & Planchon, O. (1991). The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. Hydrological processes, 5(1), 59-79.

Expected date of thesis defence

2015/16 SS – FES

The Diploma Thesis Supervisor

Ing. Vojtěch Barták, Ph.D.

Supervising department

Department of Applied Geoinformatics and Spatial Planning

Electronic approval: 15. 4. 2016	Electronic approval: 15. 4. 2016
doc. Ing. Petra Šímová, Ph.D.	**prof. RNDr. Vladimír Bejček, CSc.**
Head of department	Dean

Prague on 17. 04. 2016

Declaration

I hereby declare that work presented in this diploma thesis is entirely done by myself. The used literature and sources are stated in the attached list of references.

Prague, 19th of April 2016_________________

Daria Rapoport

Acknowledgements

I would like to thank my superviserIng. Vojtěch Barták, Ph.D. for being patient with such a “nightmare” student, providing me with valuablesuggestions, comments and criticisms and supplying me with diploma thesis of previous diplomantPetr Novák, which helped me a lot in developing new tools.

Special thanks to my dear husband Andrey and daughter Iliana and all my family and friends for moral support, understanding and love.

Prague, 19th of April 2016_________________

Daria Rapoport

Abstract

Component of many hydrological applications using digital terrain models is thecalculation of flow accumulation and specific catchment area, which are broadly used conceptions in hydrological modelling.

The goal of this diploma thesiswas a proposal of tools for calculation of flow accumulation and specific catchment area by Multiple Flow Direction (MFD) algorithm as more appropriate for many hydrological applications then Single Flow Direction algorithm built-in in ArcGIS. The toolset should serve as an educational and research tool at the Faculty of Environmental Sciences, CULS, but not only there.

The theoretical part of the thesis gives examples ofdigital terrain models applications in general and in hydrology, summarizes basic methods that define flow direction and flowaccumulation of surface runoff in grid-based terrain models and their significance for hydrological modelling. Selected methods and their implementations are described indetails in methodological part of the thesis. The tools were created in Python language and maybe used in ArcGIS interface.

The functionality of the toolset was tested on a coarse-resolution hydrologically correctdigital terrain model of basin Moravian Dyjeand simulation results has been extensively compared to those produced by Single Flow direction algorithm.

It is suggested that created toolset would bedeveloped further in direction of calculation of additional related terrain attributes(e.g.Topographic Wetness Index), river network extraction with MFD, dealing with flat and sink areas in input DEM, and other flow direction algorithms implementation.

Keywords: ArcGIS, Digital terrain analysis, Multiple Flow Direction, GIS, Python

Content

1 Introduction

Determination of flow direction plays a key role in all steps of hydrological analysis based on Digital Elevation Models: detection and filling of sinks, calculation of flow accumulation and contributing area, delinating of watershed and detection of likely paths of toxic material (Gruber&Peckham, 2009; Mitchell, 2012, Resources.arcgis.com, 2015).

ArcGIS is one of the most popular user-friendly software, which permits to perform hydrological analysis, but among different flow routing algorithms only Single Flow Direction or D8 (O'Callaghan&Mark, 1984) is built-in in ArcGIS. TauDEM (Terain Analysis Using Digital Elevation Models) is ArcGIS Toolbox developed by Tarboton, which allows to calculate flow direction and contributing area by other single flow direction algorithm D-infinity (Tarboton, 1997) (Utah State Unversity, 2016).

Single Flow Direction algorithms have their limitations, especially in calculating of flow accumulation and specific catchment area (SCA), which are extremely important concepts for a calculation of range of hydrological indices. SCA is used extensively as a discharge indicator, thus broadly used in hydrology and geomorphology in studies of hillslope hydrological response, channel location, soil water content, landslide risk, long-term basin evolution and vulnerability to pollution (Costa-Cabral&Burges, 1994). Multiple Flow Direction algorithm (Freeman, 1991; Quinn et al., 1991) gives better SCA patterns then D8, especially on hillslopes, where natural divergence of flow occurs.

As no publicly-available ArcGIS Toolboxes with implemented Multiple Flow Directions algorithms so far exist, creation of such toolbox could greatly extend ability to use ArcGIS in hydrological analysis.

2 Aims of Diploma Thesis

Main goal of the diploma thesis is to create ArcGIS toolset for calculation of flow accumulation by Multiple Flow Direction algorithm (Freeman, 1991).

To achieve the goal several tasks have to be performed:

1) Write a literature review on basic flow routing algorithms, their applications in hydrology, advantages and disadvantages and methods for their comparison
2) Learn basics of programming language Python in general and its usage for ArcGIS, learn how to work with raster datasets
3) Write scripts and create tools for determination of flow direction, flow accumulation and specific catchment area by Multiple Flow Direction algorithm
4) Compare results of flow simulation obtained by SFD and MFD algorithms on chosen dataset

3 Literature Review

3.1 Introduction to DTM

Digital Terrain Model (DTM) is a representation of terrain surface in digital form. There are several similar terms for digital terrain models.According to Mach and Petschek (2007), *Digital Terrain Model (DTM)* is pure geometric information of the terrain surface and is also called *Digital Elevation Model (DEM)*. *Digital Landscape Model (DLM)* or *Digital Surface Model (DSM)*in addition include description of buildings and vegetation.

DEM is digital or numeric representation of topography (elevations as a function of geographic location) (O'Callaghan&Mark, 1984; Garbrecht&Martz, 2000) or a computerized representation of the Earth's relief, when the elevation information is represented as elevation data in a digital format (Sulebak, 2000).

Mach and Petschek (2007) give a history of terrain representation models starting with the use of perspective for realistic 3-D depicting of terrain in Leonardo da Vinci's maps of Tuscany in 1502-1503, through the 1st using of contour lines by Pieter Bruinss in 1584 for his sea charts in Netherlands, then the use of shading and coloring of height layers without side view in 19th century and, finally, the 1st DEM creation by Miller and Laflamme in the Photogrammetry Laboratory of the Civil Engineering Department of M.I.T. in the late 50s of 20th century with the intention of its employment in road construction.

The basis of DTM creation are aerial images and stellite photos(Mach and Petschek, 2007) It is so-called remote-sensing in contrast to on site observation or obtaining of information without physical contact by using special sensors placed on helicopters, planes, and satellites, which measure an object's transmission of electromagnetic energy from reflecting and radiating surfaces (Pidwirny, 2006). For many parts of the earth's surface analogue contour maps are alredy exist, and are stored in digitized form (O'Callagan&Mark, 1984).

There is number of commercial and state providers of DEM data of different quality, obtained using different methods including new technologies such as Laser Altimetry (LA) and Radar Interferometry (RI). For example, in the United States the most widely available DEMs are distributed by US Geological Survey (USGS). USGS DEMs have different quality levels: 1) data obtained from National Photography programs with the maximum vertical Root Mean Square Error (RMSE) of 15 m (7m as the targeted accuracy standart); 2) processed data sets smoothed for consistency with removed systematic errors, with the maximum RMSE of one-half of the original map contour interval; 3) data derived from hypsography (contours, spot elevations), hydrography (lakes, shorelines, drainage) and others features (ridge lines, main transporation featuers) with the maximum RMSE of one-third of the contour interval. Level 2 data are now the most available now (Garbrecht&Martz, 2000).

Level 1 data sets associated with manual profiling of photogrammetric stereo-models often display east-west striping patterns that make them unsuitable for drainage features parameterization as first, outlines of drainage features (depressions, drainage paths) are not well defined, second, drainage paths are biased towards east-west, third, north-south flow path can be blocked by this striping creating artificial depressions. An elevation difference even of 1-2 meters can greatly affect flow path and runoff characteristics (Garbrecht&Martz, 2000), thus level 1 data sets are not good for drainage investigations.

Another important feature of DEM data is vertical resolution and horizontal to vertical resolution ratio, which is important for the calculation of a slope. Since DEM elevations are generally in integer meters, the computed slope takes only a limited number of discreet values. This becomes problematic for low relief landscape leading to artificial pits and flat areas calculation with lack of downslope flow paths, which leads to incomplete drainage pattern definition (see*3.1.2.Errors in DEM*) (Garbrecht&Martz, 2000). Level 1 and level 2 data both have coarse resolution (precision of data) and broad accuracy standards for the elevation, which makes surface drainage modelling difficult in low relief landscapes, as is derivation of such parameters as slope and curvature.

3.1.1 DEM types

Different DEM formats exist, among the most usual are *triangulated irregular networks (TIN)*, *regular grids*, *contour lines* and *scattered data points*(O'Callaghan&Mark, 1984; Quinn et al., 1991; Tarboton, 1997; Sulebak, 2000). TIN and contour lines are continuous models, while grid DEM is a discrete model (Freeman, 1991).

Usually initially we have scattered points of known elevation values, which should be interpolated to one of the three model structure: contours, TIN or grid.TIN is original scattered point, which is connected by irregular triangles constructed usually using Delauney triangulation(*Figure 3.1.*) (Wilson&Gallant, 2000; Wise, 2013). Contours are isolines of the same heights(*Figure 3.2.*). The main advantage of contour based models is that they unambiguously define where the runoff goes – perpendicular to contours (Holmgren, 1991).

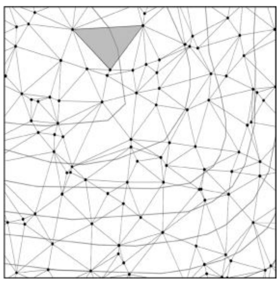


Figure 3-1(Wise, 2013). TIN model for surface data. Triangles have been created by joining the spot heights.

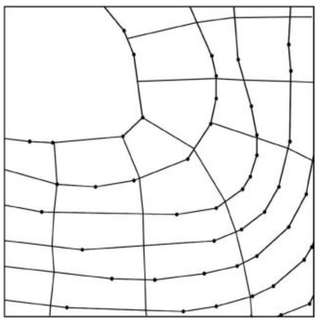


Figure 3-2.(Wise, 2013). Contour-based model for surface data.

Grid-based DEM, other terms are cell-based DEM, raster DEM, is a matrix of square grid cells with the mean cell elevation stored in 2-D array(*Figure 3.3.*). The cell column and the row define its geographic location, provided that georeferences (the boundary coordinates of the array) are known (Garbrecht&Martz, 2000).

Cell-based DEMs are the most common digital data of the shape of the earth's surface (Resources.arcgis.com, 2015; Wise, 2013). Besides, square DEMs are more preferable in comparison with the triangular pixels in distributed models, as TINs greatly increase necessary processing giving in return little accuracy (Lea, 1992). Raster digital elevation models (DEM) are widely used source of data in hydrology, soil science, ecology and have great potencial because of its rapidly increasing quality and availability (Bartak, 2009). Grid DEM data structure is commonly used for its simplicity, processing ease, computational efficiency (Garbrecht&Martz, 2000) and wide availability (O'Callaghan&Mark, 1984).Grid based DEMs are uncomplicated for model programming and readily compatible with other data types, such as satellite images (Holmgren, 1991). As grid DEM is stored as a simple matrix, values are accessed easily without having to resort to special data structure or interpolation procedures (Freeman, 1991).

There are some disatvantages of raster DEM models. First, raster resolution affects the storage requirments, computational efficiency, and the quality of results. Second, square grids cannot handle abrupt changes in elevation and skip important details of the land surface, especially if elevation data are stored in whole meters instead of floating point numbers. Third, computed flow paths tend to zigzag across the landscape causing difficulties to calculate specific catchment area accurately (see *3.4.Flow routing algorithms*) (Wilson&Gallant, 2000).

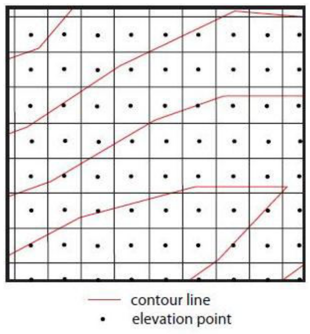


Figure 3-3(„Digital Topography...", 2016). Grid-based DEM. Equally spaced points in the center of each square grid cell represent elevation of topography.

Currently, different interpolation methods exist to convert one DEM type into other, so initial choice of data structure is not so important (Wise, 2013).

3.1.2 Errors in DEM

For a fully exploitation the DEMs should be available at sufficient accuracy, detail, projection and format (Sulebak, 2000).*The accuracy of the DEM* is determined by: 1) the resolution (the distance between sample points); 2) data type (integer or floating point) and 3)the actual sampling of the surface when creating the original DEM (Resources.arcgis.com, 2015).

In some situations, the automated correlation of stereophotos for elevation matrices production, may rise errors, because they reflect the elevations of the tops of vegetation or some surface features rather than the ground (Freeman, 1991). Interpolation from scattered points will rise errors as well (Freeman, 1991).

Due to inaccuracy in the DEM, errors may occur. *Errors in DEMs* are usually *sinks* or *peaks*. *A sink (depression, pit)* is a cell or group of cells surrounded by cells of higher elevation, thus is an area of internal drainage. The flow will travel into the sink but will not travel out creating a break in the stream. Thus, stream segments are disconnected and do not form a complete network (Mitchell, 2012).Some of these pits, howwever, may be natural, particularly in glacial or karst areas, or calderas, but many sinks are imperfections in the DEM (O'Callagan&Mark, 1984). Generally, sinks with a drop of more than 30 meters from adjacent cells do not occur naturally, except for karst and glacial areas, so it is possible to remove only sinks with a drop greater than a certain value (such as 30 meters) (Mitchell, 2012). Sinks sometimes occur in flat or low elevation areas, such as along wide stream channels, in marshes, floodplains or along a coastline, so it is highly recommended to use rasters with floating point values rather than integer to minimise sink errors (Mitchell, 2012). O'Callaghan&Mark (1984) confirm that most pits in fluvial landscapes are due to data errors, especially where the overall slope is gentle (the „signal" is weak). Sinks in grid DEM may occur after interpolation from irregular point elevations (Quinn et al., 1991).

The number of sinks is higher for DEM with coarser resolution and if elevation data are stored as integers (not as numbers with a floating point), particularly in areas of low vertical relief. For example, 1 percent of the cells in a 30-meter-resolution DEM is to be sinks. This can increase as much as 5 percent for a 3-arc-second DEM, the actual distance in ground units of which varies with latitude, but, for example, in the vicinity of Southern California, measure 76.86 meters in x and 92.36 meters in y (Resources.arcgis.com, 2015).

Likewise, *peak (spike)* is an area surrounded by cells of lower value, more commonly natural and less detrimental to the calculation of flow direction (Resources.arcgis.com, 2015).Such,dam' features are created eighther during interpolation being artificially raised above ‚true' surface, or if the grid scale is too coarse to resolve local incised channel features (Quinn et al., 1991).

The finest resolusion DEMs will ensure accuracy of the hydrological analyses results. For study area at the county or regional level, a DEM with 10 or 30 meters cell size will suffice. Most publicly available DEMs need to be processed before using them – errors such as sinks should be identified and fixed (Mitchell, 2012).

Such errors, especially sinks, should be removed before using DEM for deriving any surface information, like slope and aspect calculation and flow direction determination (Resources.arcgis.com, 2015). Peaks as well as sinks to some extent can be mitigated by smoothening the surface using special filtering tools, but some details may be lost in the process. Usually sinks are „filled": the cell value is changed to the lowest value of surrounded cells, so water no longer accumulates in the cell (Mitchell, 2012). It is done in an iterative process, as the boundaries of the filled area may create new sinks, which then need to be filled (Resources.arcgis.com, 2015). Natural sinks should be flagged to avoid removing in DEM preprocessing as well(O'Callaghan&Mark, 1984), so user should be accostomed to the real geomorphology of the area he is working with.

Flat spots are groups of points with equal heights resulted from storing elevation data in integers instead of floats, interpolation from close cotours or from some mehods of eliminating sinks (Freeman, 1991). Different approaches for handling flat areas, when determining flow accumulation, exist (e.g. approach adopted by Jenson and Dominigue (1988))

Dealing with the natural flat areas in DEM, such as lakes or ocean, GIS will attempt to calculate drainage channels across these flat areas. To avoid this, such areas are assigned NoData, so no calculation will occur (Mitchell, 2012).

DEMs may also contain *striping artifacts*, a result of systematic sampling errors when creating the DEM, the most noticeable on integer data in flat areas(Resources.arcgis.com, 2015).". Artifacts in general are spatially structured errors of a systematic nature (Albani and Klinkenberg, 2003).Striping can be described as successive ridges and valleys, usually oriented approximately East-West, that run through the elevation model. Such striping is an artifact of a data collection technique known as Manual Profiling, prone to systematic errors (Russell, Kumler & Ochis, 1995).Before using these data forapplications such as slope analysis, watershed delineation, water flow patterns, or landscape rendering, DEMs must be filtered. For now, different destriping methods exists, e.g. Fast Fourier Transform (Russell, Kumler & Ochis, 1995) or spatial filtering algorithmfor treating vertical striping (Albani and Klinkenberg, 2003).

3.1.3 DTM Applications

The knowledge of the surface topography is essential in process modeling in hydrology, climatology, geomorphology and ecology, and is a prerequisite for many applications in civil and military agencies, in industrial areas like telecommunications (specifically, radio wave propagation), navigation, disaster management (prevention,relief, assessment), transportation andinfrastructure planning (Sulebak, 2000). For example, distributed watershed models, used for investigation of hydrological processes and water related problems, require such parametrs as channel network configuration, channel lengths and slopes and others, which can be obtained from DEM (Garbrecht&Martz, 2000). Hydrological parameters extracted from DEMs such as flow path and slope are used as inputs into rainfall-runoff model for flood forecast (Sulebak, 2000).Generally, an accurate distributed model must reflect proper flow pathway and spatial and temporal variations in flow velocity, which are the function of the gravitational potentials defined by the topography in catchmets dominated by surface flow processes, which makes DEM a powerful mean for hydrological modelling in such areas. Surface flow dominates at valley bottoms, while elsewhere in the catchment, a deeply weathered soil and deep water table mean that the surface topography may not be a good indicator for flow pathways (Quinn et al., 1991).

According to Weibel and Heller (1991, exMach and Petschek, 2007, p.6), five fields of terrain models applications can be defined, and Mach and Petschek (2007) added the 6th field:

1) Surveying and photogrammetry
2) Civil engineering and landscape architect
3) Resource Management
4) Geology and earth sciences

5) The military
6) Game industry

Sulebak (2000) gives another classification of DTM applications with vast examples in different areas:

- Scientific applications (e.g. in the fields of ecology and wildlife management, hydrological modelling, geomorphology and landscape analysis, climatology, mapping)
- Commercial applications(e.g. in telecommunication, air traffic routing and navigation, planning and construction of roads, railway tracks, gas pipelines etc., geological exploration, hydrological and meteorological services including risk assessment for insurance companies, geocoding of remote sensing and market of multimedia applications and computer games)
- Industrial applications (in telecom for positioning of radio towers, intelematics for car navigation systems and transportation network planning, in avionics for collision avoidance and flight management systems and training flight simulators, inmining andmineral exploration for promising regions determination, in tourism for virtual cruise over the terrain of region of interest)
- Operational applications (e.g. in regional planning for location of infrastructure developement and investment, for finding of ground drinkingwater sources in arid regions, in forest planning and management for calculation of slope and respective erosion process due to clear cutting of hill slopes, for aspect computation and solar insolation effect on forest growth, for breakwater location along the coastlines, for detection of risk areas of rock fall and avlanches to prevent building constructions or start protection actions, in disaster management for damages and changes analysis or location of spots for dropping of aid supplies).
- Military applications

Topographic analysis of DEM in hydrology is typically used as surrogate for the spatial variation of hydrological conditions (topographic indices) and flow routing (Seibert&McGlynn, 2007). Topographic attributes are used for quantitative spatial predictions of specific soil properties, landslides, vegetation and land cover types, for landform classifications (Wilson&Gallant, 2000), hazard mapping (ice/avalanches, debris flow) in steep terrain (Gruber & Peckham, 2009).

3.1.4 Attributes derived from DEM

Topography plays an important role in distribution of energy and water on the land surface. As quality and resolution of DEM data as well as GIS capabilities is increasing, DEM serves as the main source for automated topographic parameters extraction. For example, distributed watershed models, used for investigation of hydrological processes and water related problems, require such parametrs as channel network configuration, channel lengths and slopes and others, which can be obtained from DEM (Garbrecht&Martz, 2000). The accuracy of these topographic properties is a function of, firstly, DEM quality and resolution, and secondly, DEM processing algorithms used for these properties extraction (Garbrecht&Martz, 2000).

These parameters or attributes can be divided into primary terrain attributes (e.g.slope, aspect, horizontal and vertical curvature)(Table 3.1.), which are derived directly from the elevation data,and secondary or compound terrain attributes (e.g. topographic index or wetness index of soil saturation) (Table 3.2.), which are derived from the primary ones (Moore et al., 1991).

Seibert and McGlynn (2007) use term “topographic indices” rather than “terrain attributes”, dividing them into locally determined (elevation, slope, aspect, curvature, upslope contributing area etc.) and combinations of indices (topographic wetness index), which are basically equivalent terms for the primary and secondary terrain attributes respectively.

Primary attributes describe the morphometry and catchment position, while secondary attributes describe patterns as a function of process – they quantify role of topography in water redistribution or in recieved solar radiation modification, which in itc turn, affect characteristics of soil, flora and fauna (Wilson&Gallant, 2000).

For example, *topographic wetness* index (TWI) ln(a/tanβ) reflects the tendency of water to accumulate at any point of the catchment (in terms of a – cumulutive upslope area draining through the point per unit contour length(see *3.2.Basic terms or understanding drainage systems*) and the tendency for gravitational forces to move water downslope (in terms of tanβ as an approximate hydraulic gradient (Quinn et al., 1991) or, in other words, measure of the potential drainage from a place (Seibert&McGlynn, 2007)). Values a and tanβ in its turn depend on the analysis of flow pathways (see *3.4.Flow routing algorithms*) and grid resolution. The higher the grid resolution, the larger the area of each cell, wich leads to the higher percentage of higher topographic index, as low values of ln(a/tanβ) no longer exist (Quinn et al., 1991).

It is important to note that such primary terrain characteristic as a slope, which is used in calculations of some other terrain attributes like TWI and in determination of flow direction by some flow routing algorithms (see *3.4.Flow routing algorithms*), can be actually calculated in different ways. Let us consider we want to calculate slope in central cell of grid based DEM (*Figure 3.4.*).

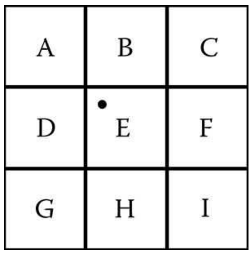


Figure 3-4(Wise, 2013). Location of cells in 3 × 3 window used for estimating surface properties at a point. Letters stand for different elevations in the centers of pixels.

To define slope in y direction, we can use three approaches. The simplest method is to take into calculations only the elevations from the points below and above the central point:

$$\text{Y slope} = \frac{B - H}{2d}$$

, where d is grid spacing. The result will be very dependent on the accuracy of the heights in each of the grid points.

Table 3-1.(Wilson&Gallant, 2000). Primary attributes derived from DEM

Attribute	Definition	Significance
Altitude	Elevation	Climate, vegetation, potential energy
Upslope height	Mean height of upslope area	Potential energy
Aspect	Slope azimuth	Solar insolation, evapotranspiration, flora and fauna distribution and abundance
Slope	Gradient	Overland and subsurface flow velocity and runoff rate, precipitation, vegetation, geomorphology, soil water content, land capability class
Upslope slope	Mean slope of upslope area	Runoff velocity
Dispersal slope	Mean slope of dispersal area	Rate of soil drainage
Catchment slope	Average slope over the catchment	Time of concentration
Upslope area	Catchment area above a short length of contour	Runoff volume, steady-state runoff rate
Dispersal area	Area downslope from a short length of contour	Soil drainage rate
Catchment area	Area draining to catchment outlet	Runoff volume
Specific catchment area	Upslope area per unit width of contour	Runoff volume, steady-state runoff rate, soil characteristics, soil-water content, geomorphology
Flow path length	Maximum distance of water flow to a point in the catchment	Erosion rates, sediment yield, time of concentration
Upslope length	Mean length of flow paths to a point in the catchment	Flow acceleration, erosion rates
Dispersal length	Distance from a point in the catchment to the outlet	Impedance of soil drainage
Catchment length	Distance from highest point to outlet	Overland flow attenuation
Profile curvature	Slope profile curvature	Flow acceleration, erosion/deposition rate, geomorphology
Plan curvature	Contour curvature	Converging/diverging flow, soil-water content, soil characteristics
Tangential curvature	Plan curvature multiplied by slope	Provides alternative measure of local flow convergence and divergence
Elevation percentile	Proportion of cells in a user-defined circle lower than the center cell	Relative landscape position, flora and fauna distribution and abundance

Table 3-2.(Wilson&Gallant, 2000).Some of the secondary attributes derived from DEM

Attribute	Definition	Significance
Topographic wetness indices	$W_T = \ln\left(\frac{A_s}{T \tan \beta}\right)$	This equation assumes steady-state conditions and describes the spatial distribution and extent of zones of saturation (i.e., variable source areas) for runoff generation as a function of upslope contributing area, soil transmissivity, and slope gradient.
	$W = \ln\left(\frac{A_s}{\tan \beta}\right)$	This particular equation assumes steady-state conditions and uniform soil properties (i.e., transmissivity is constant throughout the catchment and equal to unity). This pair of equations predicts zones of saturation where A_s is large (typically in converging segments of landscapes), β is small (at base of concave slopes where slope gradient is reduced), and T_i is small (on shallow soils). These conditions are usually encountered along drainage paths and in zones of water concentration in landscapes.
	$W = \ln\left(\frac{A_e}{\tan \beta}\right)$	This quasi-dynamic index substitutes effective drainage area for upslope contributing area and thereby overcomes the limitations of the steady-state assumption used in the first pair of equations.
Stream-power indices	$\text{SPI} = A_s \tan \beta$	Measure of erosive power of flowing water based on assumption that discharge (q) is proportional to specific catchment area (A_s). Predicts net erosion in areas of profile convexity and tangential concavity (flow acceleration and convergence zones) and net deposition in areas of profile concavity (zones of decreasing flow velocity).
	$\text{LS} = (m+1)\left(\frac{A_s}{22.13}\right)^m \left(\frac{\sin \beta}{0.0896}\right)^n$	This sediment transport capacity index was derived from unit stream power theory and is equivalent to the length–slope factor in the Revised Universal Soil Loss Equation in certain circumstances. Another form of this equation is sometimes used to predict locations of net erosion and net deposition areas.
	$\text{CIT} = A_s (\tan \beta)^2$	Variation of stream-power index sometimes used to predict the locations of headwaters of first-order streams (i.e., channel initiation).

Second method deals with the difference between the average height of the points from the rows above and below the central point:

$$Y \text{ slope} = \frac{(A + 2B + C) - (G + 2H + I)}{8d}$$

Third method is to fit a surface through the points in the neighborhood, which will actually allow us to calculate slope not only of a central point but at any other points on the fitted surface. However, interpolation method should be chosen carefully. If we try to fit surface to every point, resulting surface may bend rather sharp, giving additional errors in-between points. Better way may be to capture overall trend, rather than trying to fit surface to every point. *Figure 3.5.* explains the differences using example of fitting line for series of points.

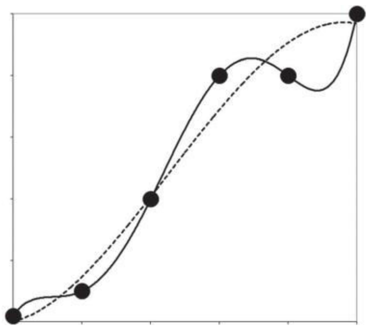


Figure 3-5(Wise, 2013). Lines through a series of points. Solid line – fifth-order polynomial. Dotted line – third-order polynomial.

3.2 Basic terms or understanding drainage systems

A drainage system is the area upon which water falls plus the network through which it travels to an outlet. *Drainage basin*(known as catchment, contributing area, drainage area,upslope area, flow accumulation, basin or watershed) is an area of land, from which all water and other substances come to a common *outlet*, or a *pour point*, the lowest point along the drainage basin boundary, at which water flows out of an area.*A drainage divide* or *watershed boundary* is the boundary between two basins (*Figure 3.6.*) (Resources.arcgis.com, 2015).

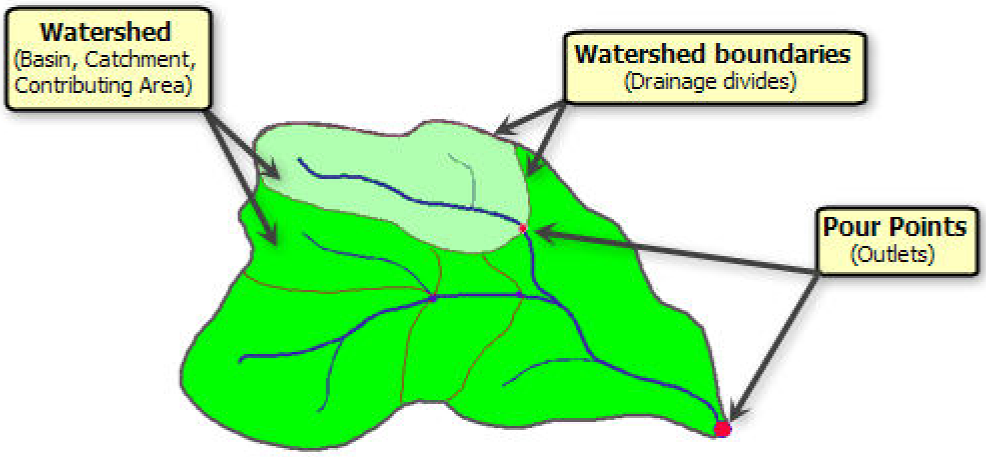


Figure 3-6(Resources.arcgis.com, 2015) Components of drainage basin

Basins differ by some hydrological and morphological characteristics: length of water streams, waterdivide location, area of elementary subcatchments and sub-waterdivides, slopes etc. Traditionally computation of these characteristics is based on maps and ground-based measurements, but alternative way is use of DEM, which can greatly speed up the process through automatization.

The total contributing area (TCA) of a contour segment is an uphill area of terrain that contributes flow to the contour segment. *Specific contributing area* (SCA) is TCA divided by contour segment length, may be interpreted as an equivalent upslope flow path length (Costa-Cabral&Burges, 1994). SCA indicates the amount of area flowing to a specific location, may serve as a proxy for water flow (Seibert and McGlynn, 2007). SCA is used extensively as a discharge indicator, thus broadly used in hydrology and geomorphology in studies of hillslope hydrological response, channel location, soil water content, landslide risk, long-term basin evolution and vulnerability to pollution (Costa-Cabral&Burges, 1994). Most often in literature SCA is used for the calulation of topographic wetness and stream-power indices or similar indices (see *3.1.4 Attributes derived from DEM*).Thus, SCA is important parameter for the determination of saturation zones, erosion and deposition potentials, locations of headwaters of first-order streams (i.e. channel initiations) (Wilson&Gallant, 2000).

3.3 Hydrologicall modelling with DEM

Most hydrologically-oriented digital terrain analisis (DTA) techniques are based on models of catchment rainfall-runoff response and river basin management needs with the main tasks of river network and catchment boundaries identification. Some methods are based on local topology evaluation (ridges and valley lines determination), while others are modelling of overland flow, hypotetical flow over completely uncovered impermeable terrain after initial unit rainfall regularly spaced over entire DEM (Bartak, 2009).

Overland flow modelling process in general consists of: 1)assignment of water flow directions from a particular cell to one or several neighboring cells; 2)computation of flow accumulation (upslope area draining through a particular cell, so-called contributing area) and respectively size and the position of the catchment (Bartak, 2009).

> Flow accumulation models determine where water flows and accumulates on a terrain surface, which allows then to define stream channels and hydrological basins and to measure the amount of rainfall runoff and respectevely substances carried by the flow that will accumulate at a given downstream point (e.g. watershed outlet) and how long it will take to travel there (Mitchell, 2012).

General steps for modelling flow accumulation (Mitchell, 2012):

1) Obtain the elevation surface (choose DEM of appropriate resolution, remove errors from DEM)
2) Create the flow direction surface
3) Create the required output (drainage network, drainage area boundaries, flow volume, travel time through a drainage area)
4) Evaluate the results

With ArcGIS software hydrological modelling and calculation of catchment propertiescan be performed by using instruments *Hydrology* of*Spatial Analyst toolbox* and additionally created modules. The hydrologic tools allow to identify sinks, determine flow direction, calculate flow accumulation, delineate watersheds, and create stream networks. Using DEM as input, it is possible to automatically delineate a drainage system and quantify its characteristics. What is more, the first step of extracting hydrologic information, such as watershed boundaries and stream networks, from a digital elevation model (DEM) is assigning flow direction (Figure 2) (Resources.arcgis.com, 2015). (O'Callaghan&Mark, 1984)

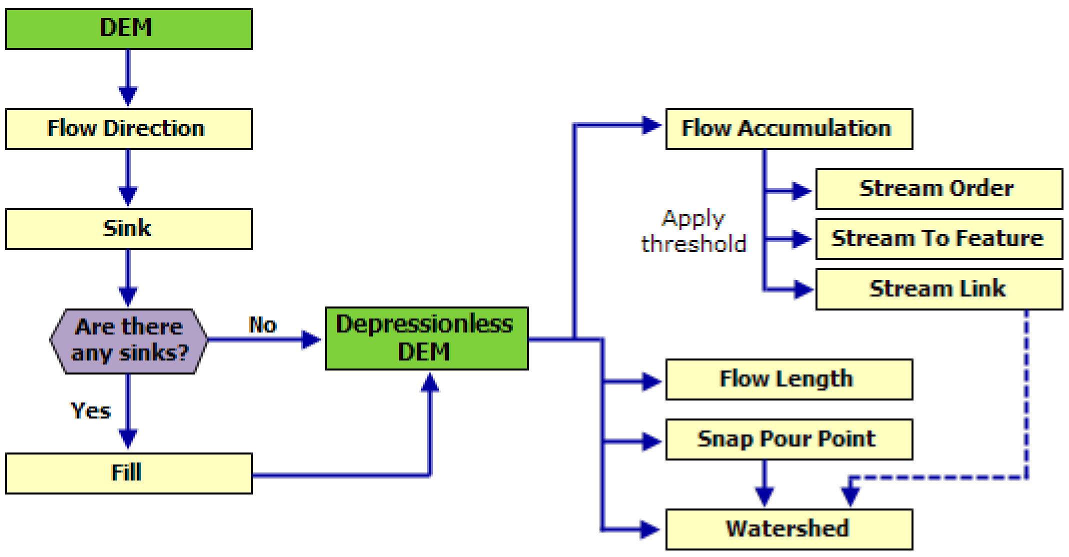


Figure 3-7(Resources.arcgis.com, 2015) Hydrological modeling flowchart

Thus, flow direction plays a key role in all subsequent steps of hydrological analysis: detection and filling of sinks, calculation of flow accumulation and contributing area, delinating of watershed.

The ArcGIS hydrologic analysis tools are designed to model the convergence of flow across a natural terrain surface with an assumption that the surface contains sufficient vertical relief that a flow path can be determined (Resources.arcgis.com, 2015). Here appears the problem of flat areas. Another assumption is that for any single cell, water can flow in from many adjacent cells but out through only one cell (Resources.arcgis.com, 2015) as D8 flow direction algorithm is used follows an approach presented in Jensen and Domingue (1988).

Delineating drainage channel is based on specifying a threshold or cutoff value for flow accumulation. All cells greater than a threshold will be part of channels. For example, if you know that in your study area a stream will form when the drainage upstream is at least 100 hectares in size, you need to calculate how many cells are in 100 hectares from a cell size value of flow accumulation raster, this will be a threshold (Mitchell, 2012). This approach is based on a hydrophysical assumption that a drainage channel appears at points at which runoff is sufficiently concentrated that fluvial processes dominate over slope processes (O'Callaghan&Marka, 1984). Changing the threshold, we change the resolution of the drainage system: the higher threshold value, the more major streams will be identified, smaller threshold will reveal water flowing only during a rainstorm (Mitchell, 2012). Speight (1968, from O'Callaghan&Mark, 1984) applied the same principle manually: he draw a grid on a contour map and traced a slope lines perpendicular to the contour lines from each grid cell – cells with more than 100 slope lines were assigned to lie on a "watercourse".

3.4 Flow routing algorithms

The first step of extracting hydrologic information, such as watershed boundaries and stream networks, from a digital elevation model (DEM) is assigning flow direction (*Figure 3.7.*) (Resources.arcgis.com, 2015). (O'Callaghan&Mark, 1984)

Moreover, most distributed catchment models require some topographic input, many of which require flow pathways defined a priori on the basis of catchment topography (Quinn et al, 1991). Thus, the accuracy of predictions of distributed hydrological models partly depends on proper flow pathways identification. For example, TOPMODEL predicts distributed moisture status on the basis of spatial indices which depend on flow direction (Quinn et al., 1991).

Determination of flow directions in hydrology is important for defining water, sediment and contaminants movement. Upslope area A (total catchment area above a point or short length of contour, TCA) and specific catchment area $a = A/L$ (upslope area per unit width of contour L contributing to flow, SCA) are dependent on flow direction. Specific catchment area is useful for calculating relative saturation and runoff from the saturation excess in models such as TOPMODEL, in erosion and landslides analysis (Tarboton, 1997). There are lots of hydrological indices that combines SCA as a discharge indicator with other variables (e.g. ln(SCA/Slope) is used to predict the soil moisture deficit, $\text{Slope}^2*\text{SCA}$ is used to predict channel initiation by overland flow (Costa-Cabral&Burges, 1994)).

When there is not enough data, flow routing algorithms may serve as a simple substitute for a distributed hydrological model to predict stream and catchment locations and can form the basis of a simple sediment redeposition model to be used in conjunction with empirically developed soil loss estimators such as the Universal Soil Loss Equation in sediment yield modelling (Lea, 1992). A simple sediment redeposition model for each pixel requires estimation of the delivery ratio (i.e. proportion of sediment redeposited before reaching the outlet), which is calculated in two components: overland redeposition and river and floodplain redeposition. The delivery ratio during the overland flow phase (i.e. until flow reaches a principal stream) is dependent on the length and slope of the flow path, and the overland discharge at each pixel in the flow path, which can be estimated after implementation of flow routing algorithms (Lea, 1992).

Designing and evaluating DEM flow direction procedures several issues should be taken into account (Tarboton, 1997):

1) the need to avoid or minimize the dispersion (which is important for the calculation of TCA);
2) the need to avoid grid bias due to the grid orientation;
3) the precision with which flow directions are resolved;
4) a simple and efficient grid based matrix storage structure;
5) robustness to 'difficult' data (e.g. saddle, pits and flat areas)

Evaluation of flow direction algorithmcan be done by checking:

1) Consistency of channels derived by algorithm with the channels which could be inferred from contours by experienced geomorphologist – realistic or unrealistic patterns (O'Callaghan&Mark, 1984)
2) Performing the analysis on different landform type, finding "best" and "worst" case for the method (O'Callaghan&Mark, 1984), on artificial ideal surfaces like cone, saddle, comparing elevation contours of analytically 'real' case, resulting catchment area map and resulting drainage contours (Freeman, 1991), so that square error can be evaluated quantitatively (Tarboton, 1997).
3) Overlap proportion between analytically simulated flow distributions and flow distributions of tested model parameters (Holmgren, 1994)
4) Performing the analysis on low and high resolution DEMs (e.g. specific catchment area computation depends not only on calculation procedures, but also a grid scale, which can be used for new approach evaluation (Tarboton, 1997))
5) Accuracy of prediction of some terrain properties (e.g. ephemeral gully locations (Desmet and Govers, 1996) or different soil characteristics like pH, soil organic matter etc (Chirico et al., 2005)), which are correlated to the secondary attributes dependent on accumulated area calculation

There are different approaches for assigning of flow directions. Several reviews on contemporary flow routing algorithms exist (Bartak, 2009;Wilson et al., 2008), some algorithms sometimes have different names, which will be mentioned below.

Generally, all algorithms can be divided into two main groups: single and multiple flow direction in terms of flow routed to one or several cells. Single flow direction algorithms are algorithms permitting only convergence of flow and they are working well at valleys, where flow paths merge to form streams. But multiple flow direction algorithms, in its turn, permit flow divergence, which is naturally occurring on a cone-like divergent slopes, which are common in real world DEMs (Seibert&McGlynn, 2007). The more exact simulation of this divergence is needed for more precise calculation of flow accumulation (or contributing area), which is an important parameter for calculation of many secondary topographic attributes (like TWI, total wetness index, which is used in modelling of soil erosion). O'Callaghan&Mark (1984) report that multiple flows do occur naturally in flat areas, but have no effect on drainage channel pattern within data sets studied in their preliminary work.Too high flow dispersion though should be avoided. Upslope area *A* is a total catchment area draining to a *point*, that is why dispersion is nondesirable and should be minimized (Tarboton, 1997).

Algorithms can be also classified in a way, how they treat flow - as a point source (1D) or areal source (2D).

Other, in my opinion, very important differences between existing algorithms are how they define contour length, which is extremely important for calculation of SCA. Even different implementations of the same algorithm can use different conceptions of contour length (e.g. author of MFD algorithm Quinn (1991) writes in his article that contour length were defined arbitrary). If the purpose is channels demarcation based on threshold or critical support area, the different methods give similar result, as in channels flow concentration occurs. However, if the intent is to use SCA for distributed hydrological modelling, like runoff generation or erosion, different algorithms will give different results, especially on hillslopes with small values of flow accumulation. Finer DEM resolution will increase these differences. Thus, choice of flow routing algorithm is critical in that case (Tarboton, 1997).

3.4.1 D8

First and the simplest flow directing method is*SFD8 (Single Flow Direction from 8 possibilities)*or*D8 (Deterministic 8)*algorithm (O'Callaghan and Mark, 1984), which routes flow from a given cell towards only one cell, the steepest of its neighbours (*Figure 3.8*.). Other names are„nearest neighbour“ or „lowest neighbour“ algorithm (Lea, 1992), „the steepest descent algorithm“ (Freeman, 1991). The steepest descent for each cell is marked by the aspect (measured in degrees clockwise from the north) or using primary flow direction determined by the steepest slope (Wilson et al., 2008). The slope from the cell to its neighbour is calculated as the difference in elevations divided by the horizontal distance between the cell and its neighbour (cell size for cardinal neighbours and cell size multiplied by √2 for diagonal neighbours), and is considered as positive downhill (Bartak, 2009). For example (*Figure 3.8*), slope from the central cell to the right cell $S_{rigth} = (5-2)/1=3$, where 5 is elevation of the central cell, 2 is the elevation of the right cell and 1 is the grid width or the horizontal distance between these two cardinal cells. Analogically, slope from the central cell to the right lower corner cell $S_{lowright} = (5-1)/1*\sqrt{2}=2,83$, where 5 and 1 are elevations of the central and lower right cornor cells respectively and horizontal distance between them is grid width multiplied by √2 as the cells lay in diagonal.Thus, in D8 algorithm all water from one cell enters another single cell downslope.

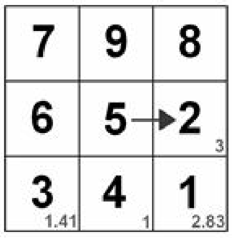


Figure 3-8(Bartak, 2009) The SFD8 algorithm. The large numbers denote the elevations, the small numbers denote the slope from the central cell, and the arrow represents the determined flow direction.

To prevent drainage channel forming at the matrix edge, edge points are assigned as ridges or pits so no input/output paths can be generated (O’Callaghan&Mark, 1984).

The upslope contributing area or total catchment area (TCA) is calculated as the number of cells whose flow reaches the cell of interest multiplied by the cell area.Specific catchment area (SCA) is the TCA divided by contour width. In some implementations of D8 contour width is assumed to be equal to the width of the cell grid or raster resolution for both cardinal and diagonal directions (Quinn et al., 1991; Gallant and Wilson, 1996), while in others, for example, in TAPES-G software(Gallant and Wilson, 1996), flow width for diagonal directions is equal to grid width multiplied by √2. Local slope of a current cell is simply assigned as the steepest downhill slope (Quinn et al., 1991).

Thus, flow is always convergent and biased towards one of the eight possible directions (extremly dependent on grid orientation), which is main disadvantage of D8 (Costa-Cabral&Burges, 1994; Bartak, 2009). It can be clearly seen on artificial surface such a cone and saddle (Freeman, 1991). On *Figure 3.9.* TCA and SCA on a cone produced by D8 is compared to the true pattern: for some pixels, TCA is overestimated by a factor of 2, while for others TCA is underestimated to the pixel size as these pixels drain only themselves. D8 does not represent well the surface flow over many natural terrains, especially on divergent surfaces (Freeman, 1991). Since flow cannot be distributed to two cells, flow tends to become concentrated to distinct, artificially straight lines. Additional problem is that the steepest gradient can in fact fall between two of the eight possible directions (Seibert&McGlynn, 2007).

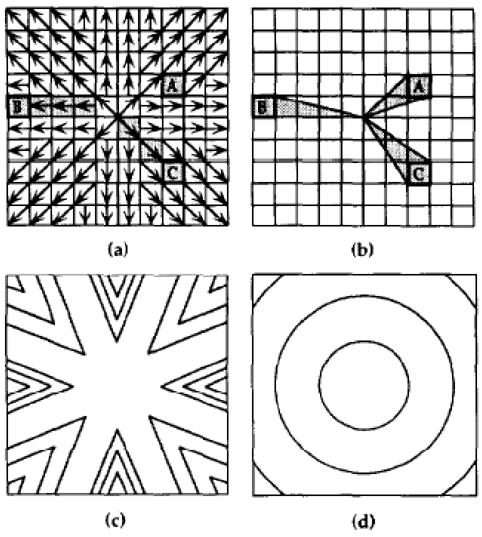


Figure 3-9.(Costa-Cabral&Burges, 1994). Flow patterns of a right circular: (a) TCA of pixels A, B, C predicted by D8; (b) true TCA of pixels A, B, C; (c) SCA contours for D8; (d) true SCA contours

Another disadvantage of D8 algorithm is that it produces unnatural parallel flow paths on planar surfaces (O'Callaghan&Mark, 1984; Wilson et al., 2008), because when there are several options (neighboring cells with the same „lowest" slope), the 1st one encountered clockwise from North is chosen arbitrary (O'Callaghan&Mark).

Lea (1992) also argues that D8 cannot be used for the construction of subcatchments boundaries neighther for the prediction of the location of principal streams. For example (*Figure 3.9.*), if we have a hillslope facing south-southeast, actual flow will proceed this direction as the steepest one, but D8 algorithm will route the water to southeast pixels. The larger the area of such hillslope, the greater the declination of D8 flow path from the actual one, leading to mismatches between modeled and actual watershed boundaries and stream locations (Lea, 1992). On a planar slopes, if aspect angle differs from strictly cardinal or diagonal, D8 gives underestimation of TCA (on *Figure 3.11.* TCA produced by D8 2 timeslower of the true TCA) (Costa-Cabral&Burges, 1994).

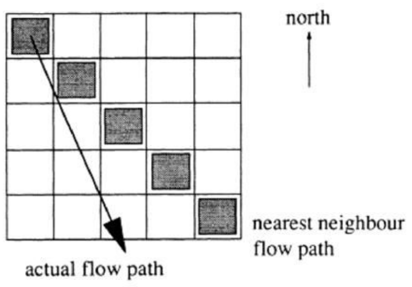


Figure 3-10(Lea, 1992). Error in the bearing of the SFD8 algorithm.

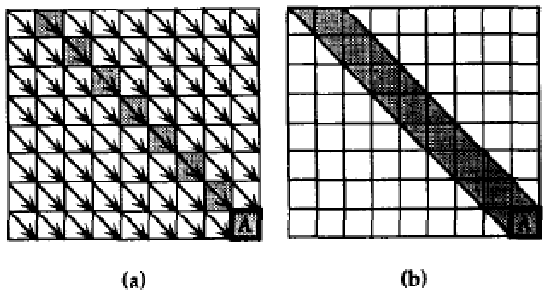


Figure 3-11.(Costa-Cabral&Burges, 1994). TCA of a pixel A on a planar surface with aspect angle 315º: (a) predicted by D8 and (b) true TCA

Quinn et al. (1991) add that with grid scale of 50 m and coarser, the single flow direction approach may give local inaccuracies, particulary on divergent hillslopes. Features tend to be sharper and give a ‚banding‘ effect (Quinn et al., 1991).Most of the drainage is concentrated in the channels, while large areas on the hills are with low flow accumulation values (Freeman, 1991).

Costa-Cabral and Burges (1994) showed that D8 produces large errors in any terrain topography (divergent, planar and even convergent), one of the reason of which is the nondimensional flow origin as a point source over a two-dimensional pixel and one-dimensional flow pathway as a line. D8 gives particularly large errors on hillslopes (e.g. SCAs computed by D8 on hillslopes on the Mettman Ridge DEM differ by 5 order magnitude from those predicted by DEMON) and, thus, is inappropriate for hillslope applications, including studies of hillslope hydrological response, channel head location, lndslide risk, soil water content and long-term basin evolution (Costa-Cabral&Burges, 1994).

D8 is not robust: even a tiny elevation difference between two neighboring cells have a large effect as oe of the cells recieves all the area (Seibert&McGlynn, 2007).

Positive features are the simplicity of the algorithm for calculation, which makes it one of the most widespread flow routing algoritnms for calculation of flow accumulation and upslope area, and its suitability for reflecting of realistic patterns of accumulating area in the valley bottoms, where it gives well-defined stream lines (Quinn et al., 1991; Freeman, 1991; Costa-Cabral&Burges, 1994). D8 is useful for extraction of river networks, longitudal profiles and basin boundaries (Gruber&Peckham, 2009).

3.4.2 Rho8

Fairfield and Leymarie (1991) (exCosta-Cabral&Burges, 1994; Bartak, 2009) developed the Rho8 (random eight-node) approach to overcome parallel paths on planar surfaces by D8. Rho4 is a variation which considers flow only to 4 cardinal neighbors. As in D8 water flows from one cell to one cell but with additional slope-weighted random factor: the steepest the slope is, the greatest probability of being selected.TCA, flow accumulation and SCA are calculated in the same way as in D8 metod Thus, Rho8 flow patterns more or less match those of D8, but appear to be more realistic. The greatest disadvantage of Rho8 approach lays in its stochastic nature, so each time Rho8 produces different flow patterns (Wilson et al., 2008). Another disatvantage is that on the plane slopesdue to wiggling of adjacent flow paths they tend to converge but can not diverge back, which leads to overestimation of TCA in some pixels and underestimation in others with the errors increasingdownslope (Costa-Caral&Burges, 1994).

3.4.3 MFD

Another algorithm for flow assignment is MFD8 (Multiple Flow Direction from 8 possibilities) or MS (Multiple Slope) (Freeman, 1989; Freeman, 1991; Quinn et al., 1991), also referred as the FD8 or simply MFD (Wilson et al., 2008), MD8 (Seibert&McGlynn, 2007). Actually, there are two forms of this algorithm, developed by Freeman (1989, 1991) and by Quinn with co-authors (1991), but both are governed by the same principle. It directs flow from a given cell to all its neighboring cells with the downhill slopes with the proportion of the flow according to the gradient of each downhill flow path, so that steeper gradients attract more of the accumulated area (Quinn et al., 1991) (*Figure 3.12.*).

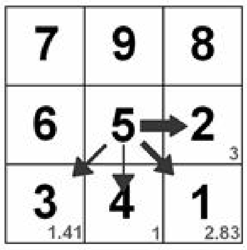


Figure 3-12(Bartak, 2009) The MFD8 algorithm. The thickness of the arrows symbolizes the proportion of flow falling on the particular neighbour.

Quinn (1991) version of flow partition:

$$\text{Ai} \;=\; \text{A} * (tan\beta i * Li) / \sum_{J=1}^{n} (tan\beta j * Lj)$$

where *n* is the total number of downhill slopes, Ai is amount of flow passing to the *i*th downhill neighbor, tanβi is slope to the *i*th downhill neighbor (difference in elevation/distance between the elevation values), Li is the contour length (orthogonal to the flow direction) of the *i*th direction (eighther cardinal L1=0,5*grid resolution or diagonal L2=0,354*grid resolution, chosen subjectively), A is he total upslope area accumulated in the current „central" cell, *j* is index for each downhill neighbor.

TCA of the cell of interest consists of partial contributions from many different cells (in formula above A is the TCA of „central" cell), as each cell recieves a fraction of the flow from each upslope cell. SCA is TCA acting through the contour length, which in that case equal to the sum of the downhill contour lengths:

$$\mathrm{SCA}=\mathrm{A}/\sum_{J=1}^{n}(Lj)$$

Local slope angle for a given cell is a weighted averege of all downhill slopes (not the maximum slope as in D8 approach):

$$tan\beta \ = \sum_{J=1}^{n}(tan\beta j * Lj) \, / \sum_{J=1}^{n}(Lj)$$

Thus, natural logarithmus of common term (C) in the formula (1) actually equal to the topographic index:

$$\ln C = \ln((A/\sum_{J=1}^{n}(tan\beta j * Lj)) = \ln(\mathrm{SCA}/\tan\beta)$$

It is obvious that MFD algorithm will produce different distributions of the topographic index than those produced by D8 (Quinn et al., 1991).

Freeman (1991) reports that from different methods for partitioning outflow the most satisfactory one is the method with the proportion of the flow according to the downhill slope values and special parametr *p* controlling the degree of slope convergence:

$$\frac{S_i^p}{\sum_{0<j\leq 8} S_j^p},$$

where Si is the slope from the cell to its i'th downhill neighbour, divided by the summation over the neighbours to which a positive slope exists.

The higher the values of parameter p, the more similar MFD8 algorithm to the SFD8 (theoretically if p →∞ then MFD8 turns into D8) (Holmgren, 1994; Bartak, 2009). On an artificial cone surface, the best p=1,1 (Freeman, 1991). Holmgren (1994) suggests p to be a compromise between 1 and ∞ or, in other word, between too smoothed and too quickly converging flow patterns. In his study he found optimal p = 4-6 and suggested that value of p should be increased with a higher resolution, approaching to SFD.As closer to channels, flow tend to be more concentrated and rather convergent, then divergent, Quinn et al. (1995) suggest to use p, which depends on the value of accumulated area:

$$p=\left(\frac{A}{CIT}+1\right)^{h},$$

where*p* is optional parameter,*A* is TCA and *CIT* is channel initiation threshold or theshold of accumulated area: cells having TCA values above CTI are indicated as water streams. Thus, the higher accumulation area, the more concentrated flow in a pixel, the higher value of p, the more restricted is flow dispersion introdused by MFD, which is situation in valley bottoms, channels. The lower a TCA, the closer p to 1, the more divergent flow is, which is ideal case on the upper part of relief (ridges and slopes). Kim a Lee (2004) developed this method for estimation of *p*, introducing additional parameters. More information can be found in the Diploma thesis of Vojtech Bartak (2008) and in the original articles.

TCA of the cell of interest consists of partial contributions from many different cells, as each cell recieves a fraction of the flow from each upslope cell.SCA is computed as the sum of the CAs from upslope cells divided by the cell width for the cardinal directions and by the cell width multiplied by √2 for the diagonal ones (Wilson et al., 2008).

In terms of elevation difference between neighboring cells, MFD8 is robust, because both cells recieve about the same portion of the accumulated area, not like in D8, where one cell recieve all the flow (Seibert&McGlynn, 2007).

Though the MFD algorithm produces realistic flow accumulation pattern on a divergent hillslopes, on convergent topography (valley bottoms, river channels) it tends to spread the flow wider than it is nessesary, particulary in case of coarse resolution (Freeman, 1991; Seibert&McGlynn, 2007). Flow pathways cross each other to a large degree, which becomes a problem for computation of the flw of the substances (solutes, sediments) (Seibert&McGlynn, 2007). So the basic recommendation is to use mixed algorithm, switching from MFD on divergent topography to SFD on convergent part of relief. However, on fine resolutionsuch mixed approach may give artifacts whenever SFD is working on smoothly differing but convergent part(Freeman, 1991). Costa-Cabral and Burges (1994) write that the most important limitations of MFD are discontiguity of computed contributing areas and dependency of the approximation quality on the geommetric symmetry and boundary proximity. The dispersion is incostistent with the physical definition of the upslope area (TCA) and SCA, so it should be minimized in TCA calculation for any multiple flow directing procedure.Another disatvantage of the algorithm is inefficient data storage, as eight possible direction will have to be recalculated each time they are needed for each pixel(Tarboton, 1997).

3.4.4 KRA

Lea's (1992) aspect-drivenkinematic routing approach (KRA) treats the cells of DEM as planar surfaces constructed on each cell using estimated elevations of its 4 corners. Flow moves across these surfaces as a point source (like a rolling ball) in the direction of the steepest slope or aspect angle, which is calculated as aspect vector in 1º increments (note that D8 increments in 45º as only 8 possible directions is allowed). The algorithm models the entry and exit points of flow on the perimeter of each pixel along the flow path. The TCA for a given cell is calculated as the number of flow paths passing through that cell multiplied by the grid cell area (Wilson et al., 2008).

Below in small letters there is an explanation why we can use aspect direction as the flow direction.

The equation of planar surface intersecting the origin in 3-D cartesian coordinate system is: $z = Ax + By$.The projection of the end of the unit path, beginning at the origin in the XY plane and described by the angle Θ, onto the plane is:$z = A\cos \Theta + B\sin \Theta$.

Water flows downhill at the steepest gradient, so the water path direction will have the same angle Θ, if the upper equation will be at minimum.We can rewrite this equation as:$z = A\cos\Theta + B\sin\Theta = R\cos(\Theta+\alpha)$.

Knowing from trigonometry that:$\cos(\alpha\pm\beta)=\cos\alpha\cos\beta\pm\sin\alpha\sin\beta$, we suppose $A = R\cos\alpha$ (1) and $B= - R\sin\alpha$ (2).Solving the system of equations (1) and (2), we derive $R=\sqrt{(A^2+B^2)}$ and $\alpha=\mathrm{arctg}(-A/B)$.As $R>0$, z is minimum (zero in our case, as it is a cosinus) when $\Theta+\alpha = \pi$.Thus, $\Theta = \pi - \alpha = \pi + \arctan(B/A)$.

From the triangle *abc* (*Figure 3.13.*): $\mathrm{tg}\alpha=b/a$; $c=\sqrt{(a^2+b^2)}$ (Pithagor's theorem) and $\cos\alpha=a/c=a/\sqrt{(a^2+b^2)}$. From its defenition: $\mathrm{tg}(\arctan(x))=x$, so if $\mathrm{tg}\alpha=b/a$, then $\arctan(b/a)=\alpha$. Thus, $\cos(\mathrm{arctg}(b/a))=\cos\alpha= a/\sqrt{(a^2+b^2)}$. Analogically, $\sin(\mathrm{arctg}(b/a))=\sin\alpha= b/\sqrt{(a^2+b^2)}$.

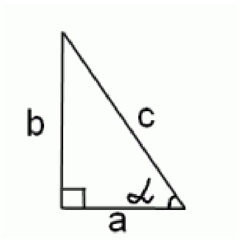


Figure 3-13. Triangle abc

Remembering that $\cos(\pi+\alpha)=-\cos\alpha$ and $\sin(\pi+\alpha)=-\sin\alpha$, the cartesian unit vector in the direction Θ on the plane XY ($\cos\Theta$, $\sin\Theta$) equals to $(-A/\sqrt{(A^2+B^2)}, -B/\sqrt{(A^2+B^2)})$. So for the plane defined by the equation $z = Ax + By$, the flow direction is defined by the vector (-A,-B).

From the other hand, we know that the aspect vector is a projection of the normal to the surface onto the plane XY, so it is 2-D vector of partial derivatives ($-\delta z/\delta x$, $-\delta z/\delta y$). From the equation of the plane surface $z = Ax + By$, we derive $A=\delta z/\delta x$ and $B=\delta z/\delta y$. So aspect vector is (-A, -B), so it is the same as the flow direction vector, which we derived in the previous paragraph.

Thus, to determine flow direction we need to determine only the aspect vector, which can be derived solely from the elevation data. For each cell the entry and the exit points of the flow on the perimeter is modeled, assuming water to move as a point in the direction of the aspect vector of the constructed one-cell plane.

The first task is to fit planes representing the surface of each pixel. Let the southwest corner of the pixel be at (0,0) in the XY plane, and the northeast corner has coordinates (1,1)(*Figure 3.14.*). Then the plane will be in the form: $z=\alpha x + \beta y + \gamma$. We neglect the height defined at the center point of the grid-based DEM and fit a plane to the co-ordinates of the four corners (instead of three points sufficient for the determination of a plane) by least squares method: $\alpha = ((z_{ne} - z_{nw}) + (z_{se} - z_{sw}))/2$ and $\beta = ((z_{nw} - z_{sw}) + (z_{ne} - z_{se}))/2$, where z_{ne}, z_{nw}, z_{se} and $z_{sw}$are the estimated heights of the North-East, North-West, South-East and South-West corners of the pixel respectively. The estimated elevation of each corner is calculated as the mean of the four surrounding spot heights (except for special case, when aspect vector cannot be defined, see below).

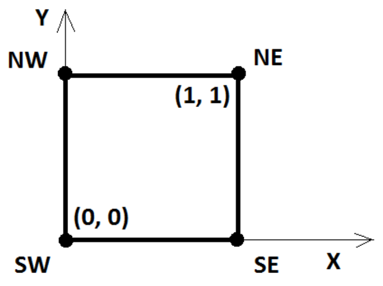


Figure 3-14. Grid cell with assigned coordinates for defining local plane (see text)

The aspect vector is simply (-α,-ß) or the angle $\theta = \pi + \arctan(ß/\alpha)$. If α and ß for some reason would be equal to zero, the aspect vector would not be defined, what is improbable in real terrain. Such situation can happen only if both pairs of opposite corners have equal elevations ($z_{ne}=z_{sw}$ and $z_{nw}=z_{se}$). To fix the problem recalculation of the estimated corner heights may be done by the expansion of the estimation. The mean of a three by three square of spot heights with its corner on the center pixel instead of four surrounding heights is used for the new estimate. If these estimates cause the same problem, a four by four square is used and so on (see *Figure 3.15.*). Thus, the aspect vector is derived for each pixel.

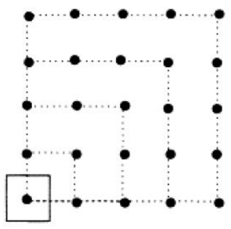


Figure 3-15(Lea, 1992). Iterative estimation of corner elevations

Second task is flow routing using information about aspect vector derived from the previous task. Firstly, flow originates at the center of the source pixel at the point (0.5, 0.5) and travels kinematically in a direction of the aspect θ as a point source and reaches the perimeter of the pixel in the outlet point with coordinate (x_0, y_0), which depend on aspect angle θ(*Figure 3.16.*).

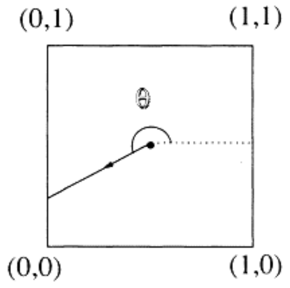


Figure 3-16(Lea, 1992) Initial flow path from the center point of a pixel.

Secondly, the outlet point *(x0,y0)* becomes an inlet point *(xi,yi)* on the perimeter of theneighboring pixel and is defined with respect to the coordinates of the new pixel. Again flow proceeds in the direction of the aspect vector of the current pixel till it meets next outlet point which will again become an inlet point for the next pixel and so on.

If the slopes of the two pixels face each other forming a valley, inlet edge forms the valley floor and flow is routed along the inlet edge to neighboring pixels.This happens when the aspect has a positive component in the direction of the inlet edgeand a full crossing is not possible. Flow is routed to the edge adjacent to the inlet inthe direction with positive aspect component(*Figure 3.17.*).

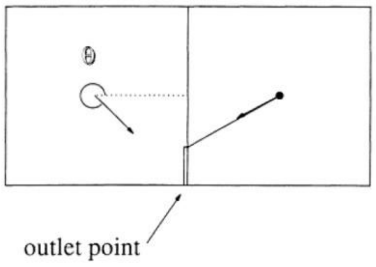


Figure 3-17(Lea, 1992). The position of the outlet point, in case when full crossing is not possible

Flow paths are built until the catchment outlet or a hollow (groups of pixels whose aspects prevent flow from escaping) is reached (*Figure 3.18.*).

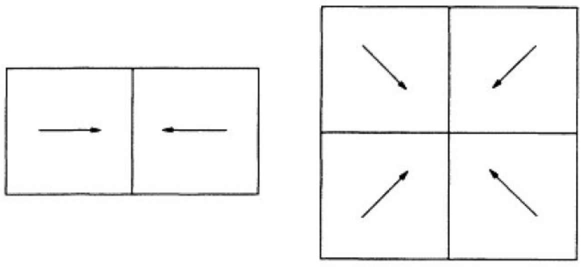


Figure 3-18 (Lea, 1992). Aspect vectors that represent typical hollows.

The main advantage of KRA, according to Tarboton (1997), is that there is no dispersion of flow and flow goes continuously as an angle between 0 and 180º. From the other hand, Costa-Cabral and Burges (1994) states that Lea's approach has the same limitation as D8 – the one-dimensional representation of a flow. Another limitation is poor robustness due to inconsistent and conterintuitive flow directions in ‚special cases' appearing from local fitting of planes over each pixel (e.g. the influence of much higher neighbors on downslope flow) (Tarboton, 1997).

3.4.5 DEMON

Digital elevation model networks (DEMON) (Costa-Cabral&Burges, 1994) represents flow in two dimensions (2-D) flow strip (which authors call flow tubes) and directs it by local aspect angle (in a similar manner as used by Lea (1992)), offering the main advantage of contour-based models – the representation of varying flow width over nonplanar topography. Width of „flow tube" is a function of local topography: it increases over divergent, decreases over convergent topography and remains constant over plane surfaces.

TCA of a pixel is defined as the plan-view area of the collection of all points located topographically upstream from te pixel. From its definition *SCA = TCA/w*, where *w* in case of DEMON is the total flow width orthogonal to the flow direction along the portion of the pixel boundary through which flow exits pixel. *w* is dependent on the flow direction angle α by the relation: $w = |sin\alpha| * x +$

$|cos\alpha|*y$, where x and y are pixel width along the respective axes. Thus, SCA is determined by the flow field (flow direction angle field), which can be obtained from the elevation field. As grid-based DEM is a matrix of discrete points, some surface-fitting method is needed to construct elevation surfaces on each pixel (elevation field). DEMON approximates the surface of each pixel by a best fit plane using local interpolation, creating discontinuous planar mosaic. For each pixel plane aspect angle and accordingly flow width can be calculated.

Next step is calculation of TCA and SCA, which can be calculated using 2 algorithms implemented in DEMON– downslope and upslope, based on the matrix of flow angles. Downslope algorithm gives opportunity to simultaneously calculate the specific dispersal area (SDA), area of the terrain that drains flow from the contour segment per unit contour, while DEMON-upslope computes SCA matrix faster. DEMON-downslope tracks flow downslope, while DEMON-upslope traces the boundary of a pixel's contributing area and calculates the area enclosed by that line. Both algorithms produce the same SCA values.

DEMON-downslope algorithm suggests that the total flow volume drained by any given pixel equals to the pixel's TCA. Each pixel at a time is treated as a source pixel of a unit flow depth and its influence matrix on all the rest DEM pixels is constructed, i.e. matrix of the flow volume parts from the source pixel which are drained by each pixel in DEM. The TCA matrix is calculated by succesive addition of the all influence matrices of every DEM pixel. The SCA matrix is obtained by division of TCA matrix by flow fidth matrix, derived in its turn from aspect angle matrix.

Algorithm used for the construction of the influence matrix for one pixel can follow only one flow path from one single pixel to the next single pixel at a time, thus the total dispersal area of a pixel is divided into several flow tubes, which represent these flowpaths from one cell to one cell and so on (*Figure 3.19.*). Flow tubes may converge becoming more narrow and finally converting from 2-D strip into 1-D line.

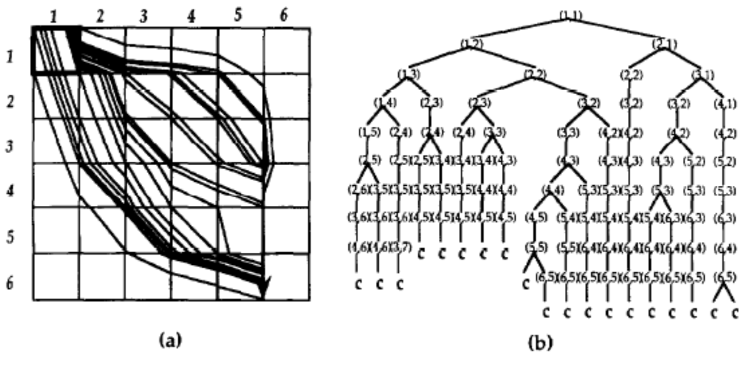


Figure 3-19(Costa-Cabral&Burges, 1994). (a) Flow tubes from pixel (1,1); (b) Schematic representation of the same flow tubes. Branching corresponds to flow being split between two pixels

Flow partioning between neighboring pixels are always in cardinal directions and can be calculated using aspect angle and grid spacing values. For example, on *Figure 3.20.* flow is splitted between eastern and southern neighbors. Eastern neighbor part is simply shaded triangle area divided by the total pixel area, northern neighbor flow part is 1 minus part of eastern neighbor.

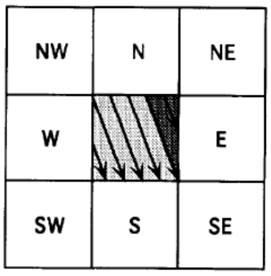
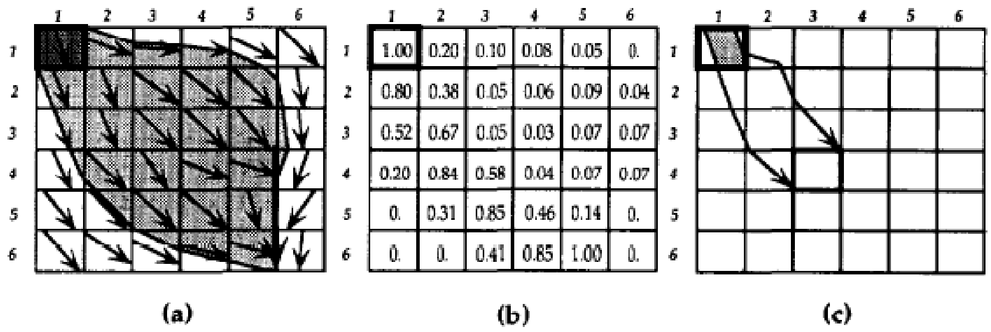


Figure 3-20.(Costa-Cabral&Burges, 1994). Flow partitioning from a source central pixel to its eastern and southern neighbors

The example of the resulting influence matrix for one pixel (1,1) is shown on *Figure 3.21.* along with the aspect direction matrix and the TDA of the same pixel (1,1).



	1	2	3	4	5	6
1	1.00	0.20	0.10	0.08	0.05	0.
2	0.80	0.38	0.05	0.06	0.09	0.04
3	0.52	0.67	0.05	0.03	0.07	0.07
4	0.20	0.84	0.58	0.04	0.07	0.07
5	0.	0.31	0.85	0.46	0.14	0.
6	0.	0.	0.41	0.85	1.00	0.

Figure 3-21.(Costa-Cabral&Burges, 1994). (a) Aspect angles matrix (arrows) and TDA of the pixel (1,1) as a shaded area; (b) Influence matrix of pixel (1,1). Each value represents a fraction of the area from the source-pixel (1,1) that is drained by a DEM pixel; (c)Physical meaning of the value o,58 from of pixel (4,3) in (b)

DEMON-upslope traces the boundary lines starting at two of its ends at the pixel corners upslope till they meet and form enclosed boundary, allowing calculation of TCA (*Figure 3.22.*).

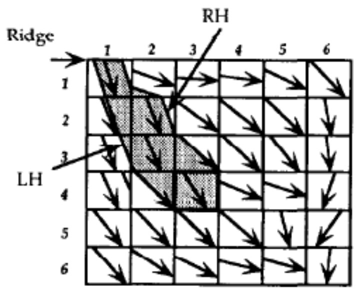


Figure 3-22.(Costa-Cabral&Burges, 1994). DEMON-upslope algorithm for TCA calculation (shaded area) by tracing of the right-hand (RH) and left-hand (LH) sides of flow lines representing TCA boundaries.

TCA pedicted by DEMON on cone surface are close to the true ones, though some pixels are overestimated(pixel B in *Figure 3.23 (a)*). It can be seen particularly on SCA contours, which being mostly circular, however have indentations to the north, south, east and west(*Figure 3.23.*).

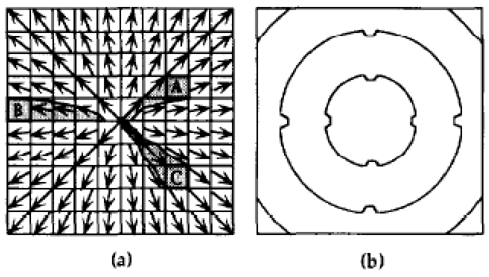


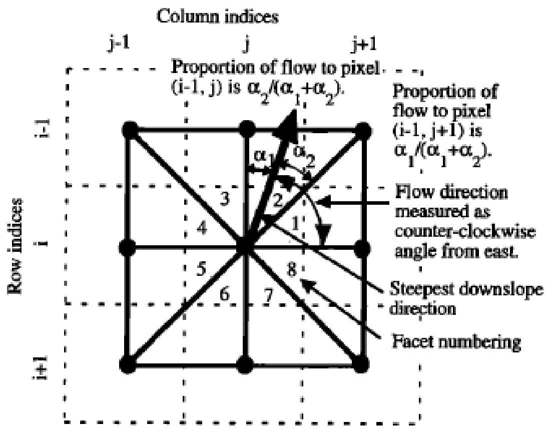
Figure 3-23 (Costa-Cabral&Burges, 1994). Flow patterns predicted by DEMON on a right circular cone: (a) TCA of pixels A, B and C; (b) SCA contours.

These errors are explained by the fact that DEMON approximates conical surface by mosaic of plane surfaces, which generates bias towards N-S and E-W directions. The identations can be avoided only if a curved rather than a planar surfaces would be fitted to each pixel, but the computational expense would be much higher (Costa-Cabral&Burges, 1994). As in Lea's KRA approach, the approximation by fitting a plane in DEMON introduces too greate influence of higher neighbors on downslope flow (Tarboton, 1997), decreasing the robustness of the algorithm.

Thus, producing errors on some terrain forms, DEMON though has a great advantage over other flow routing algorithms: being grid-based it is able to present different terrain topography by flow path width, which had been possible till present only by contour-based models. Another advantage is that algorithm may calculate simultaneously SCA and SDA. The particle-tracking approach of the DEMON-downslope algorithm allows to use it for surface sediment and pollutant tracking. It also allows to distinguish between dispersed (2-D flow tubes) and concentrated (1-D flow line from converged flow tubes) flow (Costa-Cabral&Burges, 1994).

3.4.6 3.4.6. D∞

Tarboton (1997) introduced another flow routing method D∞ (D-infinity), which directs flow in an infinite number of possible single flow directions between 0 and 2π according to the steepest downward slope on the eight triangular facets constructed in a 3x3 pixel window centered on the pixel of interest.The flow is apportioned from a pixel to maximum two downslope pixels according to how close flow angle is to the direct angle to that pixel corner. (*Figure 3.24.*).



Column indices
j-1
j
j+1
Proportion of flow to pixel (i-1, j) is $\alpha_2/(\alpha_1+\alpha_2)$.
Proportion of flow to pixel (i-1, j+1) is $\alpha_1/(\alpha_1+\alpha_2)$.
Flow direction measured as counter-clockwise angle from east.
Steepest downslope direction
Facet numbering
Row indices
i-1
i
i+1
α_1
α_2
1
2
3
4
5
6
7
8

Figure 3-24(Tarboton, 1997). Flow direction defined as the steepest downward slope on planar triangular facets on a block-centered grid.

Downslope vector in each of the eight facets is defined. The flow direction is the steepest downslope vector from all eight facets. In case of facets with equal steepest slope D-inf picks the first one in order of facets from 1 to 8 shown in the *Figure 3.24.* This bias on natural terrain seems to be negligible, as such equal-slope situation is rare (Tarboton, 1997).

Comparison of TCAs produced by D-inf with the theoretical ones revealed D-inf to produce relatively small mean square errors on outward and inward cones and a plane. Analysis of influence map of several chosen pixels (flow paths of these pixels) on a circular cone though revealed D-inf to result for pixels with flow direction aligned with grid diagonal to non-spreading flow path, same as produced by D8, thus introducing bias (Tarboton, 1997).

3.4.7 MD∞

Seibert and McGlynn (2007) developed triangular multiple flow direction algorithm (MD∞), based on Tarboton's D infinity approach (1997), but allowing multiple flow direction.As in Tarboton's D-inf, firstly, 8 triangular facets is constructed around center of a pixel, the steepest gradient is computed on each of the 8 planes – so called locally steepest directions. The flow is weighted and proportioned to all downslope directions on the basis of the gradients, as in Freeman's MFD approach (1991) with parameter p, allowing partially to control dispersion (*Figure 3.25.*). Thus, proposed algorithm overcomes the problem of overdispersion of MFD method and restriction of 1 direction flow of D-inf approach (Seibert&McGlynn, 2007).

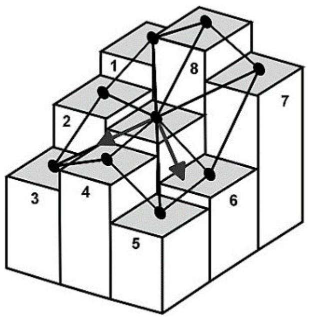


Figure 3-25.(Seibert&McGlynn, 2007). Example illustrating determination of flow directions by MD∞.

Other flow routing approaches exist (e.g. ANSWERS (Beasley&Huggins, 1978);vector-grid approach (Mitasova and Hofierka, 1993; Mitasova et al., 1995, 1996); form based algorithm (Pilesjo et al., 1998), path-based method (Orlandini et al., 2003); Mass-Flux Method (MFM) (Gruber&Peckham, 2009)), but we will not discuss them in present work.

4 Methodology

4.1 General course of action

First step was to learn basics of programming in language Python in general and for ArcGIS, which was done by performing tasks within a scope of one-semester course held in Czech University of Life Science in Prague "Geoinformaticke aplikace".

Next step was to learn how to work with raster data types in Python: how to convert rasters to an array data structure, how to access values in the array and so on. It was necessary to get accustomed to binary heap data structure and its application in work with raster data sets.

After that, a chosen flow routing algorithm had to be implemented in ArcGIS by writing Python scripts and integrating it into newly created Toolbox.

Last step was to use newly implemented flow direction algorithm on hydrologically correct DEM of real terrain and compare the results of its implementation with the results derived by using built-in ArcGIS tools.

4.2 Used Software

4.2.1 ArcGIS 10.3

ArcGIS is a geographic information system (GIS) for working with maps and geographic information developed by the company ESRI. ArcGIS is used for creating, editing and analyzing maps and geodatabases.

ArcGIS provides the opportunity to access geoprocessing tools through the integrated Python language. Python is incorporated into the automatic installation of ArcMap and ArcGIS for Server in each version of ArcGIS since 9.0. Python version 2.7.x is used in ArcGIS starting from version 10.1 up to the latest 10.4 ("What version of Python is used in ArcGIS?", 2016). Using Python scripts one can create user-friendly geoprocessing tools in ArcGIS.

ArcGIS is widespread for its simplicity, functionality and the breadth of options, solid documentation, and support. Integrated Python and ModelBuilder give ability to create user's own toolboxes. ArcGIS has extensive labelling features, symbol creation, and many features of a vector graphics software to generate basic layouts of the maps, that can be exported to different formats, such as PDF and SVG, to be edited outside the boundaries of GIS scope. Another important aspect of the ArcGIS popularity is that huge numbers of companies, government departments and educational institutions that use ESRI software. For example, students at qualifying institutions may be eligible for a 12-month license of ArcGIS for Desktop at no cost as part of the Esri Education Site License Program ("Esri Software for GIS Students", 2016).

4.2.2 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language with a highly readable simple syntax, which is a great advantage for a person new to programming. Its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.

Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Being free and open source, Python provides large number of libraries.

For the current diploma thesis Python 2.7 was used, as this version is integrated within ArcGIS 10.3. Besides, Python 2.7.x has such advantage in comparison with Python 3.x as wide number of supported non-standard libraries.

More information and all documentation can be found at official Python Programming Language homepage https://www.python.org .

4.3 Choosing of tasks and used algorithms

ArcGIS Hydrology toolset is based on SFD8 algorithm of flow routing (O'Callaghan&Mark, 1984). Terain Analysis Using Digital Elevation Models (TauDEM) (Utah State Unversity, 2016) is ArcGIS Toolbox which extends basic ArcGIS toolset with tools for determination of flow direction and contributing area using D-infinity approach (Tarboton, 1997). So far, there is no more publically available ArcGIS toolset that uses other flow routing algorithms.

For determination of flow directions and for flow simulation (TCA calculation) MFD (Freeman, 1991) was chosen as basic algorithm permitting multiple flow direction. Additional script was written for evaluation of flow dispersion extent. It is possible to use the script after for evaluation of dispersion extent of flow direction algorithms, which may be added in the future.

Tool for the creation of Influence maps by SFD8 and MFD8 with different p parameter was created to follow the flow paths and extent of dispersion from individual source pixels. Supplementary tool was created to find out coordinates of the chosen source pixels in terms of number of row and column for the individual pixel.

Tool for the calculation of contour lengths by three different methods and user-specified parameters for MFD algorithm was created. Resulting contour length rasters were used for the calculation of SCAs. All obtained data (TCAs, SCAs for different algorithms and parameters) were analyzed in Excel, preliminary extracted using ArcGIS built-in functions.

Basis of tools creation are scripts written by Novak (2015) for the determination of SFD8 flow direction and flow simulation using binary heap and 3x3 moving window.

Created tools may be added to Toolbox „DiTerAnT“ (Novak, 2015) extending its functionality.

4.3.1.1 ArcPy

ArcPy is a package of Python language, which contains all ArcGIS functions and gives opportunity to use them directly in Python scripts.

ArcPy was used to convert input raster data into NumPy array data structure, from which elevation values could be accessed and all calculations could be performed, and back from array to raster.

4.3.1.2 NumPy

NumPy (Numerical Python) is the fundamental package for scientific computing with Python. NumPy's main object is a powerful homogeneous multidimensional array (NumPy array, ndarray, N-dimensional array). One of the most important properties is that the elements in an ndarray can be accessed using indexing facilities.

The most important attributes of an ndarray object are:

ndarray.ndim

the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as *rank*: 0 for a scalar (dimensionless), 1 for a vector and 2 for matrices.

ndarray.shape

the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with *n*rows and *m* columns, shape will be (n,m). The length of the shape tuple is therefore the rank, or number of dimensions, ndim.

ndarray.size

the total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

an object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

More information can be found on: https://www.scipy.org , http://www.numpy.org .

4.3.2 Determination of flow directions – MFD

Iput DEM is converted to ndarray or matrix of elevations of terrain. Every "cell" of the array is analyzed row by row and for each "cell" 8 of its neighbors are defined (so-called 3x3 moving window). From differences in elevations and distances between cell centers slopes from "center" cell to each of its neighbors are calculated. As tool is working with regular grid DEM, distances for cardinal neighbors are taken as DEM resolution (or cell width) and for diagonal neighbors as cell width multiplied by √2.

The directions of flow from each cell is stored as a code number in that cell.

Flow direction coding is based on standard codes implemented in *ArcGIS*(*Figure 4.1.*) and represents values 2^i, where i is the index of the neighbor. Indexation starts from 0 for the Eastern neighbor clockwise to 7 for N-E neighbor (*Figure 4.2.*).

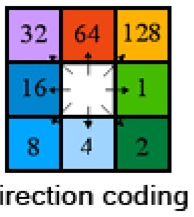


Figure 4-1(Resources.arcgis.com, 2015). Flow direction coding used for single flow direction in ArcGIS.

Thus, each number has a single 1 bit, with the rest of the bits being set to zero (*Table 4.1.*). It makes it possible to store more than one flow direction from each cell. Because each of the eight codes only uses one bit, however many of them are added together, the new number will be unique (Wise, 2008). For example, $16 + 2 = 18 = 2^4 + 2^1$ is the same as 00010000 + 00000010 = 00010010. Therefore, from the resulting sum we can always extract the positions of ones (counting from the end from 0 to 7), which is the exponent i in direction code (2^i).

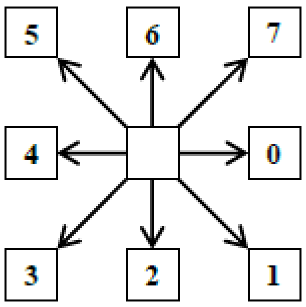


Figure 4-2. Neighbors indexing

Table 4-1 (Wise, 2008). Flow directions codes in decimal and binary

Flow Direction Code Decimal	Flow Direction Code 8 Bit Binary
1	00000001
2	00000010
4	00000100
8	00001000
16	00010000
32	00100000
64	01000000
128	10000000

Flow direction, thus, is stored in a cell as a sum of direction codes for all neighbors with positive slopes. If no positive slope is found, error message appears that there is a drainless area in DEM.

4.3.2.1 Evaluation of flow dispersion introduced by MFD approach

Flow dispersion can be evaluated by distribution of number of downslope cells receiving flow. Input flow direction raster of unique codes as defined above is converted into ndarray, for each cell of which number of non-zero bits is calculated and converted to output 'flow dispersion' raster. This raster can be subsequently analyzed visually and by construction of histograms of the number of cells that receive accumulated area from one cell.

4.3.2.2 MFD code calculator

As totally for MFD we have 255 codes, representing sum of codes for 8 basic direction, additional script was written to return all possible flow directions from the input MFD code numbers. Input codes should be pasted into array *MFDcodes* in the script. For each element of the array, its binary representation is obtained as a string and the index of encountered value „1" from the end of the string is used to define directions (see *4.3.2. Determination of flow directions*for explanation of principal of MFD codes).

4.3.3 Flow simulation

Flow simulation (or contributing area calculation) in case of MFD does not require flow direction raster as input. Firstly, input DEM is converted to ndarray (DEM array). Ndarray of ones is created as an initial array of values for accumulated area (CA array). If user gives as an input his own weights for flow accumulation, CA arrays will be sum of ones and user-specified weights (which are converted from weight raster to ndarray as well).Then, using Binary Heap, DEM array values are sorted.

Starting with the value of the highest elevation, for each cell of DEM ndarray its 8 neighbors are found and slopes to them are calculated as in *4.3.2*. Positive slopes are summated. Then to values of flow accumulation in CA array for each neighbor with a positive slope apportioned flow from a current "center" cell is added. Thus, flow accumulation of a positive slope neighbor = its current value + value of flow accumulation of "central" currently processed cell *multiple by proportion:

$$\frac{S_i^p}{\sum_{0<j\leq 8} S_j^p},$$

where *Si* is the slope from the "central" cell to its *i*'th downhill (positive slope) neighbor, divided by the summation over the neighbors to which a positive slope exists. Parameter *p* may be specified by user or *p=1* is used as a default value.

Output flow accumulation array in units of cell numbers is then converted to raster and can be used for calculating TCA in units of DEM resolution through multiplying by area of one pixel (for square-grid DEM equals to square of cell width) using ArcGIS *Math* tools. Next step is to use ArcGIS *Math* tools get ouput log-transformed for better visualization of flow accumulation.

4.3.3.1 Peucker and Douglas weights

Additional script was written to derive raster of weights, representing local curvature, according to Peucker & Douglas (1975)from input DEM raster, which can be used as user-specified weights raster for the calculation of flow accumulation.

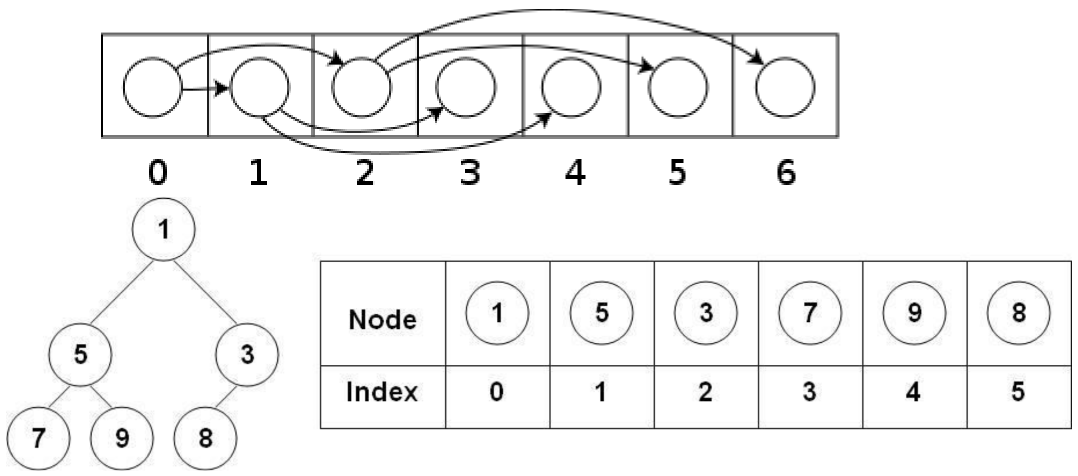
The basic principal is comparison of cell elevations in moving 2x2 window. First, input DEM is converted to ndarray and initial weight ndarray of ones of the same shape and size is created. Then each cell ([x, y]) is compared to its 3 neighbors ([x, y+1], [x+1, y+1] and [x+1, y]). The value of the cell with maximum elevation will be changed from 1 to 0 in weight ndarray of ones. After checking all DEM cells, values 1 will remain only for cells, which did not have maximum elevations in the moving windows, that are cells of potential water streams. Resulting ndarray of weights is converted and saved in raster form, and can be used as an input of user-specified weights in *Flow accumulation* tool.

4.3.3.2 Binary Heap

Calculation of flow accumulation must start with the highest cells to ensure that all visited cells, which will increase Contributing Area to the direction of their outlet, have already received accumulation from all its "tributaries". Instead of using *sort* command, Binary Heap sorting (Töpfer, 2007 from Novak, 2015) was implemented. We used the script for Binary Heap that had already been written by Novak, 2015.

The great advantage of using this type of sorting is low computational complexity, because data are represented as compleate binary tree compactly stored in an array(*Figure 4.3.*), each element of which can be easily accessed in terms of parent and children elements by using indices. If the tree root is at index 0, with valid indices 0 through $n - 1$, where n is the number of elements in the heap, then each element a at index i has children at indices $2i + 1$ and $2i + 2$ and its parent *floor* at $((i - 1) / 2)$. Sorting is performed by comparison element with elements of only one branch, not the entire data set.

.



Node	1	5	3	7	9	8
Index	0	1	2	3	4	5

Figure 4-3("Binary heap", 2016). Scheme of how binary tree is stored in array

After sorting of DEM elevation values in the heap, the item from the top of the heap (with maximum elevation value in our case) is always removed, and its flow accumulation value is apportioned to all downslope neighbors according to their slopes. After top item was removed from the heap, the heap structure is supported back by resorting the elements of the heap: removed top element ("the root" of the max-heap) is replaced with the last element on the last level (with minimum elevation), which is repeatedly compared to its children and is swapped with them, till correct heap order not returned (*Figure 4.4.*). Then procedure is repeated: new top heap item is removed and used for calculation of the contributing area, heap is restructured back with new top item with the next maximum elevation values and so on. It is so-called *priority queueing*, in which each item is assigned a priority, and the item with the highest priority goes "out" first. By this mean, contributing area of all DEM cells are calculated.

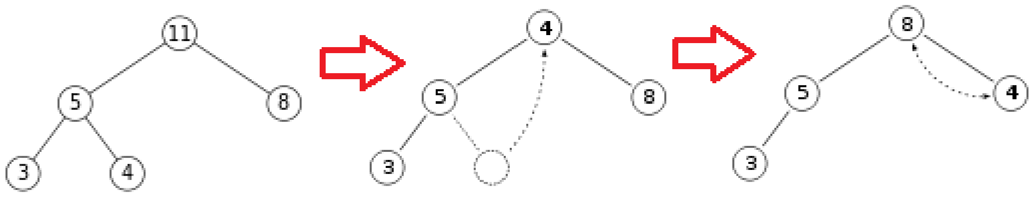


Figure 4-4. Tree representation of the maximum element extraction from a max-heap

4.3.3.3 Influence Map

Tool *Influence Map* maps the flow rout from individual pixels and the extent of flow dispersion. Two flow direction algorithms are implemented: SFD8 (O'Callaghan&Marks, 1984) and MFD (Freeman, 1991) with user-specified parameter p. The principal of algorithm calculation is described above (*4.3.3. Flow Simulation*, *4.3.3.1. Binary Heap*), the only difference is that initial flow accumulation ndarray contains zeroes in each pixel, except user-specified pixels, to which 1 values of flow accumulation are assigned. In case of SFD8 algorithm, flow is routed to only one neighbor with the maximum positive slope.

Output Influence maps derived by different flow routing algorithms (or with using different values of parameter p) will serve to show the differences in flow routing models, or for the modeling of potential flow paths of contaminants.

As *Influence map tool* requires coordinates of source pixels of DEM in terms of number of column (x) and row (y), additional tool was written to give rasters of column and row numbers of the input raster. *Column/Row Grid tool* accept input raster and converts it into ndarray. Then ndarray of column and row numbers of the same size and shape as input raster is created using NumPy built-in method *.indices(shape)*.

Source pixels are chosen at topographically different locations (with high and low DEM values, presumably on divergent slopes and close to the stream channels according to DEM raster) with several flow directions (4-8) determined by MFD by comparison of raster of *Flow dispersion* (see *4.3.2.1.Evaluation of flow dispersion*). Multi point shapefile of 7 chosen points were created, basic characteristics from analyzed rasters (number of column, number of row of pixels where points are located, number of flow directions, TCA calculated by MFD and by SFD algorithms) were added to the attribute table of point shapefile by *Extract Multi Values to Points tool* of ArcGIS *Spatial Analyst toolset*. Attribute table was exported to Excel table by *Table to Excel tool* of ArcGIS *Conversion tools*.

4.3.4 SCA calculation

SCA is TCA divided by contour length. What is contour length in case of 1D (treating flow as a point source and flow path as a line) SFD and MFD algorithms is an open question. Chirico (2005) reports that according to the suggested in literature flow width (effective contour length) values SCA even for the same flow routing algorithm may differ 4 times.

Quinn (1991) had chosen contour length somehow arbitrary, being a sum of contour length orthogonal to the flow directions towards all neighbors with positive slopes (0,5*grid resolution for cardinal and 0,354*grid resolution for diagonal neighbor). SCA in Freeman's MFD is computed as the sum of the CAs from upslope cells divided by the cell width for the cardinal directions and by the cell width multiplied by √2 for the diagonal ones (Wilson et al., 2008), but it is not clearly stated, if sum of flow widths to all downslope neighbors is used. Freeman's definition with my understanding of it: SCA=TCA/$\sum_{j=1}^{n}(Lj)$, where Lj is contour length towards a neighbor with a positive slope (Lj=grid size for cardinal neighbors and Lj=grid size*√2 for diagonal neighbors). Wolock and McCabe (1995) use different factors for cardinal neighbors (0,6*grid size) and for diagonal neighbors (0,4*grid size), explaining their choice by the fact that the maximum value of total contour length (the case of eight neighboring downslope cells) would equal the total boundary length between the cell of interest and all of its neighboring downslope cells. Anyway, each cell of DEM have different flow width and additional script for its calculation was written with the option to choose Freeman's, Quinn's, Wolock&McCabe's approach or specify the multiplication factors by user.

Firstly, input flow direction raster is converted to ndarray. For each cell of flow directions ndarray sum of contour length are calculated and converted to output raster.

Then TCA raster is divided by derived contour length raster using ArcGIS *Math tools*.

The same script can be applied for the calculation of flow widths in case of D8 (assuming contour length simply to be grid size in cardinal directions and grid size multiplied by √2 for the diagonals – which is marked as Freeman's approach in created tool), though we did not use it in present work. Calculation of SCA for D8 is performed by division of TCA derived using D8 algorithm simply by grid size using ArcGIS *Math Tools*, as invariant flow width is supposed to be the best approach for SCA calculation by D8 algorithm (Chirico et al., 2005).

4.3.5 Data analysis

Basic descriptive raster statistics (*Histograms* of distributions, *Mean*, *Maximum*, *Minimum*, *Sum*, *Standard deviation*) was obtained using ArcGIS built-in option to display properties of a layer through the section *Symbology*, subsection *Classification* of values.

Scatter plots of SCAs and TCAs derived by different methods were constructed in *Excel* by comparison of respective values extracted from TCA and SCA rasters. Firstly, 50 000 random points were created (by built-in ArcGIS *Create Random Points tool* of *Data Management toolset*). Points are not located in the center of the grid cell, but no interpolation techniques were used for subsequent data extraction, so each point is associated with the value in the center of the pixel where point is located. Some of these points belong to the same pixels.

Secondly, rasters values were extracted to attribute table of the randomly created points (by *Extract Multi Values to Point Tool* of *Spatial Analyst toolset*). Thirdly, multipoint attribute table was converted to *Excel* table (by *Table to Excel tool* of *Conversion tools toolset*).

For better visualization of resulting rasters logarithmic scaling (obtained by *Ln tool* of ArcGIS *Math tools*), hillshade background (derived by *Hillshade tool* of ArcGIS *Spatial Analyst toolset*) and 25 m contour lines derived from DEM (by *Contour tool* of ArcGIS *Spatial Analyst toolset*) were used. All visual pairwise comparisons were performed using the same color scaling.

5 Results

5.1 Toolbox description

The result of the diploma thesis is a ArcGIS toolbox „MFD“ (stands for Multiple Flow Direction) of 7 tools for hydrological terrain analysis with Multiple Flow Direction algorithm (Freeman, 1991) (*Figure 5.1.*).

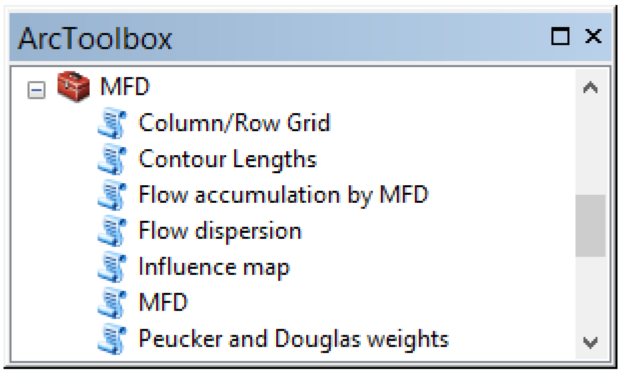


Figure 5-1 MFD toolbox

The structure of the MFD toolbox is presented on *Figure 5.2*.

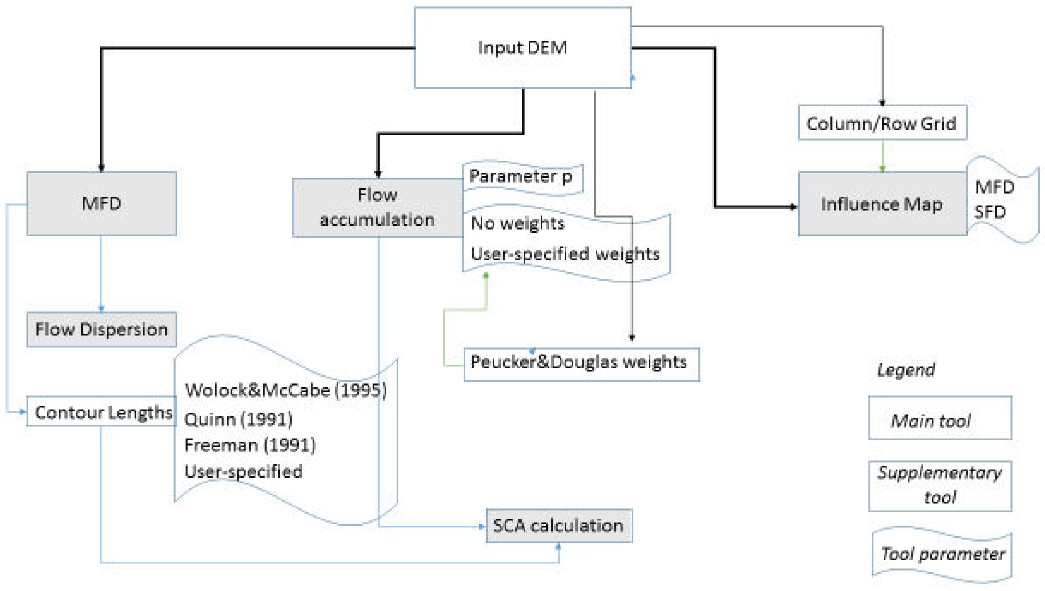
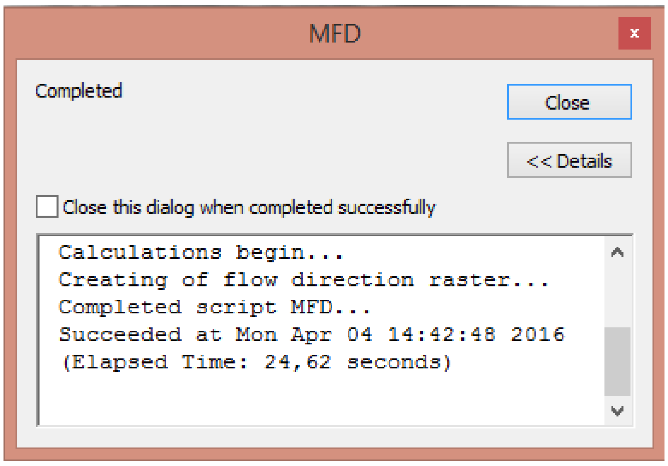


Figure 5-2. Structure of MFD toolbox

There are several tasks, which can be performed with MFD Toolbox. Main tools uses input DEM raster to calculate Multiple Flow Direction (*MFD tool*), flow accumulation (*Flow accumulation tool*) and Influence Map of required pixels. Another main tool is *Flow Dispersion tool*, which calculates the number of flow directions in each pixel using as input raster of flow directions, derived by *MFD tool*.

There are number of supplementary tools, results of implementation of which are used as a predictor or a parameter of another tool. For example, *Column/Row Grid tool* is used to find out coordinates of wanted pixels in terms of row and column number of a given raster, to use these coorinates after as an input of *Influence map tool*. Another supplementary tool is *Peucker&Douglas weights tool*, which calculates weight raster according to elevation from input DEM. The resulted weights raster can be optionally used as a user-specified raster of weights in *Flow Accumulationtool*. *Contour Lengths tool*, which uses as an input raster of flow directions, may be considered as supplementary tool as well, because resulting raster of contour widths is used together with flow accumulation raster to derive specific catchment area.

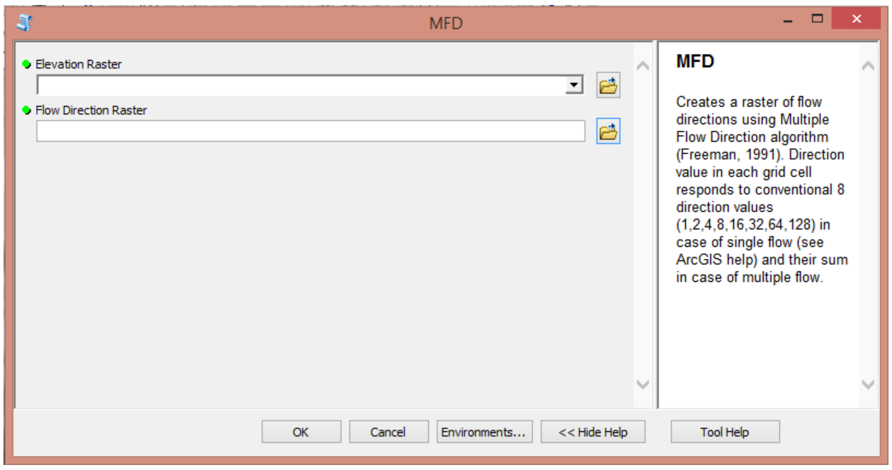
All tools are supplied with metadata with description of each parameter, so user can easily understand, how to use the tool. Each tool also is provided with informative messages, which appear in a window during calculation process (*Figure 5.3.*).



```
Calculations begin...
Creating of flow direction raster...
Completed script MFD...
Succeeded at Mon Apr 04 14:42:48 2016
(Elapsed Time: 24,62 seconds)
```

Figure 5-3. ArcMessage window

MFD (Multiple Flow Direction) tool (*Figure 5.4.*) has input Elevation raster or DEM and output will be raster of codes for multiple flow direction. No other parameters are required.

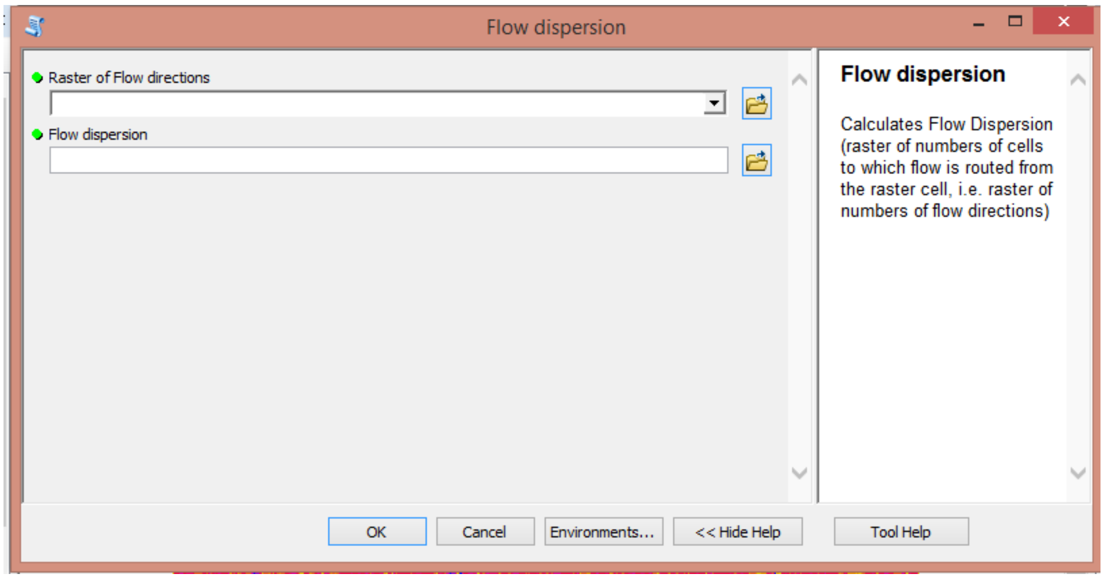


MFD

Creates a raster of flow directions using Multiple Flow Direction algorithm (Freeman, 1991). Direction value in each grid cell responds to conventional 8 direction values (1,2,4,8,16,32,64,128) in case of single flow (see ArcGIS help) and their sum in case of multiple flow.

Figure 5-4. Dialog window of MFD tool

Flow dispersion tool uses direction raster, derived from a previous step, as an input and gives output raster of numbers of flow directions in each pixel (*Figure 5.5.*).

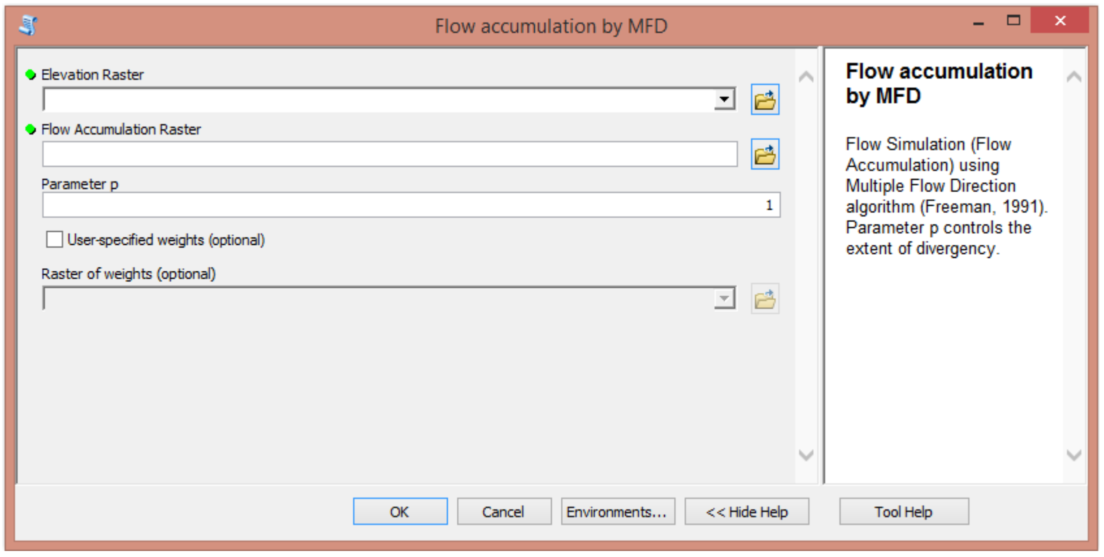


Flow dispersion

Calculates Flow Dispersion (raster of numbers of cells to which flow is routed from the raster cell, i.e. raster of numbers of flow directions)

Figure 5-5. Dialog window of Flow Dispersion tool

Flow accumulation tool (*Figure 5.6.*) calculates contributing area in pixels from the input DEM by MFD algorithm. User can partly control flow divergency changing parameter p, default value of which is set to 1. Additional option is to add own raster of weights, otherwise default raster of ones-weights will be used.



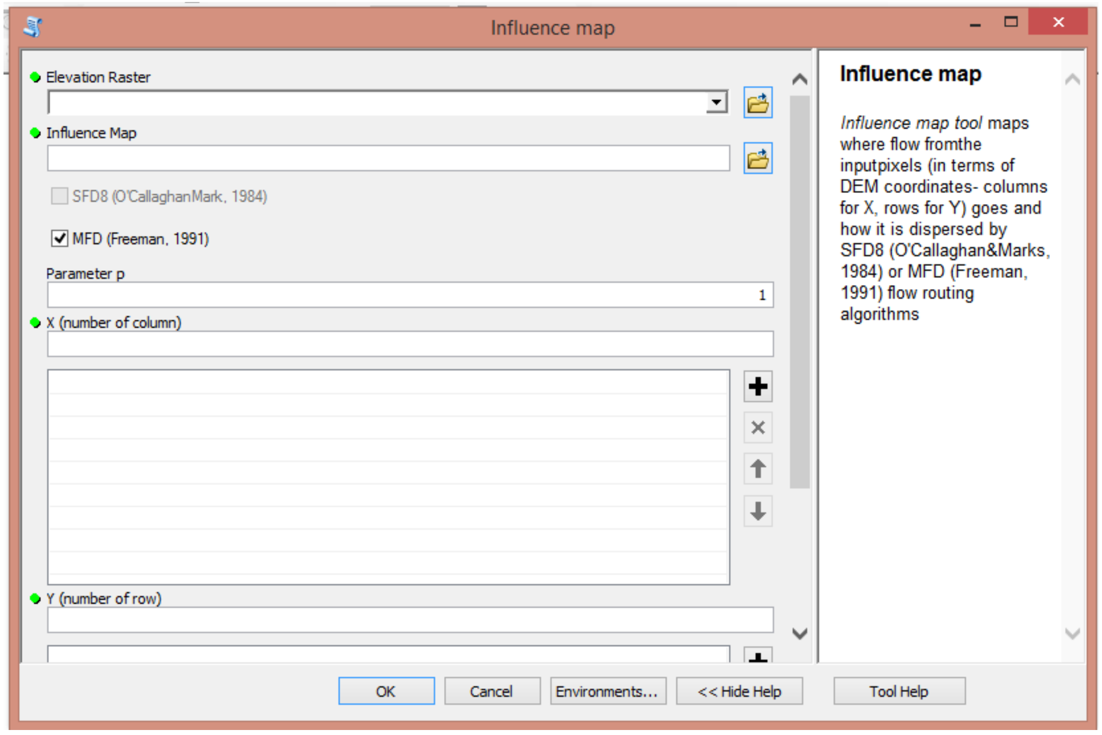
Flow accumulation by MFD

Flow Simulation (Flow Accumulation) using Multiple Flow Direction algorithm (Freeman, 1991). Parameter p controls the extent of divergency.

Figure 5-6. Dialog window of Flow accumulation tool

Influence Map tool(*Figure 5.7.*) provides possibility to map flow paths and accumulation values by SFD or MFD algorithms from individual pixels. Required parameters are input DEM, output influence map, x and y coordinates in terms of raster column and row number of pixels of interests. In case of checking MFD, user can optionally set value of parameter p, which is otherwise by default is set to 1.

Tool can be easily further extended by providing option to use weights raster, as in *Flow accumulation tool.*



Influence map

Elevation Raster

Influence Map

SFD8 (O'CallaghanMark, 1984)

[✓ MFD (Freeman, 1991)]

Parameter p

1

X (number of column)

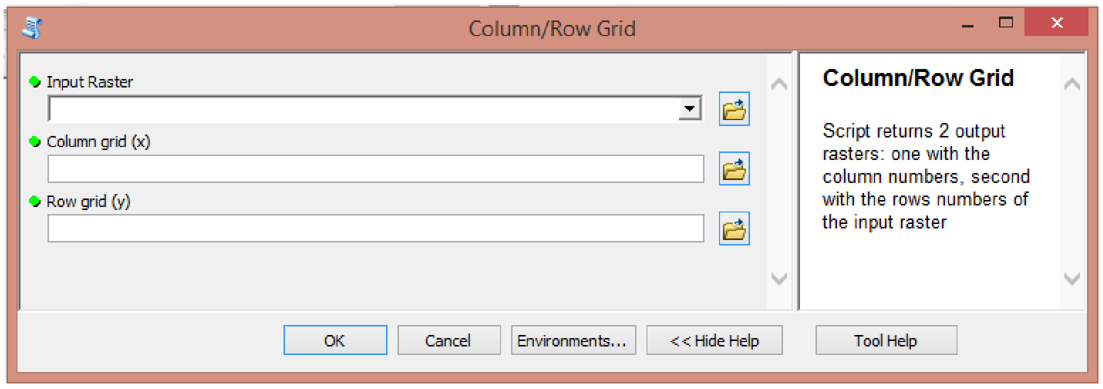
Y (number of row)

Influence map

Influence map tool maps where flow fromthe inputpixels (in terms of DEM coordinates- columns for X, rows for Y) goes and how it is dispersed by SFD8 (O'Callaghan&Marks, 1984) or MFD (Freeman, 1991) flow routing algorithms

Figure 5-7. Dialog window of Influence Map tool

To define column and row number for chosen pixels, one can use supplementary tool Column/Row Grid, which from any input raster creates 2 output rasters – with column and row numbers (*Figure 5.8.*).



Column/Row Grid

Input Raster

Column grid (x)

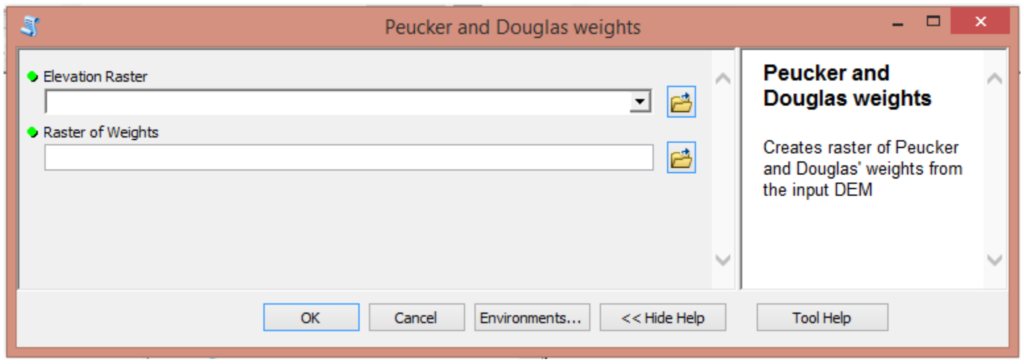
Row grid (y)

Column/Row Grid

Script returns 2 output rasters: one with the column numbers, second with the rows numbers of the input raster

Figure 5-8. Dialog window of Column/Row Grid tool

Peucker&Douglas weights tool from input DEM creates raster of weights 0 or 1 according to elevations: 1 for lower parts which tend to accumulate flow (*Figure 5.9.*). This raster can be used in *Flow accumulation tool*. It can be especially useful for water stream delination.


Peucker and Douglas weights

Elevation Raster

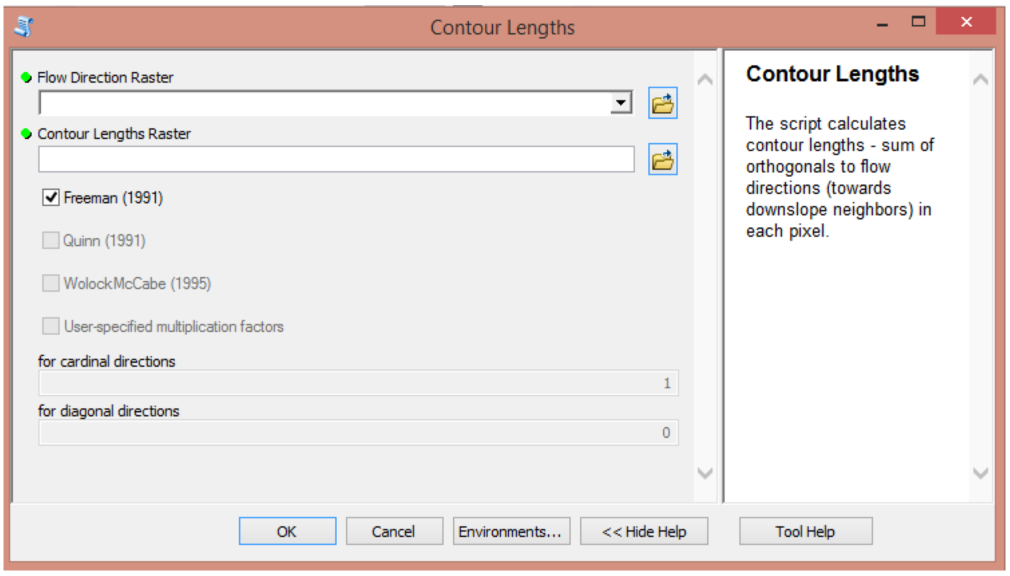
Raster of Weights

Peucker and Douglas weights

Creates raster of Peucker and Douglas' weights from the input DEM

Figure 5-9. Dialog window of Peucker and Douglas weights tool

Contour Lengths tool uses flow direction raster as input and gives output raster of contour width for each pixel (*Figure 5.10.*). User specify, which definition of contour length to use or specify his own multiplication factors for different flow directions. Resulted raster is used in combination with flow accumulation raster to calculate SCA by built-in ArcGIS *Math tools*.


Contour Lengths

Flow Direction Raster

Contour Lengths Raster

[✓] Freeman (1991)

[ ] Quinn (1991)

[ ] WolockMcCabe (1995)

[ ] User-specified multiplication factors

for cardinal directions: 1

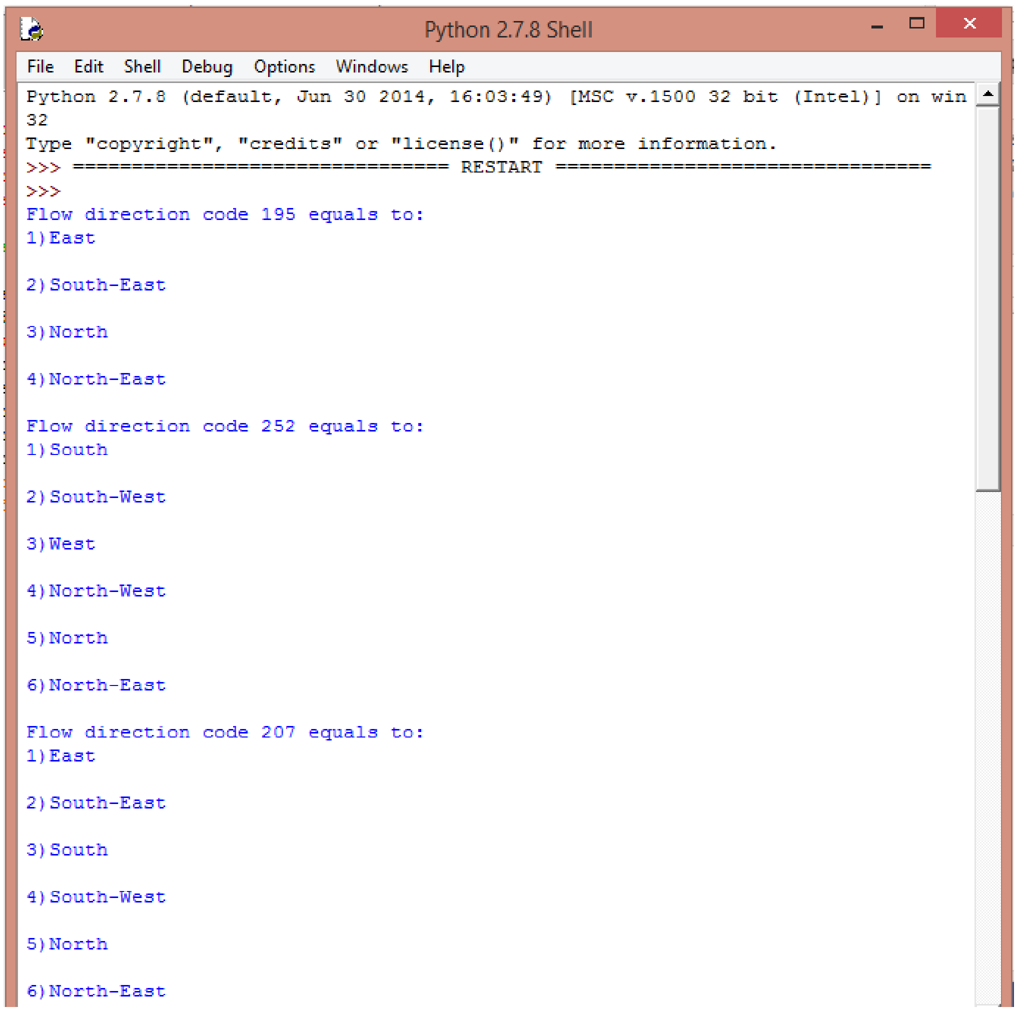
for diagonal directions: 0

Contour Lengths

The script calculates contour lengths - sum of orthogonals to flow directions (towards downslope neighbors) in each pixel.

Figure 5-10. Dialog window of Contour Lengths tool

Extra script was written to obtain flow directions from any given MFD code from flow direction raster. Providing *MFD code calculator* with a list of MFD codes of interest, one can get resulting directions in *Python Shell*window (*Figure 5.11.*).



```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
Flow direction code 195 equals to:
1)East

2)South-East

3)North

4)North-East

Flow direction code 252 equals to:
1)South

2)South-West

3)West

4)North-West

5)North

6)North-East

Flow direction code 207 equals to:
1)East

2)South-East

3)South

4)South-West

5)North

6)North-East
```

Figure 5-11 MFD code calculator output in Python Shell

5.2 Scripts documentation

5.2.1 MFD.py

The script contains 2 functions: *neighbors_z* and *slope*.
Neighbors_z is looking for the elevations of the neighboring cells.
Input: ndarray, x, y – coordinates of the central cell.
Returned value: array of elevations of 8 neighboring cells
Slope calculates slope to each of the neighboring cells.
Input: elevation of the central cell, array of neighbors' elevations, grid size
Returned value: array of slopes to 8 neighboring cells
Input DEM is transfered to ndarray. Flow directions to all neighborswith positive slopes for each element of DEM array(i.e. each cell of DEM)
are then determined and summated into output array of flow directions. Simultaneously, control of sink areas is performed (if maximum slope will be less or equal zero, it is flat or sink in the central cell).
Input: sys.argv[1] = DEMraster
sys.argv[2] = name and directory of output raster
Output: rasterof flow directions

5.2.2 Fdisp.py

The script calculates flow dispersion - number of cells to which flow is routed from the current cell. Input Flow Direction raster is transfered to an array, for each cell of which number of non-zero bits (equivalent to the number of directions towards neighboring cells) is calculated and converted to output 'flow dispersion' raster.
Input: sys.argv[1] = raster of flow directions
sys.argv[2] = name and directory of output raster
Output: raster of dispersion

5.2.3 FSmfd.py

The script contains 3 functions: *neighbors_z*, *neighbors_xy* and *slope*.

Neighbors_z is looking for the elevations of the neighboring cells (see 5.2.1).

Neighbors_xy searches for the coordinates [x,y] of the neighoring cells.

Input: x, y – coordinates of the central cell
Returned value: array of coordinates of 8 neighboring cells
Slope calculates slope to each of the neighboring cells (see 5.2.1).

Input DEM is transfered to an array. Initial array of flow accumulation is created (array of ones + weights if provided).Heap of vectors(x,y,z) sorted by the value of elevation "z" is created.Starting with a cell with maximum elevation flow directions to all neighbors with positive slopes are determined.Accumulation value of a cell in the direction of a flow is increasedby an accumulation value of the currentlyprocessed cell according to the weight of its slope value in all positive neighboring slopes.

Input: sys.argv[1] = DEM raster
sys.argv[2] = output raster of flow accumulation
sys.argv[3] = parameter p
sys.argv[4] = True/False (User-specified weights)
sys.argv[5] = raster of user-specified weights
Output: flow accumulation raster

5.2.4 PaD.py

The script has two functions: *nPaD* a *PaD*.

nPaD is looking for elevations and coordinates of 3 neighboring cells.
Input: array, x, y – coordinates of the central cell
Returned value: array of 6 values – first 3 are elevations and second 3 are coordinates of 3 neighboring cells
PaD reads input DEM and converts it to ndarray, then using 2x2 pixels window, compares elevation of central cell with its 3 neighbors for each cell of DEM. Cells, which has never had the highest elevation, will take value 1 for output raster.
Input: DEMraster
Returned value: array – local terrain curvature or raster of Peucker&Douglas weights

5.2.5 Influence_map.py

The script maps where flow goes from the input pixels (coordinates in column, row terms) and how it is dispersed. User can choose MFD8 (Freemn, 1991) or SFD8 (O'Callaghan&Mark, 1984) algorithm for flow routing.

The script contains 3 functions: neighbors_z, neighbors_xy and slope (see 5.2.3.).Neighbors_z is looking for the elevations of the neighboring cells.Neighbors_xy searches for the coordinates [x,y] of the neighoring cells.Slope calculates slope to each of the neighboring cells.

Input DEM is transfered to an array. Initial array of flow accumulation is created (array of zeroes). Input coordinates of source pixels are used to assign flow accumulation values to ones for the input source pixels.

Heap of vectors(x,y,z) sorted by the value of elevation "z" is created. Slope is calculted.If SFD8 was chosen, flow is routed towards direction of maximum slope, starting with a cell with maximum elevation. In case of MFD8, also starting with a cell with maximum elevation, flow directions to all neighbors with positive slopes are determined.Accumulation value of a cell in the direction of a flow is increased by an accumulation value of the currently processed cell according to the weight of its slope value in all positive neighboring slopes.

Input: sys.argv[1] = DEM raster
sys.argv[2] = output raster of influence
sys.argv[3] = True/False (SFD)
sys.argv[4] = True/False (MFD)
sys.argv[5]=input x coordinates (DEM column numbers) of pixels of interest
sys.argv[6]= input y coordinates (DEM row numbers) of pixels of interest
Output: map of influence raster

5.2.6 nColRow.py

The script calculates the number of column and row of pixels of the input raster, producing 2 rasters of column and row numbers of each pixel. Input raster is coverted to ndarray.Ndarray of columns and rows numbers of the input raster is created using numpy.indices method. Arrays of columns and rows are extracted from ndarray by indexing and converted to rasters.

Input: sys.argv[1] = raster of interest
sys.argv[2] = output raster of columns numbers
sys.argv[3] = output raster of rows numbers
Output: 2 rasters – column and row grids

5.2.7 ContourLength.py

The script calculates contour lengths (width) - sum of orthogonals to flow directions (towards downslope neighbors), which equals to: 1) grid size for cardinal neighbors and grid size*√2 for diagonal neighbors (Freeman, 1991); 2)grid size*0,5 for cardinal neighbors and grid size*0,354 for diagonal neighbors (Quinn et al., 1991); 3)grid size*0,6 for cardinal neighbors and grid size*0,4 for diagonal neighbors (Wlock&McCabe, 1995) or 4) grid size multiplied by user-specified factors different for cardinal and diagonal directions.

Input Flow Direction raster is transfered to an array, for each cell of which number of flow directions towards diagonal and cardinal neighboring cells is determined. Accordingly, sum of contour length is calculated and converted to output raster.

Input: sys.argv[1] = flow direction raster
sys.argv[2] = output raster of contour widths
sys.argv[3] = True/False (Freeman)
sys.argv[4] = True/False (Quinn)
sys.argv[5]= True/False (Wolock&McCabe)
sys.argv[6]= True/False (User-Specified multiplication factors)
sys.argv[7]= number representing multiplication factor for cardinal direction
sys.argv[8]= number representing multiplication factor for diagonal direction
Output: raster of contour widths

5.2.8 rta.py

The script contains one function *rta* (Raster To Array), which converts input raster to ndarray
Input: raster
Returned value: new_array, XMax, YMax, vCell, lowerLeftX, lowerLeftY
(ndarray, number of columns, number of rows, grid size, X,Y coordinates of the left lower corner of raster)

5.2.9 Heap.py

Scripthas 4 functions: *cr_heap, inz_el, ret_max a cr_list.*
inz_el inserts new item to the end of the heap, compares it with the previous one and may swap them, till correct heap order not returned
Input: array, index ofvalue, on which heap should be created (x,y,z)
Returned value: heap–array sorted to heap structure

*cr_heap*with the help of*inz_e*l creates from the array binary heap.
Input: heap, inserted item , index of value after sort is performed
Output: resorted heap
*ret_max*inserts the last item from heap's leaf to its root after root value (maximum) was removed and by iterative comparisons put it back to the last position, so heap is restructured back.
Input: heap, index of value for sorting
Output: resorted heap
cr_list makes sorted array from binary heap
Input: heap
Returned value: sorted array

5.2.10MFD_code_calculator.py

The script returns directions for any given codes of multiple directions. For every element of the input code array, the position of non-zero bits in its binary representation is interpreted as the same index in array of directions, which is printed out.

Input: array of flow direction codes

Output: lists of flow directions for every element of the input

5.3 Demonstration of developed tools

5.3.1 Used data

For the demonstration of the functionality of the developed tools coarse-resolution SRTM DEM (Shuttle Radar Topography Mission: http://www2.jpl.nasa.gov/srtm/) of basin Moravska Dyje was used (*Figure 5.12.*). Grid size is 90x90m. Total raster is 468 columns x 330 rows = 154440 pixels or 1251 km^2. Flat areas and depressions were removed inprogram DEMETERR (Bartak, 2008)by method *Combined Gradient* and *Carving* respectively.

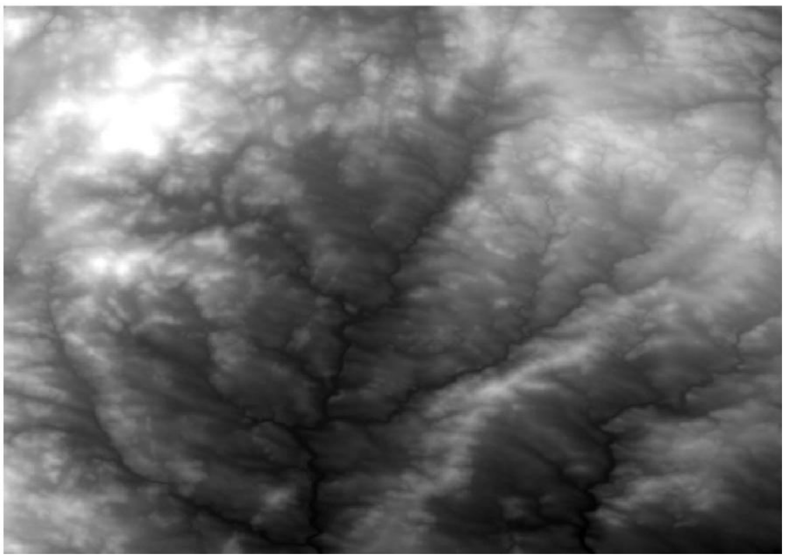
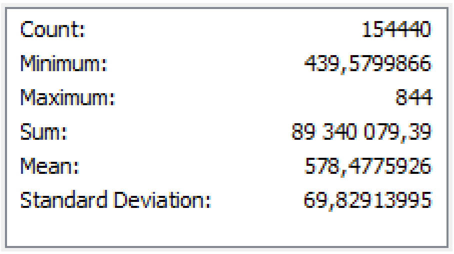


Figure 5-12 DEM of Moravska Dyje basin. Black color represents the lowest elevation

Basic statistic parameters and elevation distribution of the area are shown in *Table 5.1.* and *Figure 5.13.* respectively.

Table 5-1 Basic statististic characteristics of DEM, m



Count:	154440
Minimum:	439,5799866
Maximum:	844
Sum:	89 340 079,39
Mean:	578,4775926
Standard Deviation:	69,82913995

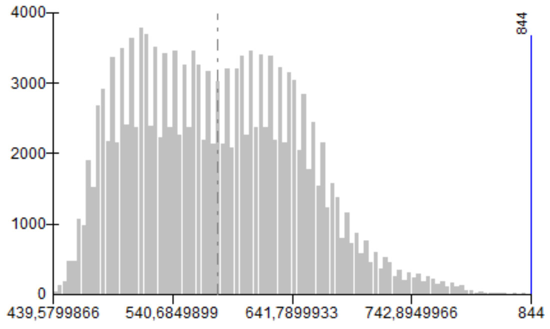


Figure 5-13. Histogram of DEM: Y - number of cells, X - elevations, m

Ruggedness of terrain is presented in profile from North-West to South-East corner of DEM raster (*Figure 5.14.*)

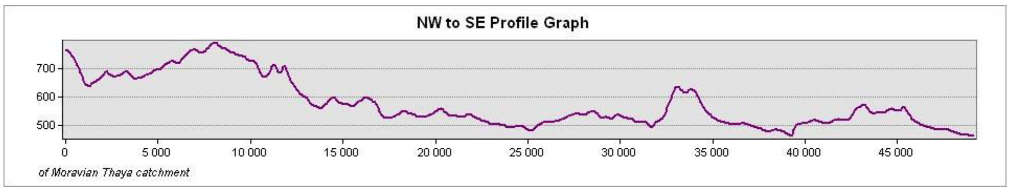


Figure 5-14 Profile graf from North-West to South-East corner of DEM, m

Thus, the terrain is quite rugged with elevations range around 400 m with high percentage of hillslope area.

5.3.2 Determination of flow direction

Resulting flow direction rasters derived by SFD8 and by MFD are shown on *Figure 5.16* and*5.17.* respectively. Color scheme for SFD direction are shown on *Figure 5.15.* Color scheme for MFD raster is not shown because of high number of unique flow direction values.

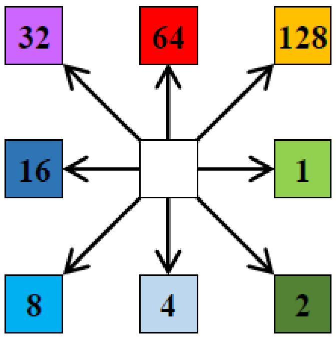


Figure 5-15 Color scheme for SFD direction codes

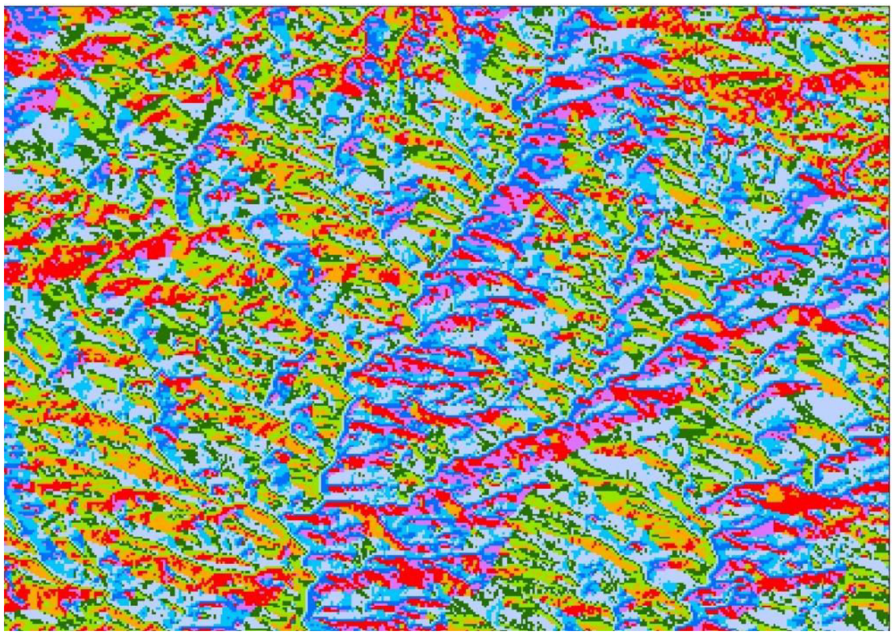


Figure 5-16 Raster of SFD flow directions

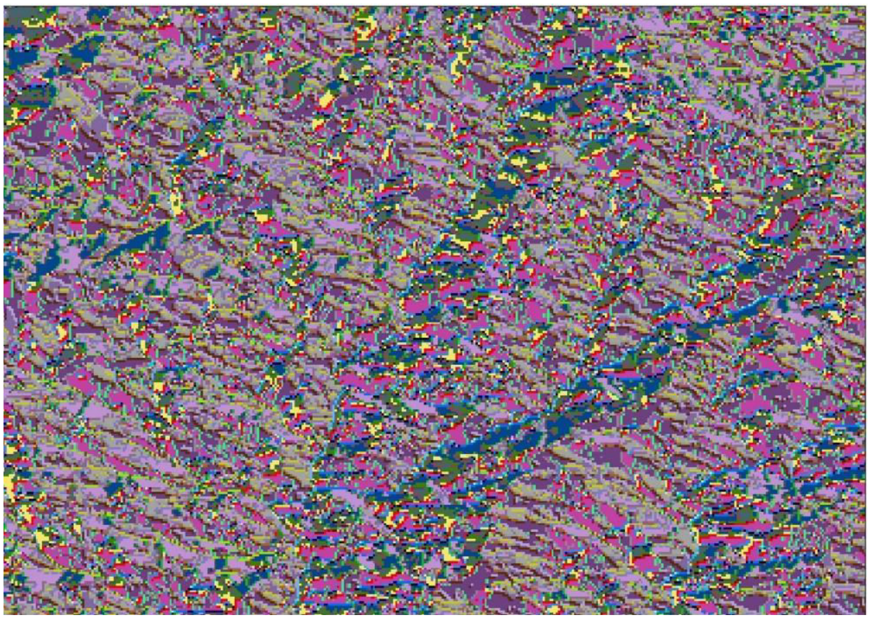


Figure 5-17 Raster of MFD flow directions

Visual comparison of resulting rasters indicates that for MFD color patches with the same colors are much smaller, than those for SFD.

Single flow directions were determined only for 5% of total DEM area (7629 pixels of 1 direction from 152848 pixels of total area with determined flow directions.Single directions simulated by MFD algorithm occur in the lowest parts of terrain, obviously along the stream channels and has the same direction as a stream as can be seen from the bands of the same colour(*Figure 5.18.*)

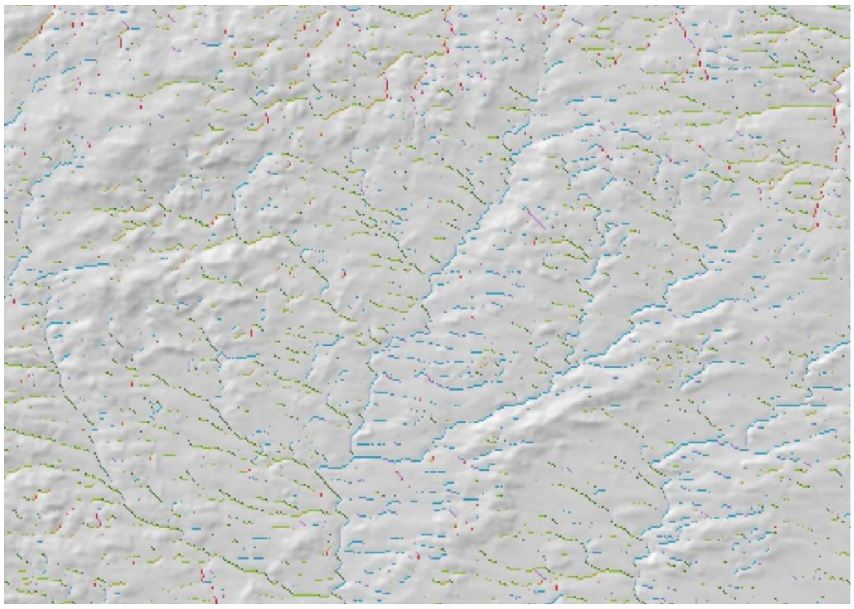


Figure 5-18 Single flow directions within MFD direction raster

The maximum number of single-directional cells is with Southern direction, which is in accordance with the orientation of the main stream or main valley bottom features (Northern-Southern) (*Table 5.2.*).

Table 5-2 Single flow directions within MFD irection raster

Flow direction	Code	Number of pixels
E	1	19726
SE	2	19612
S	4	34458
SW	8	15720
W	16	14039
NW	32	11583
N	64	20928
NE	128	16782

5.3.2.1 *Evaluation of flow dispersion introduced by MFD approach*

Flow dispersion tool produced raster of numbers of directions in each pixel from MFD direction raster. The highest number of DEM is represented by cells with 4 flow directions (62766 pixels) (*Figure 5.19*)

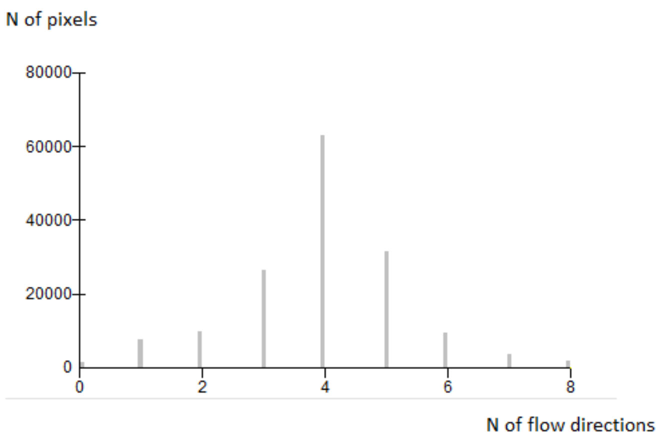


Figure 5-19 Distribution of numbers of directions in each pixel of DEM

If the number of directions is more than 3, it is possible that the flow is overdispersed (Tarboton, 1997). In our case, only about 40% of DEM area is area with “realistic” number of flow directions (1-3), while the majority of DEM pixels route flow towards 4 and 5 neighboring pixels (41% and 21% accordingly) (*Figure 5.20* and *5.21*). So choosing of paprameter p is supposed to play important role in subsequent calculation of TCA and SCA for controlling extent of flow dispersion.

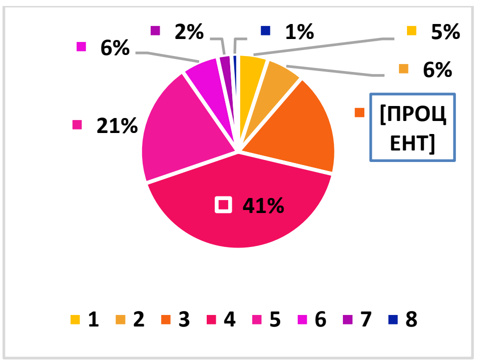


Figure 5-20 Shares of pixels with different numbers of flow directions simulated by MFD

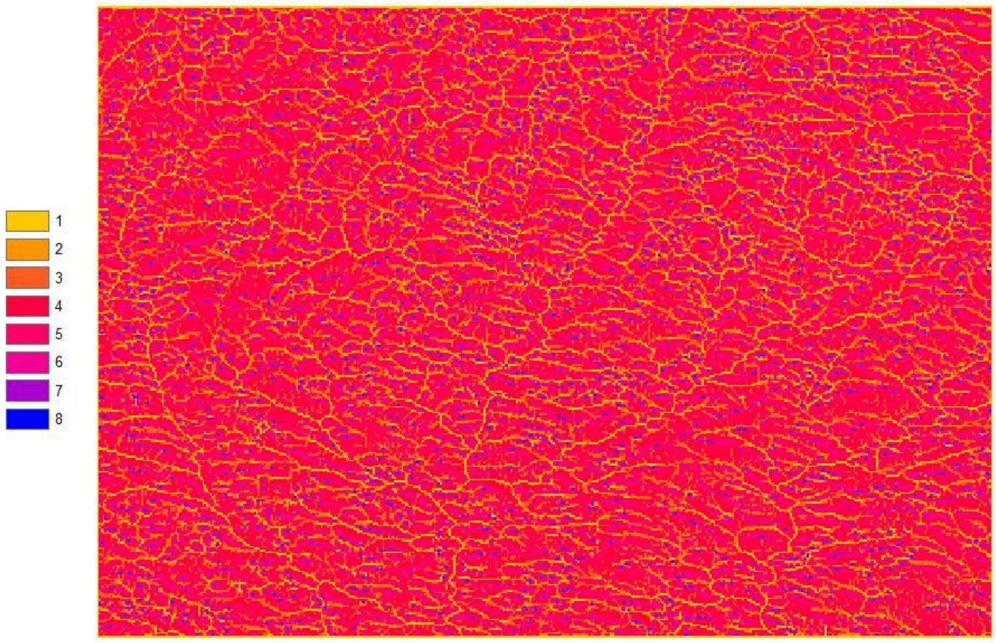


Figure 5-21 Flow dispersion raster (in legend: numbers of flow directions simulated by MFD)

Flow dispersion raster can be potentially used in river network extraction. If we set the threshold of number of directions to 2, we will have raster, similar to stream network (*Figure 5.22.*).

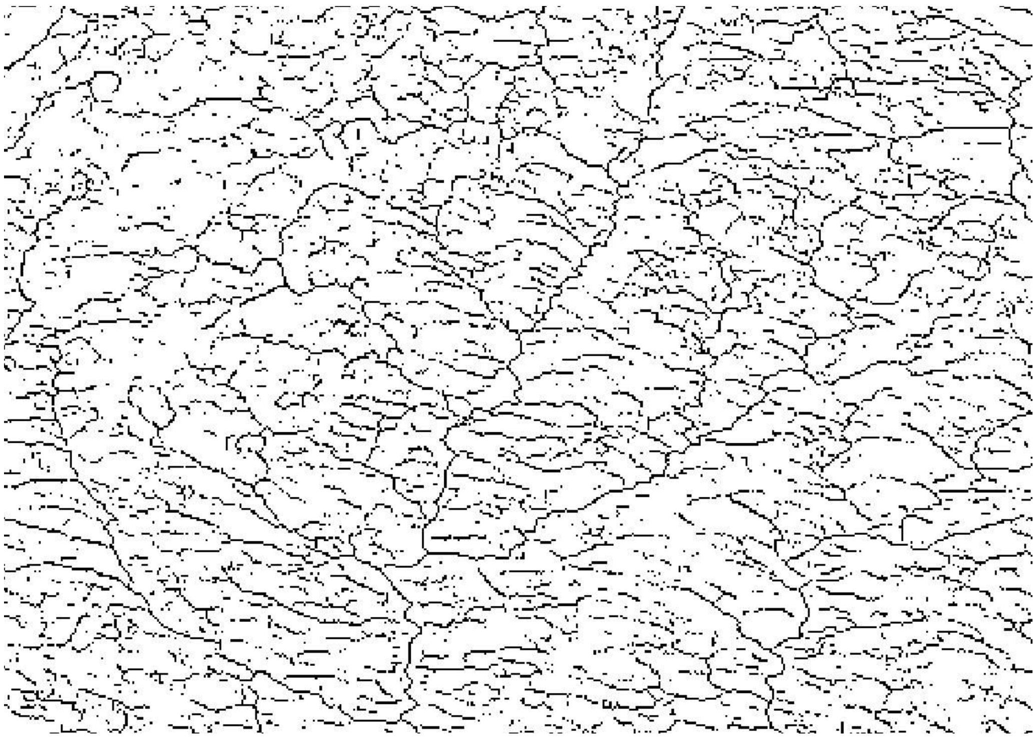


Figure 5-22 Network of pixels with 1-2 directions simulated by MFD

5.3.3 Flow simulation

Rasters of accumulated area derived with MFD algorithm with different values of parameter p and with SFD8 algorithm visually look very similar (we use the same color scale with the same maximum value of accumulation to be white and minimum black) (*Figure 5.23*).

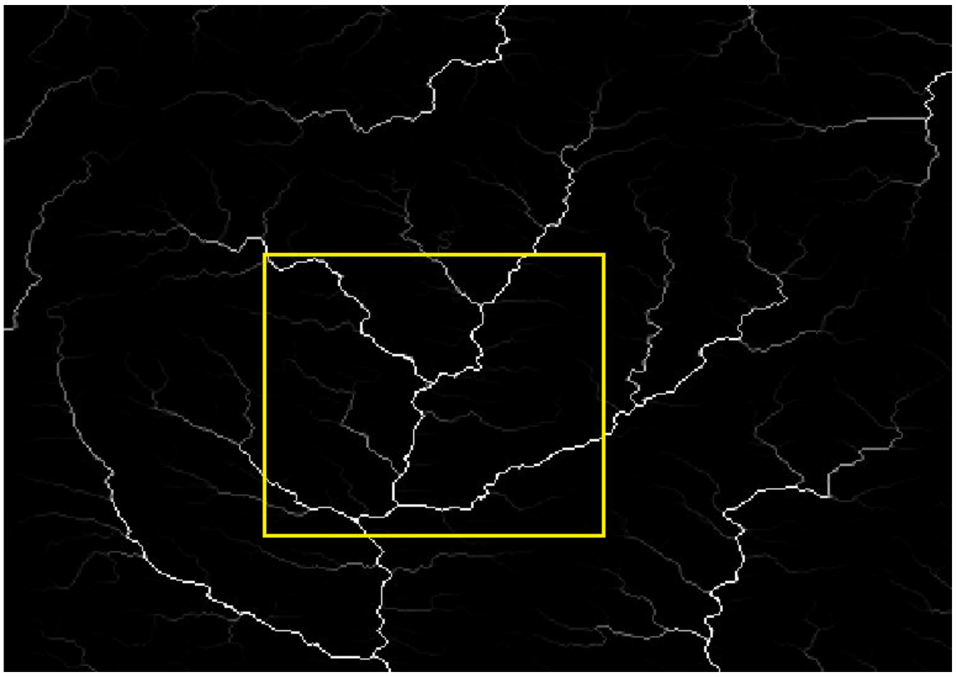


Figure 5-23 Flow accumulation (maximum values are white)

Enlarging part of the area (which is depicted by rectangle in *Figure 5.23*), we can notice at some pixels higher dispersion of accumulation values with lower parameter *p* (*Figure 5.24*). Flow accumulation raster derived using SFD8 algorithm look similar to that of MFD with p=10. Raster from p=5 is visually something middle between rasters from p=1 and p=10: more dispersion (pixels with higher flow accumulation values than for p=10, but less then for p=1).

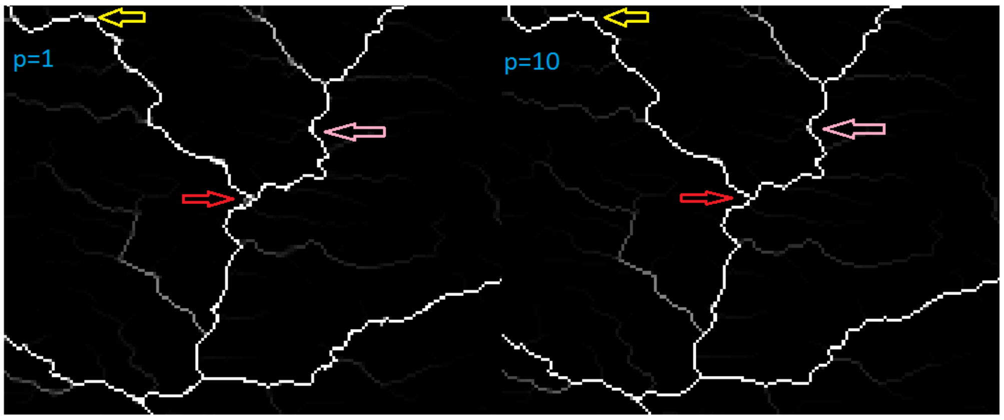
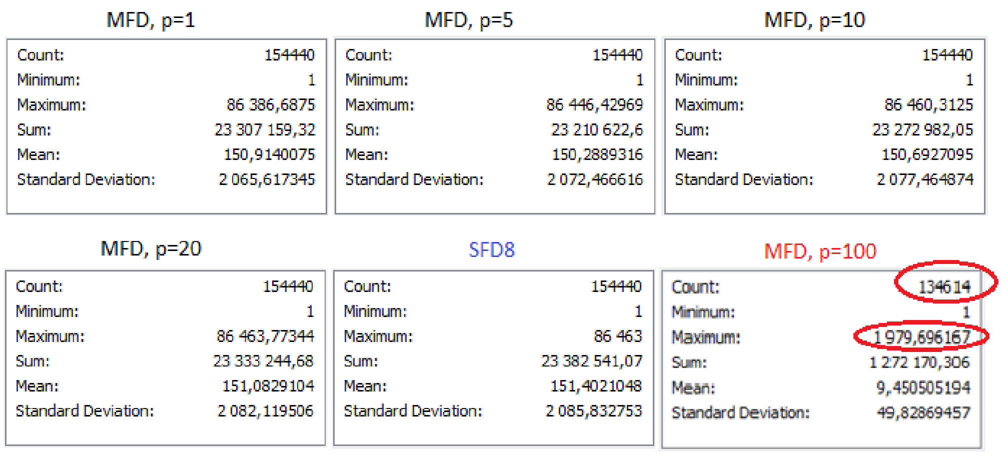


Figure 5-24 Flow accumulation by MFD with p=1 and p=10 (arrows indicate dispersed flow for p=1)

Rasters of flow accumulation derived for p=1.0 and p=1.1 (recommended by Freeman (1991) as the best choice for flow simulation on right cone surface) are absolutely identical (quick check is performed using *Minus tool* from *Math* toolset of ArcMap *Spatial Analyst Toolbox*, resulting raster being raster of zeroes). Probably, there is no differences, because we use DEM with coarse resolution (90 m). For subsequent comparative analysis of influence of parameter p on resulting TCAs and SCAs we choose to use p=1.0.

Flow accumulation rasters statistics shows that with increasing p up to 20 mean, maximum, sum and standard deviation of flow accumulation values are increasing being close to that of SFD8 algorithm (*Table 5.3*).

Table 5-3 Basic statistical characteristics of Flow Accumulation, in pixels



	MFD, p=1	MFD, p=5	MFD, p=10
Count:	154440	154440	154440
Minimum:	1	1	1
Maximum:	86 386,6875	86 446,42969	86 460,3125
Sum:	23 307 159,32	23 210 622,6	23 272 982,05
Mean:	150,9140075	150,2889316	150,6927095
Standard Deviation:	2 065,617345	2 072,466616	2 077,464874

	MFD, p=20	SFD8	MFD, p=100
Count:	154440	154440	134614
Minimum:	1	1	1
Maximum:	86 463,77344	86 463	1 979,696167
Sum:	23 333 244,68	23 382 541,07	1 272 170,306
Mean:	151,0829104	151,4021048	9,450505194
Standard Deviation:	2 082,119506	2 085,832753	49,82869457

Histograms will not show us significant differences, as majority of cells will display low flow accumulation values, thus for better observation of differences logarithmic transformation of values is needed. However, with higher deviation, sum and mean of flow accumulation for SFD8, one can conclude that MFD has a smoother distribution of flow accumulation values, while SFD8 rapidly “concentrate” the flow. Theoretically with p = ∞, MFD will be equal to SFD. Trying to simulate flow with p = 100, unrealistic result was derived: flow accumulation was not calculated for all cells of the current DEM (only for 134 614 pixels instead of 154 440 pixels of whole input DEM), the highest value of flow accumulation was too low (more than 40 times lower than this value for SFD8 and MFD with p = 1, 5, 10, 20) (*Table 5.3.*). It is possible that problem lies in dealing with too small numbers with large numbers of digits after floating point which will be created by powering slope values (which are always <1) by 100 (in calculation of flow portion weighted by positive slopes). On a typical machine running Python, there are 53 bits of precision available for a Python float. Thus, if we have slope value with only 1 digit after point, the maximum p=53 in sense of not losing a precision.

If we change classification scheme to show flow accumulation less then 1 pixel (which means that such cells accumulate only themselves) and more then 10000 pixels (cells with high accumulation values, probably water streams), we notice that MFD tends to produce disconnected network of cells with high TCA in comparison with SFD, but it produces less one values cells, which is more realistic (*Figure 5.25*).

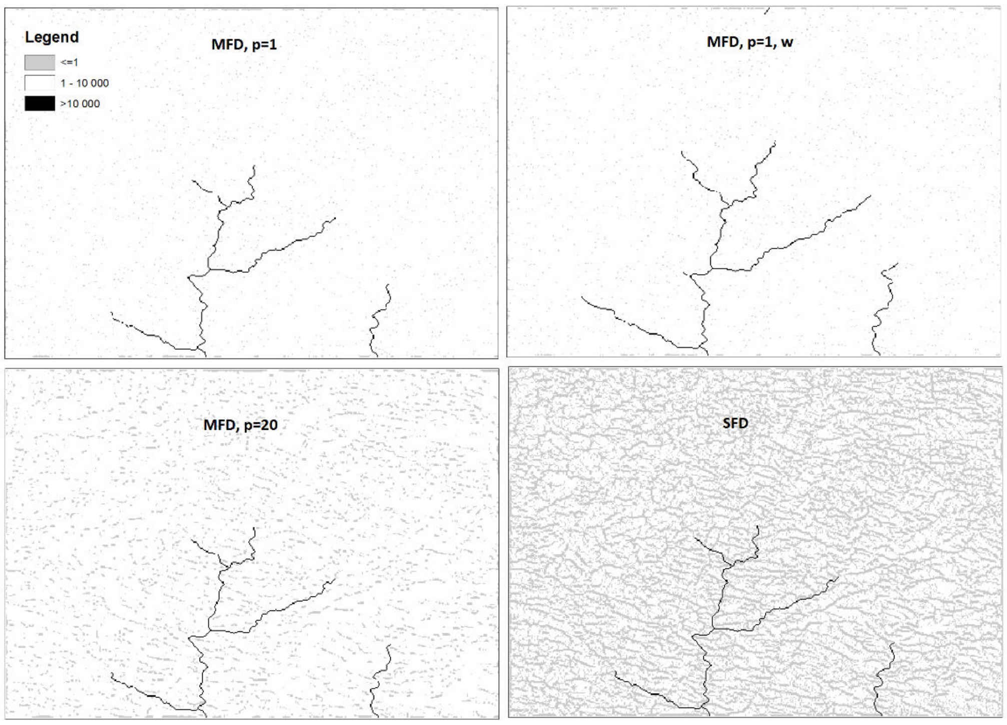


Figure 5-25 TCA (pixels) by MFD, SFD and MFD with Peucker and Douglas weights (w)

If we use raster of Peucker and Douglas weights, derived by developed tool on same DEM (*Figure 5.26*), there are more cells with high accumulation values and they seem to be more connected (*Figure 5.25, upper right corner*). Therefore, Peucker and Douglas weights could serve as a valuebale tool for water stream delination.

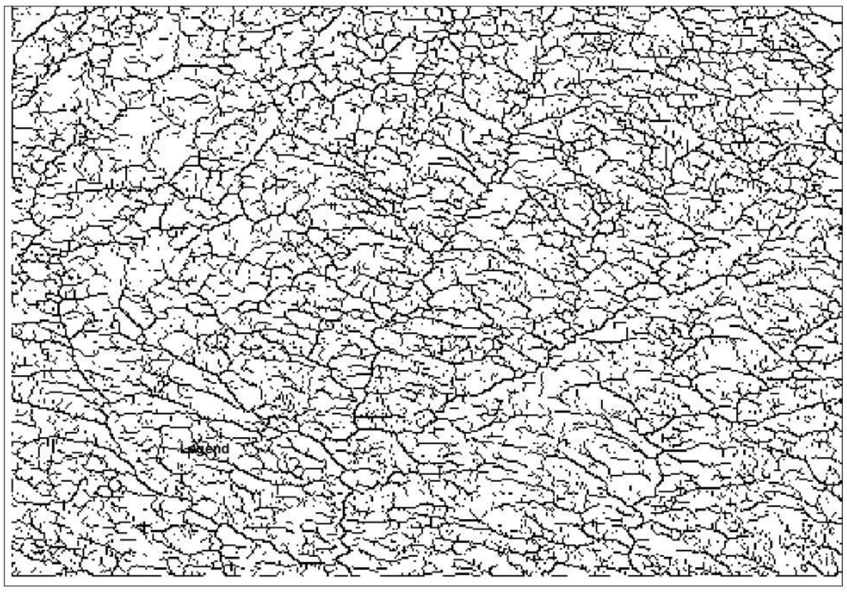


Figure 5-26 Raster of Peucker and Douglas weights for the studied DEM: white - 0; black - 1

If we simply substract resulted flow accumulation raster of MFD with p=1 from raster derived by SFD by ArcGIS *Minus tool*, we notice that differences are ambiguos: some pixels have higher TCA for SFD, some for MFD, p=1 algorithms (*Figure 5.27, left*), - but all of them are situated close to water streams. If we compare TCAs of MFD with p=1 and p=10, flow accumulation is always higher for MFD, p=10 and pixels with differences are always located near the water streams (*Figure 5.27, right*). Thus, for water streams delination, algorithm matters and parameter p in case of MFD matters as well, while for the calculation of hillslope TCAs by MFD different values of parameter *p* seem to produce similar results.

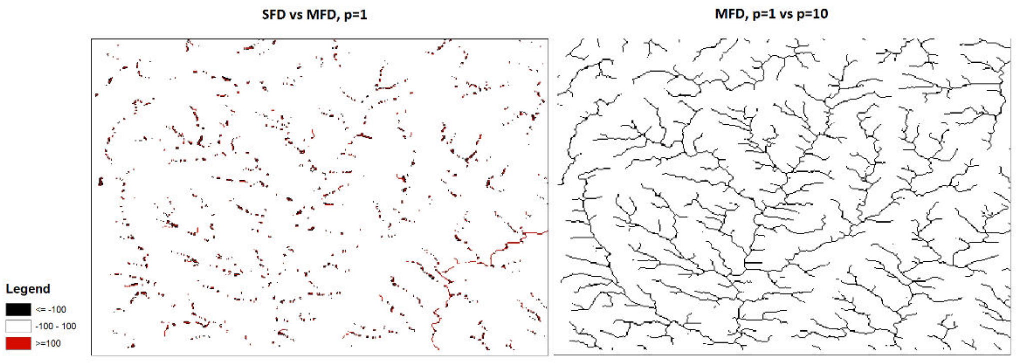


Figure 5-27 Absolute differences between TCA values (pixels): left - SFD minus MFD, p=1 ; right - MFD, p=1 minus MFD, p=10

Logarithmic scale reveals more pronounced differences between flow accumulation patterns of MFD and SFD. For SFD TCA patterns tend to be unrealistic parallel, while for MFD, p=1 features are more realistic, smoothly dispersed (*Figure 5.28.*). With increasing *p*, more distinct parallel features as in D8 appear.

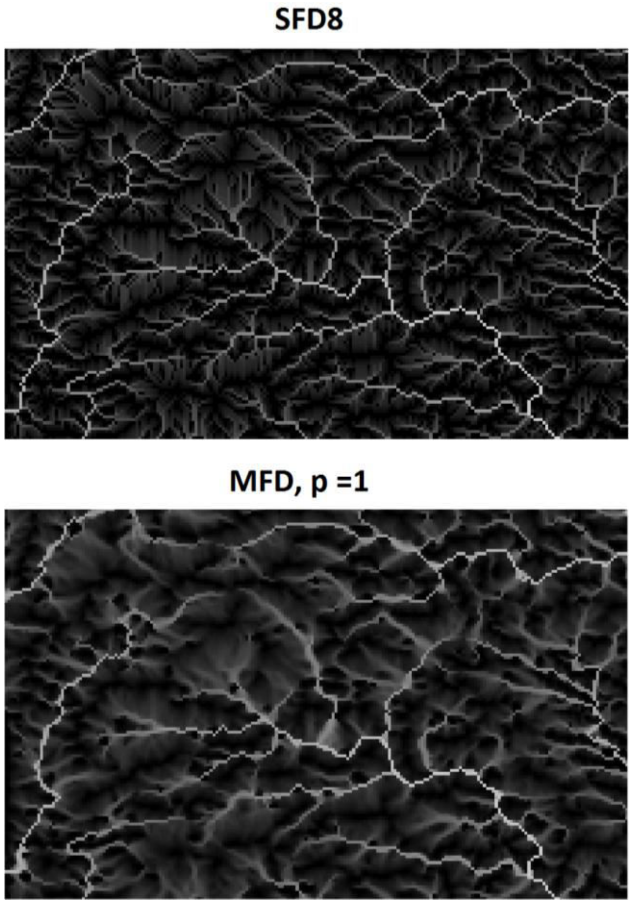


Figure 5-28 Ln (TCA) by SFD (upper) and MFD (lower) algorithms: high values are white

Histograms of ln(TCA) distributions confirm that MFD produces smoother distribution of flow accumulation, than SFD, which produce only discrete values of TCA (*Figure 5.29*), which again is unrealistic for natural body, where different properties usually change continuously in space.

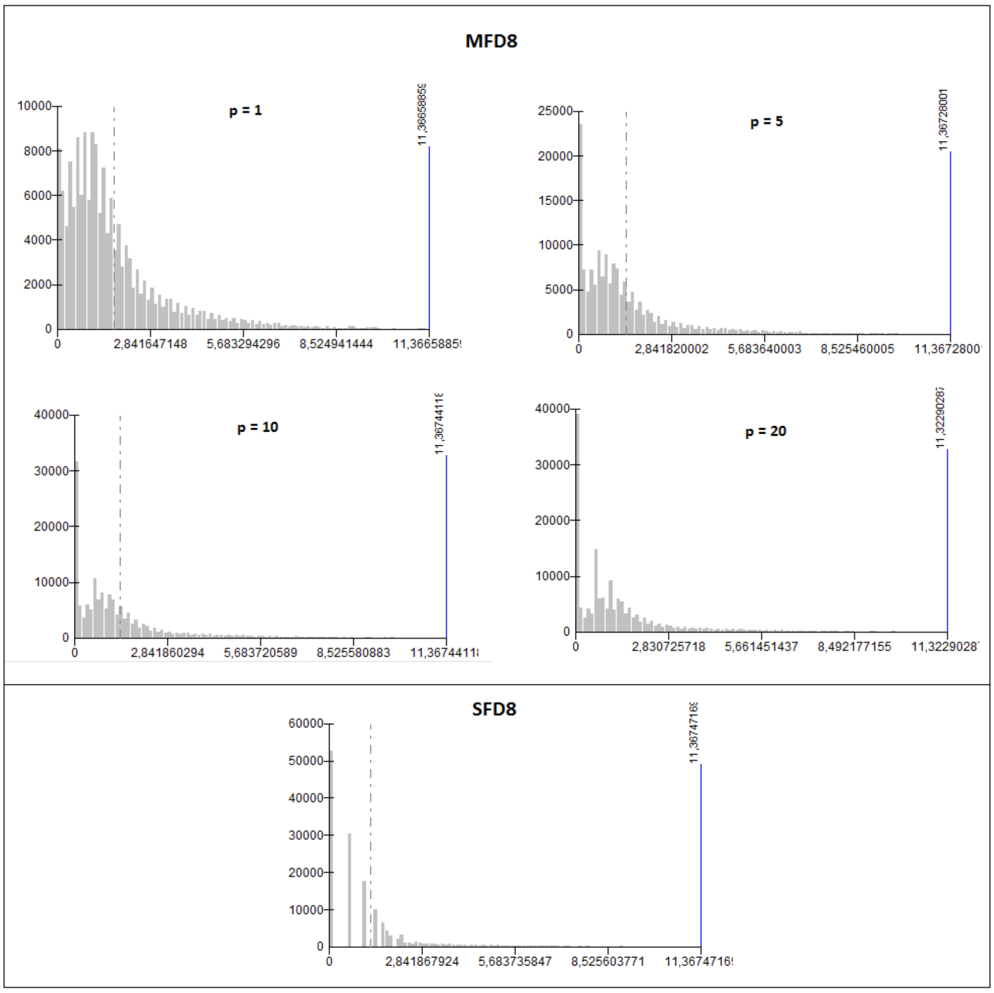


Figure 5-29 Distributions of ln(TCA) values (x) by MFD with different p values and SFD, y – number of cells

Scatter plot comparing SFD and MFD TCA values reveals that there is a line of dots representing value 1 for flow accumulation calculated by SFD8, but range of much higher values by MFD8 (*Figure 5.32*). We analyzed the location of such pixels with calculated TCA by SFD8 = 1, but by MFD8 more than 20 up to 1280(*Figure 5.30*) .

TCA with value 1 means that no other cell contributed to that particular point in SFD8 simulation, but such a great difference indicates that however slope towards such a cell was not the steepest one, it was close to the steepest, as according to that, relatively great amount of flow could enter the pixel by MFD8 algorithm. As we see in the *Figure 5.30, 5.31*, many of the points of such differences is located close to the valley bottom (stream channel), where flow accumulation values is very high. From one hand, in valley bottoms overdispersion should be avoid, so forming channels would not be too broad. From the other hand, flow accumulation values equal 1 does not seem to be realistic, as such values are supposed to be mostly on the highest part of DEM (ridges and peaks).

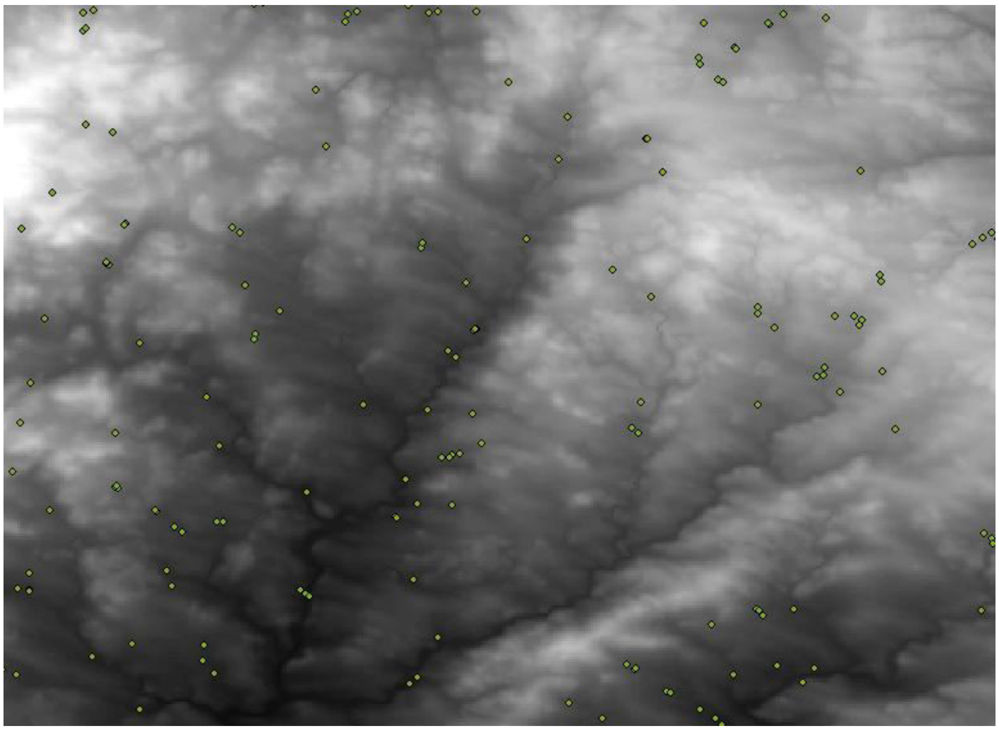


Figure 5-30 Location of pixels with calculated TCA by SFD8 = 1, but by MFD8 more than 20 up to 1280

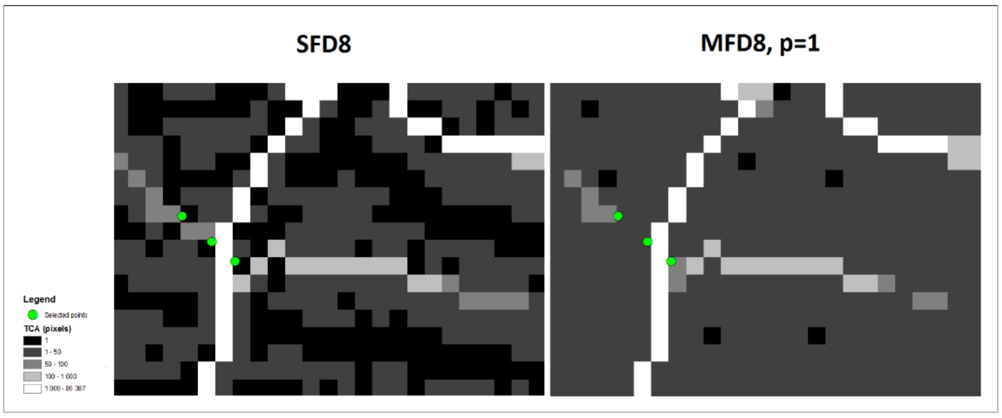


Figure 5-31 Example of great differences in TCA by MFD and SFD near water streams (white)

Comparison of scatter plots of TCA values between SFD and MFD wit *p*=1 and *p*=20 reveals that values between SFD and MFD with *p*=20 are less scattered, than between TCAs of SFD and MFD with *p*=1 (*Figure 5.32, 5.33*), which was predictable, because the higher value of parameter *p*, the more convergent flow is, the closer it to that produced by SFD.

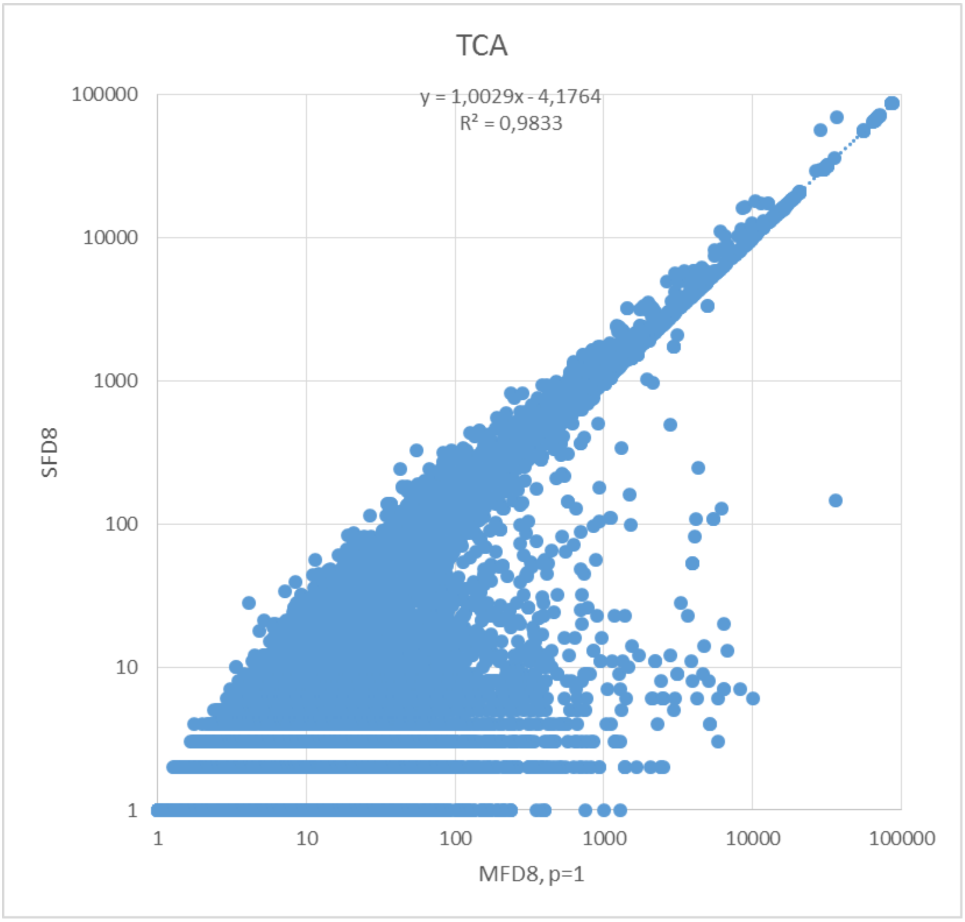


Figure 5-32 TCA produced by SFD (y) and MFD with p=1 (x)

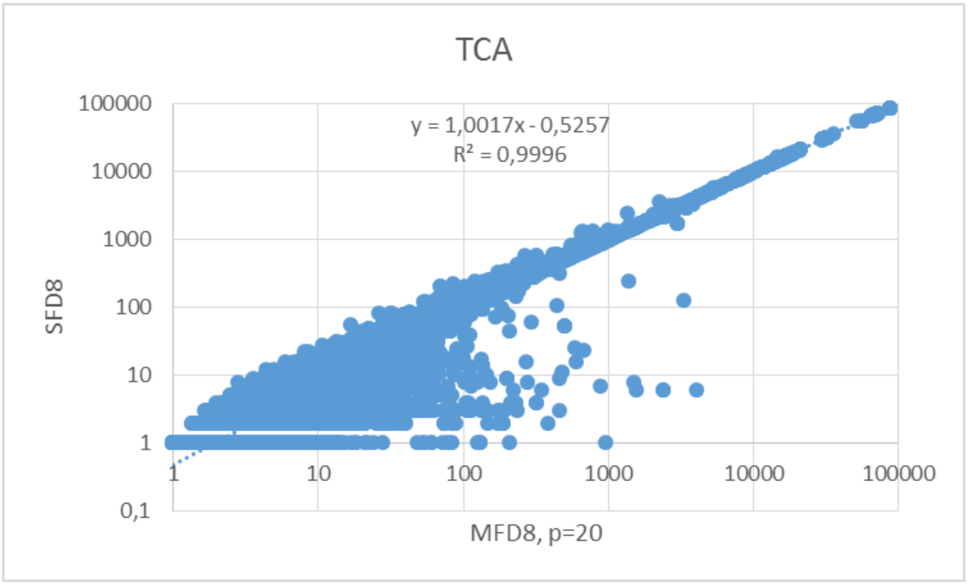


Figure 5-33 TCA produced by SFD (y) and MFD with p=20 (x)

Instead of showing all scatterplots between TCA patterns derived with different values of p, we created matrix of similarity based on R2 (coefficient of determination) of line constructed between logarithms of TCA of different algorithms (*Table 5.4.*). Basically, it is a matrix of how best linear regression performed or, in our case, it indicates how scattered data are, how dispersed from linear model.

Table 5-4 Coefficient of determination (R^2) between TCAs of SFD and MFD with different p values

Method		MFD8				SFD8
		p=1	p=5	p=10	p=20	
MFD8	p=1	1				
	p=5	0,9952	1			
	p=10	0,9901	0,9981	1		
	p=20	0,9858	0,9964	0,9992	1	
SFD8		0,9833	0,9947	0,9988	0,9996	1

The most sensitive part in the range of *p* seems to be with *p* values between 1 and 5, as R^2 differs greatly from 1 (which is hypothetically ideal case if both datasets are the same) up to 0,9833 (the worst correlation between TCAs produced by SFD and MFD with p=1(*Figure 5.34*).

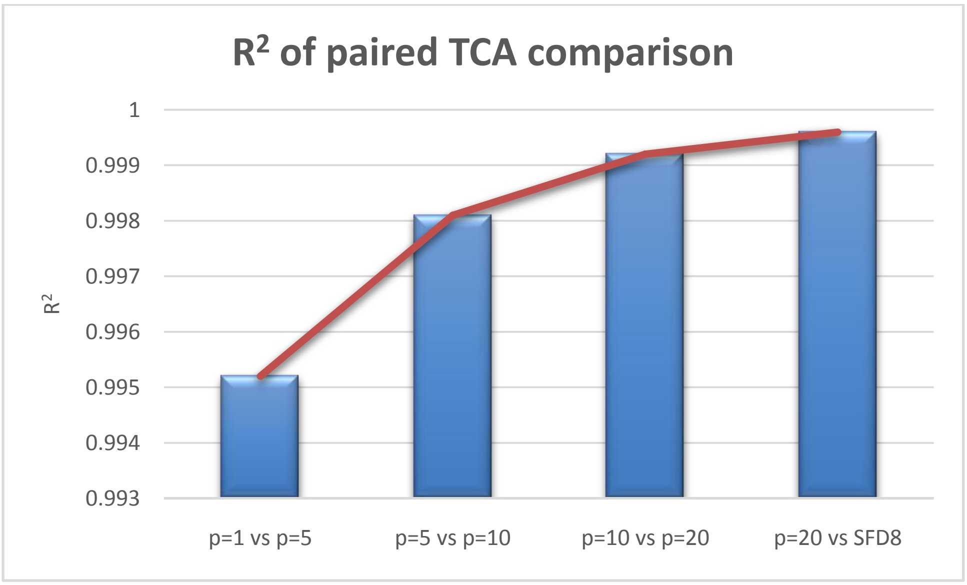


Figure 5-34 Sensitivity of determination coefficient (R^2)(y) to increasing of p value (x)

We compared tangents of linear regression, where Y-axe is always MFD with higher *p*, than on X-axis, or Y-axe is SFD8 (theoretical approximation with the infinitely high *p*) and X-axe is MFD with any *p* value (*Table 5.5.*). Mostly tangents are more than 1. It means that generally (according to the trend line) TCA for the higher *p* is also higher for the same pixel, but as we may see *in Figures 5.32, 5.33*, and *Table 5.4.*of R^2 coefficients, this rule can be applied only for the part of the line with high TCAs (which naturally occur in stream channels). In the beginning of the line the picture is absolutely different: TCAs derived by the method with the higher p (or SFD8) tend to be much lower for the same pixel than TCAs derived by the method with lower p. So for smaller TCAs (slopes, upper part of the valleys) the differences are great not only between MFD8 and SFD8 in general, but also between MFD8 with different values of p.

Table 5-5 Table of slopes of regression line between TCAs derived by SFD and MFD with different parameters (see explanation in text above)

Method		MFD8				SFD8
		p=1	p=5	p=10	p=20	
MFD8	p=1	1				
	p=5	0,9998	1			
	p=10	1,0004	1,0026	1		
	p=20	1,0002	1,0054	1,0036	1	
SFD8		1,0029	1,0064	1,005	1,0017	1

5.3.4 Influence map

For another demonstration of differences of developed MFD algorithm from ArcGIS built-in SFD, we selected 6 points, situated in on different parts of studied DEM (*Figure 5.35*) and created maps of influences of these points, showing flow distributed from these points across the terrain by SFD and MFD wit p=1.

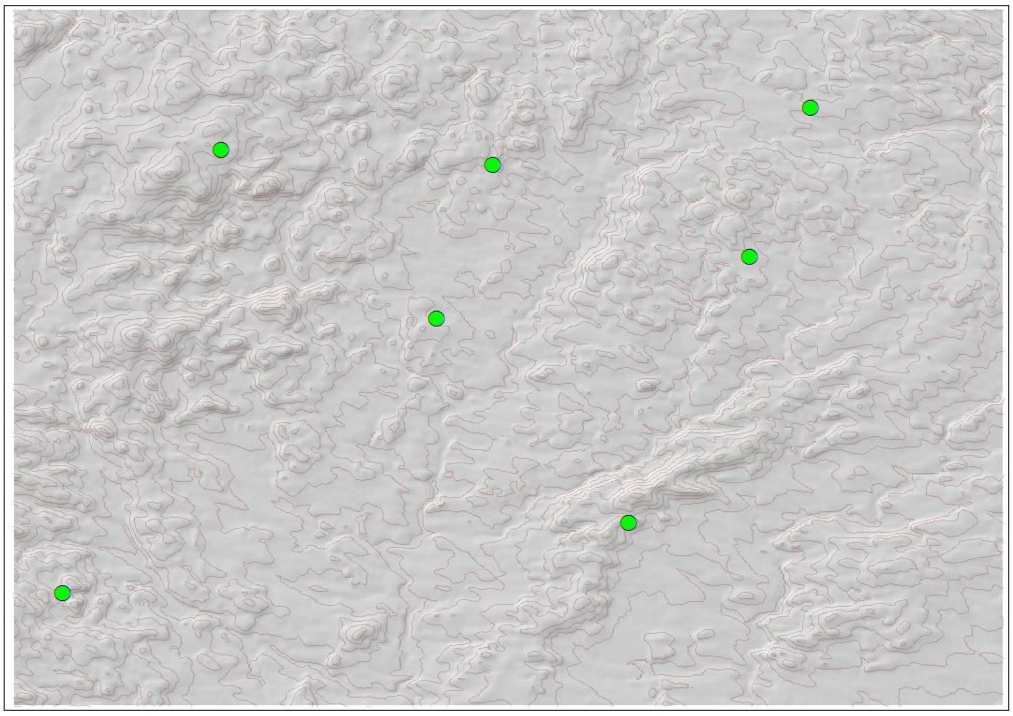


Figure 5-35Selected points for which Influence maps were drawn

All the points are situated at convergent parts of relief (*Table 5.6*) with numbers of permitted directions from each varying from 4 to 8.

Table 5-6Properties of selected source points

N of point	MFD direction code	SFD direction code	TCA, MFD, p=1	TCA, SFD	Flow dispersion by MFD
1	195	128	6,4	5	4
2	252	8	1,3	1	6
3	207	128	1,4	1	6
4	60	16	2,8	3	4
5	195	128	2,5	2	4
6	255	32	1	1	8
7	255	64	1	1	8

As one could have expected, flow paths from points simulted by SFD and MFD differ greatly (*Figure 5.36*). This could be extremely important for tracing, for example, point source contaminants. The differences would become even greater, because influence maps are created assuming intial TCA of source points to be 1, but their „real“ values may be higher (*Table 5.6.*), so the influence would be higher as well.

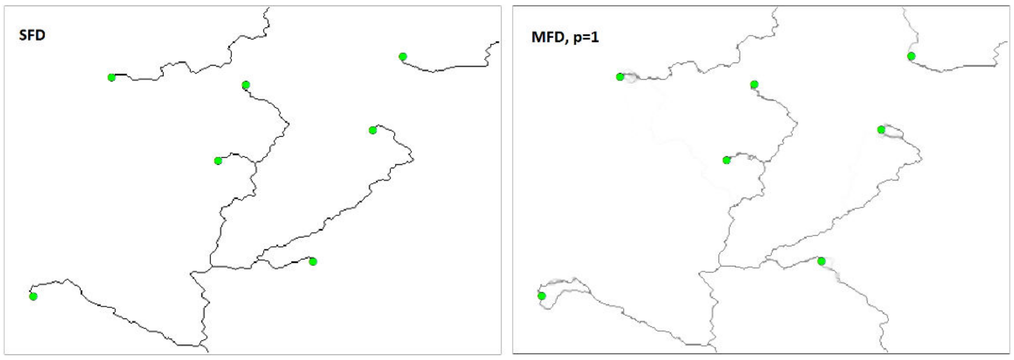


Figure 5-36 Influece maps for 7 points (green dots) produced by SFD (left) and MFD with p=1 (right) algorithms

5.3.5 SCA calculation

For the SCA calculation it is important to define contour width in each pixel. From flow dispersion raster we know that the most majority of pixels for MFD algorithm route flow to 4 direction. Logically, such directions should be adjacent, which could be in case of 2 cardinal and 2 diagonal adjacent directions. This assumption is actually confirmed by MFD directions raster, median code of which is 30 (13319 cells) and next to median code 15 (13218 cells). Using *MFD code calculator* we found out that these codes indeed represents 4 directions, 2 of which diagonal and 2 cardinal.

For current DEM with grid size 90 m, we calculated theoretical range of contour width by different methods, proposed in literature (*Table 5.7.*).While Quinn's and Wolock&McCabe's contour lengths seem to be more or less similar, Freeman's definition will give contour length 2,8 times higher for the same pixels, thus giving 2,8 times lower SCAs values. Two methods for SFD8 contour length definitions do not differ that drastically. However, the comparison of minimum contour length, which is for MFD algorithm is always part of raster with single flow to one cardinal neighbor, shows that SFD8 will produce almost 3 times higher values of contour length, than Quinn's and Wolock&McCabe's contour lengths of MFD8 algorithm.

Table 5-7Possible range of contour widths calculated by different methods for the DEM with grid size 90 m

Calculation Method		Contour length in one pixel, m		
		Minimum	Most frequent	Maximum
MFD8	Freeman's	90	435	869
	Quinn's	32	154	307
	Wolock&McCabe's	36	180	360
SFD8	Equal for all directions	90	90	90
	Different for diagonal and cardinal directions	90	90 or 127	127

We found out previously, that single flow paths in flow direction patterns produced by MFD8 algorithm probably occur in the stream channels parts of terrain where there is high values of flow accumulation. From the other hand, we saw that TCAs with high values are predicted similarly by SFD8 and MFD8. Thus, we can assume that SCAs in such parts of terrain predicted by SFD8 would be almost 3 times lower than those predicted by MFD8.

Logically, the coarser raster resolution, the greater absolute differences in contour length width calculated by different methods. Accordingly, the different SCAs patterns will be derived in inverse dependency to contour length.

For determination of SCAs by MFD we used the same TCA raster derived by MFD (Freeman's) with parameter p=1, but different contour lengths (Freeman's, Quinn's and Wolock&McCabe's). Thus, SCAs calculated by these 3 methods differed proportionally to differences in contour lengths: SCA range derived by Quinn's and Wolock&McCabe's contour length definitions look more or less similar, while Freeman's SCA values seem to be almost 3 times lower (*Table 5.8*).

For the calculation of SCAs by SFD8 algorithm we used TCA raster derived by SFD8 algorithm and divided it by pixel size, thus choosing approach of equal contour length for both cardinal and diagonal directions.

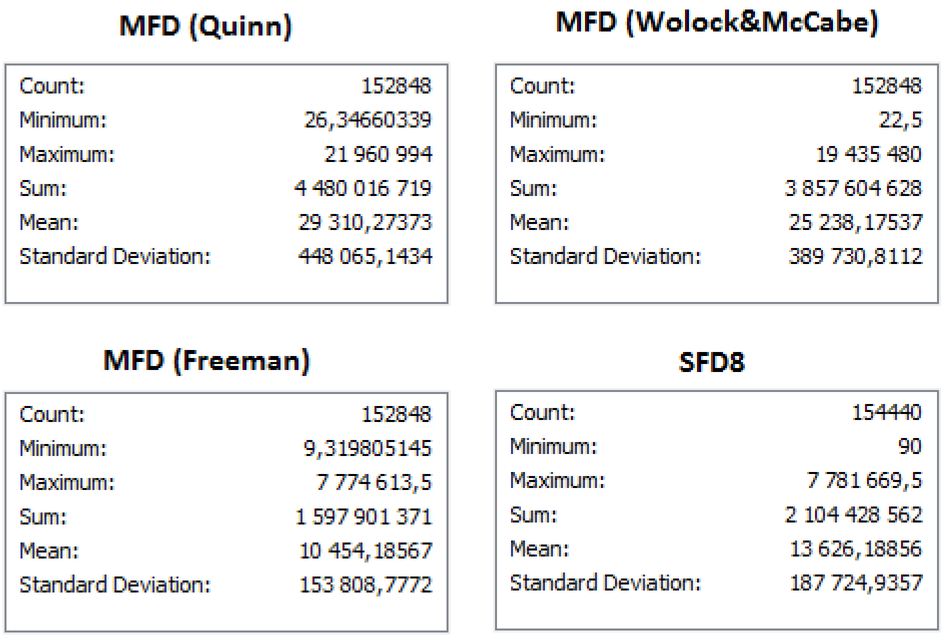
Basic descriptive statistics of resulted SCAs, however, differ from those predicted by MFD algorithm, in different manner than contour length (*Table 5.8*).

Table 5-8 SCA ranges derived by MFD with 3 different contour length methods and SFD

Calculation Method		SCA, m		
		Minimum	Mean	Maximum
MFD8	Freeman's	9	10454	7774614
	Quinn's	26	29310	21960994
	Wolock&McCabe's	23	25238	19435480
SFD8	Equal for all directions	90	13626	7781670

Basic SFD8 SCA statistics, except of *Minimum*, are similar to those derived by MFD with Freeman's contour length definition, being almost 3 times lower, than those derived by MFD of Quinn and Wolock&McCabe (*Table 5.9*).

Table 5-9 Basic statistical characteristics of SCA values, m, derived by SFD and MFD with different contour length


MFD (Quinn)

```
Count:                       152848
Minimum:                26,34660339
Maximum:                 21 960 994
Sum:                  4 480 016 719
Mean:                  29 310,27373
Standard Deviation:   448 065,1434
```

MFD (Wolock&McCabe)

```
Count:                       152848
Minimum:                       22,5
Maximum:                 19 435 480
Sum:                  3 857 604 628
Mean:                  25 238,17537
Standard Deviation:   389 730,8112
```

MFD (Freeman)

```
Count:                       152848
Minimum:                9,319805145
Maximum:                7 774 613,5
Sum:                  1 597 901 371
Mean:                  10 454,18567
Standard Deviation:   153 808,7772
```

SFD8

```
Count:                       154440
Minimum:                         90
Maximum:                7 781 669,5
Sum:                  2 104 428 562
Mean:                  13 626,18856
Standard Deviation:   187 724,9357
```

Comparison of spatial SCA patterns, however, reveals, that SFD8 and Freeman's MFD behave much differently, while Quinn's and Wolock&McCabe's MFD remain pretty similar. SCAs produced by MFD (Freeman) seem to be unrealistic, because there are large area occupied by SCAs with values less then 90 m, these are cells draining only area of 1 pixel (*Figure 5.37*).

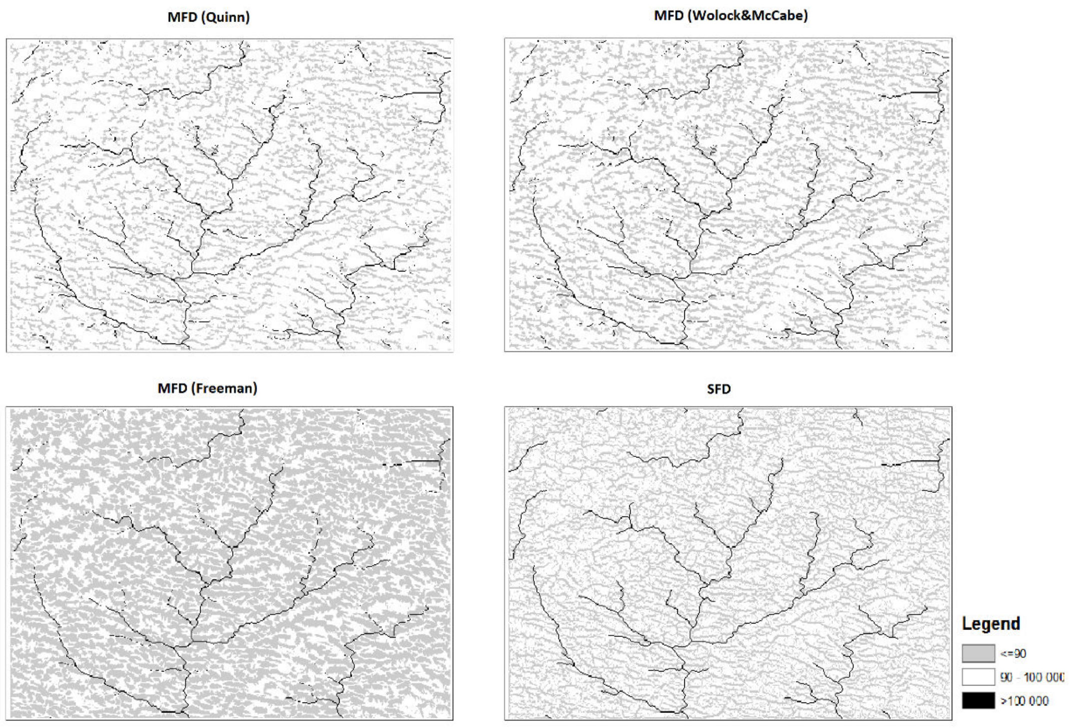


Figure 5-37 SCA (m) patterns calculated by SFD and MFD with different countour width methods

As in case of TCAs values, SFD algorithm produce great amount of cells with SCA equals to 90 m, that is value for a pixel, which drain only itself, which seem to be unrealistic for such type of terrain with diverse topography (differences in elevations are about 400 m).

Comparison of SCA distribution in logarithmic scale reveals the same tendency as in case of TCA: all MFD algorithms produce smooth distribution of SCAs, while SFD8 SCA values varies discretely, which seem to be unrealistic in real terrain, as all properties of any natural body should be distributed continuously. The discrete nature of SD8 SCA patterns and smooth features of MFD patterns can be seen on logarithmically transformed raster images as well, but we didn't iclude them as redundant (everything can be found in supplementary .mxd file). Differences between MFD variations are not so clear though.

6 Discussion

Created toolbox with implemented MFD algorithm proved to be functional for the terrain analysis of hydrologically correct DEM. Even for thecoarse-resolution DEM we found out significant differences in TCA and SCA patterns produced by ArcGIS built-in SFD algorithm and MFD algorithm implemented by author of the current diploma thesis.

We suppose that SFD algorithm is better for river network extraction then MFD approach, because it tends to produce connected network of pixels with high values of TCA, while MFD does not. However, MFD algorithm may be better for the calculation of SCA, especially on divergent hillslopes. Another promising application field of MFD approach could be in modelling of contaminants transport.

Choosing of flow partitioning parameter *p* allows user to control extent of flow divergence in calculation of TCA by developed *Flow Accumulation by MFD tool*. However, more studies are needed on optimum *p* values for different relief types and different DEM resolution. Created tool could be further developed by implementation of spatially varying*p* value methods.

Further studies could be also concentrated on development of other tools for hydrological analysis for calculating of different hydrological indices dependent on flow accumulation values (e.g. TWI). Implementation of other flow routing approaches in ArcGIS environment will greatly extend abilities of basic *Hydrology toolset*. Another interesting field of research is application of SCA concept in pysically-based modelling of natural distribution of vegettion or soil properties.

Thus, developed tools could serve as valuable means for different research purposes and can be publically available at FES CULS for students and researchers.

7 Conclusion

All assigned tasks were accomplished, the main goal of the present diploma thesis was achieved:

1) Literature review on basic flow routing algorithms and their application was written
2) Chosen Multiple Flow Direction algorithm was implemented in ArcGIS as a user-friendly toolbox with 7 tools for hydrological analysis
3) Results of MFD implementation were compared with those derived by SFD

Main conclusions from comparison of SFD and MFD algorithms behavior:

1) There are significant differences between TCA and SCA patterns calculated by SFD and MFD algorithm
2) MFD is recommended to use for the calculation of flow accumulation and SCA for subsequent use for the calculation of different hydrological indices
3) SFD is recommended to use for channel delineation
4) For the calculation ofSCA by MFD, it is not recommended to use Contour width calculated by Freeman method

8 References

1. Albani, M. and Klinkenberg, B. (2003). A Spatial Filter for the Removal of Striping Artifacts in Digital Elevation Models. *Photogrammetric Engineering & Remote Sensing*, 69(7), pp.755-765.
2. Barták, V. (2008). Algoritmy pro zpracování digitálních modelů terénu s aplikacemi v hydrologickém modelování. Diploma thesis. pp. 202
3. Barták, V. (2009). How to extract river networks and catchment boundaries from DEM: a review of digital terrain analysis techniques. *Journal of Landscape Studies*, *2*, 57-68.
4. Beasley, D. B., & Huggins, L. F. (1978, December). ANSWERS: A hydrologic/water quality simulator for watershed research. In *Proceedings of the 10th conference on Winter simulation-Volume 2* (pp. 507-515). IEEE Computer Society Press.
5. *Binary heap*. (2016). *Wikipedia*. Retrieved 3 April 2016, from https://en.wikipedia.org/wiki/Binary_heap
6. Chirico, G., Western, A., Grayson, R., & Blöschl, G. (2005). On the definition of the flow width for calculating specific catchment area patterns from gridded elevation data. *Hydrol. Process.*, *19*(13), 2539-2556. http://dx.doi.org/10.1002/hyp.5730
7. Costa-Cabral, M. C., & Burges, S. J. (1994). Digital elevation model networks (DEMON): A model of flow over hillslopes for computation of contributing and dispersal areas. *Water resources research*, *30*(6), 1681-1692.
8. Desmet, P. J. J., and Gerard Govers (1996). Comparison of routing algorithms for digital elevation models and their implications for predicting ephemeral gullies. *International Journal of Geographical Information Science* 10(3): 311-331.
9. *Digital Topography: Should you choose a TIN or raster interpolation of the landscape?*. (2016). *Vignette Collection*. Retrieved 9 April 2016, from http://serc.carleton.edu/vignettes/collection/42681.html
10. *Esri Software for GIS Students*. (2016). *Esri.com*. Retrieved 2 April 2016, from http://www.esri.com/industries/apps/education/offers/promo/
11. Freeman, T. G. (1989). Drainage with divergent flow over a regular grid: Proc. 8th Biennial Conf. Simulation Society of Australia, Canberra, p. 160-165.
12. Freeman, T. G. (1991). Calculating catchment area with divergent flow based on a regular grid. *Computers & Geosciences*, *17*(3), 413-422.
13. Gallant, J. C., & Wilson, J. P. (1996). TAPES-G: a grid-based terrain analysis program for the environmental sciences. Computers & Geosciences, 22(7), 713-722.
14. Garbrecht, J., & Martz, L. W. (2000). Digital elevation model issues in water resources modeling. *Hydrologic and hydraulic modeling support with geographic information systems*, 1-28.
15. Gruber, S., & Peckham, S. (2009). Land-surface parameters and objects in hydrology. *Developments in Soil Science*, *33*, 171-194.

16. Holmgren, P. (1994). Multiple flow direction algorithms for runoff modelling in grid based elevation models: an empirical evaluation. *Hydrological processes*, *8*(4), 327-334.
17. Jenson, S. K., & Domingue, J. O. (1988). Extracting topographic structure from digital elevation data for geographic information system analysis. Photogrammetric engineering and remote sensing, 54(11), 1593-1600.
18. KIM S., LEE H. (2004) A digital elevation analysis: a spatially distributed flow apportioning algorithm. *Hydrological Processes* 18(10): 1777-1794.
19. Lea, N. L. (1992). An aspect driven kinematic routing algorithm. *Overland flow: hydraulics and erosion mechanics*, *147*, 175.
20. Mach, R., & Petschek, P. (2007). *Visualization of digital terrain and landscape data: a manual.* Springer Science & Business Media. 364
21. Mitasova, H., Hofierka, J., Zlocha, M., & Iverson, L. R. (1996). Modelling topographic potential for erosion and deposition using GIS. *International Journal of Geographical Information Systems*, *10*(5), 629-641.
22. Mitasova, H., Mitas, L., Brown, W. M., Gerdes, D. P., Kosinovsky, I., & Baker, T. (1995). Modelling spatially and temporally distributed phenomena: new methods and tools for GRASS GIS. *International Journal of Geographical Information Systems*, *9*(4), 433-446.
23. Mitášová, H., & Hofierka, J. (1993). Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis.*Mathematical Geology*, *25*(6), 657-669.
24. Mitchell, A. (2012). The Esri Guide to GIS Analysis, Volume 3: Modeling Suitability, Movement, and Interaction. ESRI Press. 419 pp
25. Moore, I. D., Grayson, R. B., & Ladson, A. R. (1991). Digital terrain modelling: a review of hydrological, geomorphological, and biological applications. *Hydrological processes*, *5*(1), 3-30.
26. Novak, P. (2015).Tvorba nástroje pro digitální analýzu terénu. Diploma thesis. pp. 54
27. O'Callaghan, J. F., & Mark, D. M. (1984). The extraction of drainage networks from digital elevation data. *Computer vision, graphics, and image processing*,*28*(3), 323-344.
28. Orlandini, S., Moretti, G., Franchini, M., Aldighieri, B., & Testa, B. (2003). Path-based methods for the determination of nondispersive drainage directions in grid-based digital elevation models. *Water resources research*, *39*(6).
29. Peucker, T. K. & Douglas, D. H., 1975. Detection of Surface-Specific Points by Local Parallel Processing of Discrete Terrain Elevation Data. Computer Graphicsand Image Processing, 4(4), p. 375–387.
30. Pidwirny, M. (2006). "Introduction to Geographic Information Systems". *Fundamentals of Physical Geography, 2nd Edition*. Date Viewed.
31. Pilesjö, P., Zhou, Q., & Harrie, L. (1998). Estimating flow distribution over digital elevation models using a form-based algorithm. *Geographic Information Sciences*,*4*(1-2), 44-51.
32. Quinn, P. F. B. J., Beven, K., Chevallier, P., & Planchon, O. (1991). The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. *Hydrological processes*, *5*(1), 59-79.
33. Quinn, P. F., Beven, K. J., & Lamb, R. (1995). The In (a/tan/β) index: How to calculate it and how to use it within the Topmodel framework. *Hydrological processes*, *9*(2), 161-182.

34. Resources.arcgis.com, (2015). ArcGIS Help (10.2, 10.2.1, and 10.2.2). [online] Available at: http://resources.arcgis.com/en/help/main/10.2/index.html#/An_overview_of_the_Hydrology_tools/009z0000004w000000/ [Accessed 16 Dec. 2015].
35. Russell, E., Kumler, M., & Ochis, H. (1995). Identifying and removing systematic errors in USGS DEMs. In *Proceeding of GIS in the Rockies conference, Denver, CO.*
36. Seibert, J., and B. L. McGlynn (2007), A new triangular multiple flow direction algorithm for computing upslope areas from gridded digital elevation models, Water Resour. Res., *43*, W04501, doi:10.1029/2006WR005128.
37. Sulebak, J. R. (2000). Applications of digital elevation models. *DYNAMAP Project*, 11. [online] Available at: http://gisknowledge.net/topic/terrain_modelling_and_analysis/sulebak_dem_applications_00.pdf
38. Tarboton, D. G. (1997). A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water resources research*, *33*(2), 309-319.
39. Utah State Unversity, D. (2016). *David Tarboton: Hydrology Research Group-Terain Analysis Using Digital Elevation Models (TauDEM)*. *Hydrology.usu.edu.* Retrieved 30 March 2016, from http://hydrology.usu.edu/taudem/taudem5/license.html
40. *What version of Python is used in ArcGIS?*. (2016). *Support.esri.com*. Retrieved 10 April 2016, from http://support.esri.com/fr/knowledgebase/techarticles/detail/43889
41. Wilson, J. P., & Gallant, J. C. (2000). Digital terrain analysis. *Terrain analysis: Principles and applications*, 1-27.
42. Wilson, J. P., AGGETT, G., & Yongxin, D. E. N. G. (2008). Water in the landscape: a review of contemporary flow routing algorithms. In *Advances in digital terrain analysis*. Springer Berlin Heidelberg, 213-236
43. Wise, S. (2013). *GIS Fundamentals, Second Edition*. Abingdon: CRC Press. 290 pp
44. Wolock, D., & McCabe, G. (1995). Comparison of Single and Multiple Flow Direction Algorithms for Computing Topographic Parameters in TOPMODEL. *Water Resources Research*, *31*(5), 1315-1324. http://dx.doi.org/10.1029/95wr00471
45. Zhou, Q., Lees, B., & Tang, G. A. (Eds.). (2008). *Advances in digital terrain analysis*. Berlin/Heidelberg, Germany: Springer, 3-10

Appendix

A.1. MFD.py

```
# -*- coding: cp1250 -*-

# Multiple Flow Direction

# Created by Daria Rapoport, 2016/04,

# based on script created by Bc. Petr Novák, 2015/04:

# module "rta" and functions "slope" and "neighbors_z" are taken unchanged.

# Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

"""

The script contains 2 functions: neighbors_z and slope.

Neighbors_z is looking for the elevations of the neighboring cells.

Slope calculates slope to each of the neighboring cells.

Input DEM is transfered to an array. Flow directions to all neighbors

with positive slopes for each element of DEM array(i.e. each cell of DEM)

are then determined. Simultaneously, control of sink areas is performed.

"""

import os, math, arcpy

import numpy as np

import rta as rt

arcpy.env.overwriteOutput = True

# Function searches for the elevations of 8 neighbors of a cell

def neighbors_z(array,x,y):

    n1=array[x,y+1]

    n2=array[x+1,y+1]

    n4=array[x+1,y]

    n8=array[x+1,y-1]
```

```
  n16=array[x,y-1]

  n32=array[x-1,y-1]

  n64=array[x-1,y]

  n128=array[x-1,y+1]

  return n1, n2, n4, n8, n16, n32, n64, n128

# Function calculates slopes to each of 8 neighbors of a selected cell

def slope (cell, nb, vCell):

  slp=[]

  for k in range(8):

    if k%2==0:

      s=(cell-nb[k])/vCell

    else:

      s=(cell-nb[k])/vCellSqrt

    slp.append(s)

  return slp

arcpy.AddMessage("Reading data from DEM ... ")

# Input:

# Raster DEM

in_dem = sys.argv[1]

# Raster DEM is transfered to np.array

ar_dem = rt.rta(in_dem)

XMax = ar_dem[1]

YMax = ar_dem[2]

vCell = ar_dem[3]

LeftX = ar_dem[4]

LextY = ar_dem[5]

vCellSqrt = vCell*math.sqrt(2)
```

```
# Output:

# Raster of flow directions

out_ras = sys.argv[2]

arcpy.AddMessage("Calculations begin...")

# Creation of array of zeros

dir_aray = np.zeros([XMax,YMax])

# Basic 8-neighbor values for determination of flow direction

rastr_dir=[1,2,4,8,16,32,64,128]

# Going through all the raster

for x in range (1,XMax-1):

   for y in range (1,YMax-1):

       # Calculates slopes to 8 neighbors of a cell

       slp=slope(ar_dem[0][x][y],neighbors_z(ar_dem[0],x,y),vCell)

       #Pits and flat areas error

       if (max(slp)) <= 0:

           arcpy.AddError(str("Raster error(flat area or sink),maximum slope value =
")+str(max(slp)))

           break

               #Flow direction of a given cell is a sum of 8 possible direction values

       #of all positive slopes (maximum 253 unique combinations,

       #e.g. if we have 2 positive slopes from a cell to the North(direction vlue=1)

       #and North-West (direction value=128), direction value of a cell = 1+128=129)

       posslp=0

       for i in range(8):

           if slp[i]>0:

               posslp=posslp+rastr_dir[i]

       dir_aray[x,y]=posslp
```

```
    # Pits and flat areas error
    if (max(slp)) <= 0:
        arcpy.AddError(str("Raster error(flat area or sink),maximum slope value = ")+str(max(slp)))
        break
arcpy.AddMessage("Creating of flow direction raster...")
#Creating of flow direction raster
new_raster = arcpy.NumPyArrayToRaster(dir_aray,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)
new_raster.save(out_ras)
```

A.2. Fdisp.py

```
# -*- coding: cp1250 -*-
# Flow Dispersion
# Created by Daria Rapoport, 2016/04,
# Faculty of Environmental Sciences
# Czech University of Life Sciences Prague
"""
The script calculates flow dispersion - number of cells to which flow is routed
from the current cell.
Input Flow Direction raster is transfered to an array, for each cell of which
number of non-zero bits (equivalent to the number of directions towards neighboring cells)
is calculated and converted to output 'flow dispersion' raster.
"""
import os, arcpy
import numpy as np
import rta as rt
arcpy.env.overwriteOutput = True
arcpy.AddMessage("Reading data from DEM ... ")
# Input:
# Raster of flow directions with single flow direction codes used in ArcGIS
#and their summation for multiple flow directions
in_dem = sys.argv[1]
# Raster of flow directions is transfered to np.array
ar_dem = rt.rta(in_dem)
XMax = ar_dem[1]
YMax = ar_dem[2]
vCell = ar_dem[3]
LeftX = ar_dem[4]
LextY = ar_dem[5]
# Output:
# Raster of flow dispersion (raster of numbers of directions form one cell)
out_ras = sys.argv[2]
arcpy.AddMessage("Calculations begin...")
# Creation of array of zeros - basis for the output creation
dir_num = np.zeros([XMax,YMax])
   # Going through all the input raster
for x in range (1,XMax-1):
   for y in range (1,YMax-1):
      dirn=0
      for i in range(8):


            try:
               if bin(int(ar_dem[0][x][y]))[-i-1]=='1':
                  dirn+=1
            except:
               pass
```

```
    dir_num[x,y]=dirn

arcpy.AddMessage("Creating of flow dispersion raster...")

#Creating of flow dispersion raster

new_raster = arcpy.NumPyArrayToRaster(dir_num,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)

new_raster.save(out_ras)
```

A.3. FSmfd.py

```
# -*- coding: cp1250 -*-

# Multiple Flow Simulation surface drainage

# Created by Daria Rapoport, 2016/04,

# based on script created by Bc. Petr Novák, 2015/04:

# modules "rta" and "Heap"; functions "slope","neighbors_xy" and "neighbors_z" are taken unchanged.

# Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

"""

The script contains 3 functions: neighbors_z, neighbors_xy and slope.

Neighbors_z is looking for the elevations of the neighboring cells.

Neighbors_xy searches for the coordinates [x,y] of the neighoring cells.

Slope calculates slope to each of the neighboring cells.

Input DEM is transfered to an array. Initial array of flow accumulation is created (array of ones + weights if provided).

Heap of vectors(x,y,z) sorted by the value of elevation "z" is created.

Starting with a cell with maximum elevation flow directions to all neighbors with positive slopes are determined.

Accumulation value of a cell in the direction of a flow is increased

by an accumulation value of the currently processed cell according to the weight of its slope value in all positive neighboring slopes.

"""

import os, math, arcpy

import numpy as np
import rta as rt
import Heap as h
arcpy.env.overwriteOutput = True
# Function searches for the elevations of 8 neighbors of a cell
def neighbors_z(array,x,y):
    n1=array[x,y+1]
    n2=array[x+1,y+1]
    n4=array[x+1,y]
    n8=array[x+1,y-1]
    n16=array[x,y-1]
    n32=array[x-1,y-1]
    n64=array[x-1,y]
    n128=array[x-1,y+1]
    return n1, n2, n4, n8, n16, n32, n64, n128
# Function calculates slopes to each of 8 neighbors of a selected cell
def slope (cell, nb, vCell):
    slp=[]
    for k in range(8):
        if k%2==0:
            s=(cell-nb[k])/vCell
        else:
            s=(cell-nb[k])/vCellSqrt
        slp.append(s)
    return slp
# Function searches for the coordinates [x,y] of the neighoring cells
def neighbors_xy(x,y):
    n1=[x,y+1]
    n2=[x+1,y+1]
    n4=[x+1,y]
    n8=[x+1,y-1]
    n16=[x,y-1]
    n32=[x-1,y-1]
    n64=[x-1,y]
    n128=[x-1,y+1]
    return n1,n2,n4,n8,n16,n32,n64,n128
arcpy.AddMessage("Reading data from DEM ... ")
# Input:
# Raster DEM
in_dem = sys.argv[1]
# Raster DEM is transfered to np.array
aar_dem = rt.rta(in_dem)
ar_dem = aar_dem[0]
XMax = aar_dem[1]
YMax = aar_dem[2]
vCell = aar_dem[3]
LeftX = aar_dem[4]
LextY = aar_dem[5]
vCellSqrt=vCell*math.sqrt(2)
del aar_dem
# Output:
# Raster of flow accumulation
out_aku = sys.argv[2]
#user-specified parameter p controlling flow divergency (Default p=1,1)
(Freeman, 1991)
p = float(sys.argv[3])
# Raster of weights
if sys.argv[4] == "true":
    self_scale = sys.argv[5]
    # Transfer of weights raster to np.array
```

```
        ar_scale=rt.rta(self_scale)
    arcpy.AddMessage( "Data is loaded, the calculation begins ...")
    ha=[]
    # Creating of heap of vectors(x,y,z) sorted by the value of elevation "z"
    for xi in range (1,XMax-1):
        for yi in range (1,YMax-1):
            h.inz_el (ha, (xi, yi, ar_dem[xi,yi]), index=2)
    # Initial np.array for accumulation values
    if sys.argv[4]== "false":
        ar_aku=np.ones([XMax,YMax])
    else:
        ar_aku=np.ones([XMax,YMax])+ar_scale[0]
    arcpy.AddMessage( "Calculation of flow accumulation begins...")
    # Accumulation value of a cell in the direction of a flow (neighbor cell with positive slope)
    #will be increased by an accumulation value of an inflow cell (current "central" cell)
    #Calculation starts from the cells with the highest elevations (using binary heap)
    while len(ha)>0:
        x= ha[0][0]
        y= ha[0][1]
        if x in range (1,XMax-1):
            if y in range (1,YMax-1):
                slp=slope(ha[0][2],neighbors_z(ar_dem,x,y),vCell)
                slsum=0
                posslop=[]
                posnb=[]
                #Determination of the neighbors with positive slopes
                for i in range(8):
                    if slp[i]>0:
```

```
                slsum=slsum+slp[i]**p
                posslop.append(slp[i])
                posnb.append(i)
            for el in posnb:
                # Neighboring cell in the direction of flow
                nxy=neighbors_xy(x,y)[el]
                #Its flow accumulation value=its current flow accumulation + flow accum.from "central" cell
ar_aku[nxy[0],nxy[1]]=ar_aku[nxy[0],nxy[1]]+ar_aku[x,y]*(posslop[posnb.index(el)]**p/slsum)
        # Deleting of maximum from the heap and putting there next maximum
        h.ret_max(ha,2)
    arcpy.AddMessage( "Creating of flow accumulation raster...")
    # Creating of flow accumulation raster
    new_raster = arcpy.NumPyArrayToRaster(ar_aku,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)
    new_raster.save(out_aku)
```

A.4. Influence_map.py

```
    # -*- coding: cp1250 -*-
    # Influence Map
    # Created by Daria Rapoport, 2016/04,
    # based on script created by Bc. Petr Novák, 2015/04:
    # modules "rta" and "Heap"; functions "slope","neighbors_xy" and "neighbors_z" are taken unchanged.
    # Faculty of Environmental Sciences
    # Czech University of Life Sciences Prague
    """
    The script maps where flow goes from the input pixels (coordinates in column, row terms) and how it is dispersed.
    User can choose MFD8 (Freemn, 1991) or SFD8 (O'Callaghan&Mark, 1984) algorithm for flow routing.
```

The script contains 3 functions: neighbors_z, neighbors_xy and slope.

Neighbors_z is looking for the elevations of the neighboring cells.

Neighbors_xy searches for the coordinates [x,y] of the neighoring cells.

Slope calculates slope to each of the neighboring cells.

Input DEM is transfered to an array. Initial array of flow accumulation is created (array of zeroes).

Input coordinates of source pixels are used to assign flow accumulation values to ones for the input source pixels.

Heap of vectors(x,y,z) sorted by the value of elevation "z" is created. Slope is calculted.

If SFD8 was chosen, flow is routed towards direction of maximum slope, starting with a cell with maximum elevation.

In case of MFD8, also starting with a cell with maximum elevation, flow directions to all neighbors with positive slopes are determined.

Accumulation value of a cell in the direction of a flow is increased

by an accumulation value of the currently processed cell according to the weight of its slope value in all positive neighboring slopes.

```
"""

import os, math, arcpy

import numpy as np

import rta as rt

import Heap as h

arcpy.env.overwriteOutput = True

# Function searches for the elevations of 8 neighbors of a cell

def neighbors_z(array,x,y):

    n1=array[x,y+1]

    n2=array[x+1,y+1]

    n4=array[x+1,y]

    n8=array[x+1,y-1]

    n16=array[x,y-1]

    n32=array[x-1,y-1]

    n64=array[x-1,y]
    n128=array[x-1,y+1]
    return n1, n2, n4, n8, n16, n32, n64, n128
# Function calculates slopes to each of 8 neighbors of a selected cell
def slope (cell, nb, vCell):
    slp=[]
    for k in range(8):
        if k%2==0:
            s=(cell-nb[k])/vCell
        else:
            s=(cell-nb[k])/vCellSqrt
        slp.append(s)
    return slp
# Function searches for the coordinates [x,y] of the neighoring cells
def neighbors_xy(x,y):
    n1=[x,y+1]
    n2=[x+1,y+1]
    n4=[x+1,y]
    n8=[x+1,y-1]
    n16=[x,y-1]
    n32=[x-1,y-1]
    n64=[x-1,y]
    n128=[x-1,y+1]
    return n1,n2,n4,n8,n16,n32,n64,n128
arcpy.AddMessage("Reading data from DEM ... ")
# Input:
# Raster DEM
in_dem = sys.argv[1]
# Raster DEM is transfered to np.array
aar_dem = rt.rta(in_dem)
```

```
ar_dem = aar_dem[0]
XMax = aar_dem[1]
YMax = aar_dem[2]
vCell = aar_dem[3]
LeftX = aar_dem[4]
LextY = aar_dem[5]
vCellSqrt=vCell*math.sqrt(2)
del aar_dem
# Output:
# Raster of flow accumulation
out_aku = sys.argv[2]
#user-specified parameter p controlling flow divergency (Default p=1)
(Freeman, 1991)
p = float(sys.argv[5])
#Input pixel coordinates is retyped to the array of integers
xval=sys.argv[7].split(';')
yval=sys.argv[6].split(';')
xvalues=[]
yvalues=[]
for xv in xval:
   xvalues.append(int(xv))
for yv in yval:
   yvalues.append(int(yv))
arcpy.AddMessage( "Data is loaded, the calculation begins ...")
ha=[]
# Creating of heap of vectors(x,y,z) sorted by the value of elevation "z"
for xi in range (1,XMax-1):
   for yi in range (1,YMax-1):
      h.inz_el (ha, (xi, yi, ar_dem[xi,yi]), index=2)
#Creating initial flow accumulation raster
```

```
ar_aku=np.zeros([XMax,YMax])

for i in range(len(xvalues)):

   ar_aku[xvalues[i],yvalues[i]]=1

arcpy.AddMessage( "Calculation of flow accumulation begins...")

# Accumulation value of a cell in the direction of a flow (neighbor cell with positive slope)

#will be increased by an accumulation value of an inflow cell (current "central" cell)

#Calculation starts from the cells with the highest elevations (using binary heap)

while len(ha)>0:

   x= ha[0][0]

   y= ha[0][1]

   if x in range (1,XMax-1):

      if y in range (1,YMax-1):

        slp=slope(ha[0][2],neighbors_z(ar_dem,x,y),vCell)

        #Checking SFD8 in the tool dialog box

        if sys.argv[3] == "true":

           indx=slp.index(max(slp))

           nxy=neighbors_xy(x,y)[indx]

           ar_aku[nxy[0],nxy[1]]=ar_aku[nxy[0],nxy[1]]+ar_aku[x,y]

       #Checking MFD in the tool dialog box

        if sys.argv[4] == "true":

            slsum=0

            posslop=[]

            posnb=[]

            #Determination of the neighbors with positive slopes

            for i in range(8):

               if slp[i]>0:

                   slsum=slsum+slp[i]**p

                   posslop.append(slp[i])
```

```
                                posnb.append(i)
                        for el in posnb:
                            # Neighboring cell in the direction of flow
                            nxy=neighbors_xy(x,y)[el]
                            #Its flow accumulation value=its current flow accumulation + flow accum.from "central" cell
ar_aku[nxy[0],nxy[1]]=ar_aku[nxy[0],nxy[1]]+ar_aku[x,y]*(posslop[posnb.index(el)]**p/slsum)
                # Deleting of maximum from the heap and putting there next maximum
                h.ret_max(ha,2)
        arcpy.AddMessage( "Creating of flow influence raster...")
        # Creating of flow accumulation raster
        new_raster = arcpy.NumPyArrayToRaster(ar_aku,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)
        new_raster.save(out_aku)
```

A.5. nColRow.py

```
# -*- coding: cp1250 -*-
# Created by Daria Rapoport, 2016/04
# Script module "rta" is credited to Bc. Petr Novák, 2015/04:
# Faculty of Environmental Sciences
# Czech University of Life Sciences Prague
""
The script calculates the number of column and row of pixels of the input raster,
producing 2 rasters of column and row numbers of each pixel.
Input raster is coverted to ndarray.
Ndarray of columns and rows numbers of the input raster is created using numpy.indices method.
Arrays of columns and rows are extracted from ndarray by indexing and converted to rasters.
 ""
```

```
import os, arcpy

import numpy as np

import rta as rt

arcpy.env.overwriteOutput = True

arcpy.AddMessage("Reading data from the input raster ... ")

# Input:

# Raster for which columns and rows will be defined

in_raster = sys.argv[1]

# Raster DEM is transfered to np.array

raster = rt.rta(in_raster)

XMax = raster[1]

YMax = raster[2]

vCell = raster[3]

LeftX = raster[4]

LextY = raster[5]

# Outputs:

# Raster of row numbers

out_rows = sys.argv[3]

# Raster of column numbers

out_cols = sys.argv[2]

arcpy.AddMessage("Calculation begins...")

#Creation of numpy array of indices

grid=np.indices((XMax,YMax))

nrows=grid[0]

ncols=grid[1]

#Writing the outputs

row_raster =
arcpy.NumPyArrayToRaster(nrows,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)
```

```
row_raster.save(out_rows)

col_raster =
arcpy.NumPyArrayToRaster(ncols,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-9999)

col_raster.save(out_cols)
```

A.6. PaD.py

```
# -*- coding: cp1250 -*-

# Only part of the script is created by Daria Rapoport, 2016/04,

# based on the script created by Bc. Petr Novák, 2015/04:

# # Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

#Script creates raster of Peucke and Douglass' weights

# Unchanged part of Novak

# Peucke and Douglass

# Created by Bc. Petr Novák, 2015/04

# Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

'''

Skript obsahuje dvě funkce: nPaD a PaD.

nPaD hledá elevaci a souřadnice tří sousedních buněk.

PaD načte vstupní DEM, převede jej na array a následně

prochází array za pomocí okna o velikosti 2x2 buňky.

Porovná tak elevaci každou buňky DEM s jejími třemi sousedy.

Buňky, které nebyly ani jednou označeny za nejvyšší

z dané čtveřice, jsou s hodnotou 1 zapsány do výsledného rastru.

'''

import os, math, arcpy, sys

import numpy as np

import rta as rt

arcpy.env.overwriteOutput = True

# x,y souřadnice a elevace tří sousedních buněk
```

```
def nPaD(array,x,y):
  n0=array[x,y+1] #elevace
  n3=[x,y+1]     #souřadnice
  n1=array[x+1,y+1]
  n4=[x+1,y+1]
  n2=array[x+1,y]
  n5=[x+1,y]
  return n0, n1, n2, n3, n4, n5
def PaD(in_dem):
  arcpy.AddMessage("PaD: Načítám data...")
  # Načte rastr DEM do np.array
  inDEM = rt.rta(in_dem)
  XMax = inDEM[1]
  YMax = inDEM[2]
  vCell = inDEM[3]
  LeftX = inDEM[4]
  LextY = inDEM[5]
  arcpy.AddMessage( "PaD: Data načtena, začíná výpočet")
  # Nový np.array hodnot = 1
  PaD_aray = np.ones([XMax,YMax])
  # Prohledá celý rastr
  for x in range (0,XMax-1):
    for y in range (0,YMax-1):
      nb=nPaD(inDEM[0],x,y)
      # Porovná čtveřice hodnot a najde nejvyšší elevaci
      if inDEM[0][x][y] > nb[0]:
        if inDEM[0][x][y] > nb[1]:
          if inDEM[0][x][y] > nb[2]:
            # Všem nejvyšším elevacím přiřadí = 0
```

```
                        PaD_aray[x,y] = 0
                    else:
                        PaD_aray[nb[5][0],nb[5][1]] = 0
                elif nb[1] > nb[2]:
                    PaD_aray[nb[4][0],nb[4][1]] = 0
                else:
                    PaD_aray[nb[5][0],nb[5][1]] = 0
            elif nb[0] > nb[1]:
                if nb[0] > nb [2]:
                    PaD_aray[nb[3][0],nb[3][1]] = 0
                else:
                    PaD_aray[nb[5][0],nb[5][1]] = 0
            elif nb[1] > nb[2]:
                PaD_aray[nb[4][0],nb[4][1]] = 0
            else:
                PaD_aray[nb[5][0],nb[5][1]] = 0
    arcpy.AddMessage("PaD: Rastr vah metodou Peucker and Douglas vytvořen")
    return PaD_aray
#Small part created by Daria Rapoport, which creates raster of Peucker and Douglas weights of input DEM raster
# Input:
# Raster DEM
in_dem = sys.argv[1]
# Raster DEM is transfered to np.array
ar_dem = rt.rta(in_dem)
XMax = ar_dem[1]
YMax = ar_dem[2]
vCell = ar_dem[3]
LeftX = ar_dem[4]
```

```
        LextY = ar_dem[5]

        out_weights = sys.argv[2]

        in_raster=PaD(in_dem)

        new_raster                                                                    =
arcpy.NumPyArrayToRaster(in_raster,arcpy.Point(LeftX,LextY),vCell,value_to_nod
ata=-9999)

        new_raster.save(out_weights)
```

A.7. ContourLength.py

```
# -*- coding: utf-8 -*-

# Countour Length

# Created by Daria Rapoport, 2016/04,

# Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

"""

The script calculates contour lengths - sum of orthogonals to flow directions

(towards downslope neighbors), which equals to: 1) grid size for cardinal neighbors

and grid size*√2 for diagonal neighbors (Freeman, 1991); 2)grid size*0,5 for cardinal
neighbors

and grid size*0,354 for diagonal neighbors (Quinn et al., 1991); 3)grid size*0,6 for cardinal
neighbors

and grid size*0,4 for diagonal neighbors (Wlock&McCabe, 1995) or 4) grid size multiplied by
user-specified factors different for cardinal and diagonal directions.

Input Flow Direction raster is transfered to an array, for each cell of which

number of flow directions towards diagonal and cardinal neighboring cells is determined.

Accordingly, sum of contour length is calculated and converted to output raster.

"""

import os, arcpy, math

import numpy as np

import rta as rt

arcpy.env.overwriteOutput = True
arcpy.AddMessage("Reading data from DEM ... ")

# Input:

# Raster of flow directions with single flow direction codes used in ArcGIS

#and their summation for multiple flow directions

in_dem = sys.argv[1]

# Raster of flow directions is transfered to np.array

ar_dem = rt.rta(in_dem)

XMax = ar_dem[1]

YMax = ar_dem[2]

vCell = ar_dem[3]

LeftX = ar_dem[4]

LextY = ar_dem[5]

# Output:

# Raster of Contour Lengths (as defined by Freeman, 1991)

out_ras = sys.argv[2]

#Freeman's (1991) contour length definition

if sys.argv[3] == "true":

    wc=vCell

    wd=math.sqrt(2)*wc

    #Quinn's (1991) contour length definition

elif sys.argv[4] == "true":

    wc=vCell*0.5

    wd=vCell*0.354

#Wolock&McCabe's (1994) contour length definition

elif sys.argv[5] == "true":

    wc=vCell*0.6

    wd=vCell*0.4
```

```
#User-specified contour length definition
elif sys.argv[5] == "true":
  wc=vCell*(sys.argv[6])
  wd=vCell*(sys.argv[7])
else:
  arcpy.AddMessage("Choose one method for the calculation of contou lengths...")
arcpy.AddMessage("Calculations begin...")
# Creation of array of zeros - basis for the output creation
contours = np.zeros([XMax,YMax])
  # Going through all the input raster
for x in range (1,XMax-1):
  for y in range (1,YMax-1):
    contour=0
    for i in range(4):
    #reading of directions and calculation of contour lengths for diagonal and cardinal
neighbors
      try:
        if bin(int(ar_dem[0][x][y]))[-2*i-1]=='1':
          contour+=wc
      except:
        pass
      try:
        if bin(int(ar_dem[0][x][y]))[-2*i-2]=='1':
          contour+=wd
      except:
        pass
    contours[x,y]=contour
arcpy.AddMessage("Creating of contour lengths raster...")
```

```
#Creating of output raster

new_raster =
arcpy.NumPyArrayToRaster(contours,arcpy.Point(LeftX,LextY),vCell,value_to_nodata=-
9999)

new_raster.save(out_ras)
```

A.8. rta.py (taken unchanged from Novak, 2015)

```
# -*- coding: cp1250 -*-
# Raster to Numpy.Array
import os, arcpy
import numpy as np
# Funkce pro převod rastru do numpy.array
def rta (rast):
# Načte rastr
inRas = arcpy.Raster(rast)
# Zjistí jeho souřadnice
lowerLeftX = inRas.extent.XMin
lowerLeftY = inRas.extent.YMin
# Zjistí velikost pixelu
sCell = inRas.meanCellWidth
vCell = inRas.meanCellHeight
# Převede rastr na numpy.array
new_ar=arcpy.RasterToNumPyArray(rast,nodata_to_value=-9999)
# Počet hodnout v řádku
XMax= new_ar.shape[0]
# Počet řádku
YMax= new_ar.shape[1]
      return new_ar, XMax, YMax, vCell, lowerLeftX, lowerLeftY
```

A.9. Heap.py (taken unchanged from Novak, 2015)

```
# -*- coding: cp1250 -*-
# Heap
# Vytvoří binární haldu ze seznamu
def cr_heap (seznam, index=2):
heap=[]
for i in seznam:
inz_el (heap, i, index)
return heap
# Přidá prvek na konec seznamu a nechá ho proskákat haldou
def inz_el (sez, x, index=2):
sez.append (x)
# Index vloženého čísla
j = len(sez)
while j > 1:
# Index nahrazeného prvku
p = j / 2
#Porovná vložený prvek a jeho nadřazený a případně je vymění
if isinstance(sez[p-1],int):
if sez[j-1] > sez[p-1]:
d = sez[j-1]
sez[j-1] = sez[p-1]
```

```
sez[p-1] = d
# Posun v haldě na předchůdce
j = p
else:
break
else:
if sez[j-1][index] > sez[p-1][index]:
d = sez[j-1]
sez[j-1] = sez[p-1]
sez[p-1] = d
# Posun v haldě na předchůdce
j = p
else:
break
# Odstraní prvek z vrcholu haldy a nahradí ho dalším nejvyšším
číslem
def ret_max (heap, index=2):
# Přesune poslední prvek haldy na vrchol
heap[0] = heap[-1]
del heap [-1]
# Index posledního čísla
j = len(heap)-1
# Index porovnávaného čísla
i = 0
# Platí pro haldu obsahující více, než jedno číslo
while j > 0:
# Index následníka
n = 2*(i+1)-1
if n<len(heap):
if isinstance(heap[n] and heap[i],int):
# Kontrola zda existují oba následníci
if n < j:
- 53 -
# Vybere ten větší
if heap [n+1] > heap [n]:
n = n+1
# Je-li číslo menší než následník, tak se vymění
if heap[i] < heap[n]:
d = heap[n]
heap[n] = heap[i]
heap[i] = d
# Posun v haldě na následníka
i = n
else:
break
else:
# Kontrola zda existují oba následníci
if n < j:
# Vybere ten větší
if heap [n+1][index] > heap [n][index]:
n = n+1
# Je-li číslo menší než následník, tak se vymění
if heap[i][index] < heap[n][index]:
d = heap[n]
```

```
heap[n] = heap[i]
heap[i] = d
# Posun v haldě na následníka
i = n
else:
break
else:
break
# Funkce pro vytvoření seřazeného seznamu z existující
seřazené haldy
def cr_list (heap):
ssez = []
# Je-li v haldě alespoň jediná položka
while len (heap)>0:
# vloží největší číslo z haldy na konec seznamu
ssez.append (heap[0])
# a vrátí maximum na začátek haldy
ret_max (heap)
return ssez
```

A.10. MFD_code_calculator.py

```
# Flow Directions from the multiple flow direction codes

# Created by Daria Rapoport, 2016/04,

# Faculty of Environmental Sciences

# Czech University of Life Sciences Prague

""

The script gives the directions from the input array of MFD codes

""

codes=[]

flowdir=['East','South-East','South','South-West','West','North-West','North', 'North-East']

#Input part - just enter flow direction codes into array

MFDcodes=[195,252,207,60,195,255,255]

for el in MFDcodes:

   codes.append(int(el))

for code in codes:

   count=1

   print 'Flow direction code '+str(code)+' equals to:'
```

```
for i in range(8):
        try:
          if bin(code)[-i-1]=='1':
              print str(count)+')'+flowdir[i]+'\n'
              count+=1
        except:
          pass
```

Attachments: content of attached CD

Part of the Diploma work is CD containing text part of diploma thesis, ArcGIS toolbox, sourcescripts

MFDtoolbox: folder with developed toolbox

MFD.tbx: ArcGIS toolbox

*.py: source scripts

dtm: folder of input sinkless DEM

MFDtest.mxd: ArcMap GIS Project

ResultingRasters: folder with resulting rasters

DiplomaRapoport2016.pdf: text part of diploma thesis