malstroem Documentation

Release 0.0.1

Asger Skovbo Petersen, Septima

May 16, 2018

Contents

Tools for screening of bluespots and water flow between bluespots

CHAPTER 1

Features

malstroem provides command line tools and a python api to calculate:

- Depressionless (filled, hydrologically conditioned) elevation models
- Surface flow directions
- Accumulated flow
- Blue spots
- Local watershed for each bluespot
- Pour points (point where water leaves blue spot when it is filled to its maximum capacity)
- Flow paths between blue spots
- Fill volume at specified rain incidents
- Spill over between blue spots at specified homogeneous rain incidents

CHAPTER 2

Assumptions

malstroem makes some assumptions to simplify calculations. The most important ones which you must be aware of:

Note:

- malstroem assumes that the terrain is an impermeable surface. This may change in a later version.
- malstroem does not know the concept of time. This means that the capacity of surface streams is infinite no matter the width or depth. Streams won´t flow over along their sides. The end result is the situation after infinite time, when all water has reached its final destination.
- Water flows from one cell to one other neighbour cell (the D8 method).
- The DEM used must cover an entire drainage basin (or more basins) in order to estimate correct stream flows from all upstream sub-watersheds within the basins.

CHAPTER 3

Example usage

Calculate all derived data including vectorization of bluespot, wateshed and streams rasters for 10mm and 30mm rain incidents ignoring bluespots where the maximum water depth is less than 5cm:

```
malstroem complete -r 10 -r 30 -filter "maxdepth > 0.5" -vector -dem dem.tif -outdir␣
↪c:\outputdirectory
```

CHAPTER 4

The project

4.1 About malstroem

4.1.1 History

The malstroem Project was initiated in November 2015 by SDFE who sponsored the development of a cloudburst screening method to be carried out by Assoc. Prof. Thomas Balstrøm, Section of Geography & Geoinformatics, Institute of Geosciences and Natural Resources, University of Copenhagen. The method was implemented as models in ModelBuilder for ArcGIS Desktop 10.4 and documented in Balstrøm (2016, submitted). In August 2016 SDFE asked Septima to implement Thomas' method in a pure Python environment based on open source technology, only, and by the end of December a first test version programmed by Asger was available. It is this version, which is presented here.

The major difference in functionality is that the ArcGIS version was based on a transfer of derived bluespots, watersheds and streams to ArcGIS' geometric network environment, where the final outputs describing the spill over in between bluespots was carried out by a custom script written in .Net and C#. In Asger's implementation the geometric network component was omitted.

4.1.2 References

Balstrøm, T. (2016, submitted): A hydrologic screening method for storm water assessments based on raster processing and geometric networks. Submitted to Computers & Geosciences, Oct. 2016.

4.1.3 Bugs and contributions

- Please report issues using the issue tracker.
- Contributions are welcome at the project home.

If you are not familiar with GitHub please read this short guide.

4.1.4 License

```
Copyright (c) 2016
Developed by Septima.dk and Thomas Balstrøm (University of Copenhagen) for the Danish␣
↪Agency for
Data Supply and Efficiency. This program is free software: you can redistribute it␣
↪and/or modify
it under the terms of the GNU General Public License as published by the Free␣
↪Software Foundation,
either version 2 of the License, or (at you option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY␣
↪WARRANTY; without
even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PORPOSE. See␣
↪the GNU Gene-
ral Public License for more details.
You should have received a copy of the GNU General Public License along with this␣
↪program. If not,
see http://www.gnu.org/licenses/.
```

CHAPTER 5

Installation

5.1 Installation

Theoretically:

```
pip install cython numpy scipy gdal
pip install git+https://github.com/Kortforsyningen/malstroem.git[speedups]
```

Unfortunately the above doesn't work on all platforms as malstroem uses som third party libraries and has some optimized code which needs to be compiled for each platform.

5.1.1 Installing on Linux

Theory is reality:

```
pip install cython numpy scipy gdal
pip install git+https://github.com/Kortforsyningen/malstroem.git[speedups]
```

5.1.2 Installing on Windows

These instructions are for Python v2.7 64bit. Change accordingly if you prefer another version of Python.

1. Download and install latest Python 2.7 "Windows x86-64 MSI installer"
2. Download and install "Microsoft Visual C++ Compiler for Python 2.7"
3. Go to Christoph Gohlke and download *numpy*, *gdal* and *scipy* wheels matching your python. For Python 2.7 64 bit it should be files ending in *cp27cp27mwin_amd64.whl*
4. Open windows command prompt and go to the scripts folder in your Python installation. In a defaut install it should be something like

```
cd c:\Python27\scripts
```

5. For each of the 3 wheel files downloaded from Gholke (starting with *numpy*) install it with pip like this:

```
pip install c:\downloads\numpy1.11.3+mklcp27cp27mwin_amd64.whl
```

6. Now (finally) install malstroem

```
pip install git+https://github.com/Kortforsyningen/malstroem.git[speedups]
```

7. Still in the scripts folder of your Python (c:python27scripts) check that malstroem responds to

```
malstroem --help
```

5.1.3 Installing on Mac OSX

The biggest problem on OSX is getting GDAL to work. One known solution is via homebrew:

1. Make sure homebrew is installed and you know how to use its Python (See for instance this guide)
2. Install GDAL and its Python bindings

```
brew install gdal
```

3. Make sure you use the homebrew Python and install malstroem and its dependencies (If you are using a virtualenv create it using *–system-site-packages*)

```
pip install cython numpy scipy
pip install git+https://github.com/Kortforsyningen/malstroem.git[speedups]
```

CHAPTER 6

Command line interface

6.1 Command Line Users Guide

malstroem's command line is a single program named `malstroem` which has a number og subcommands. Each subcommand exposes a malstroem process.

Available subcommands can be seen by invoking `malstroem --help`

```
$ malstroem --help
Usage: malstroem [OPTIONS] COMMAND [ARGS]...

  Calculate simple hydrologic models.

  To create rainfall scenarios use either the sub command 'complete' or the
  following sequence of sub command calls: filled, depths, flowdir, bspots,
  wsheds, pourpts, network, rain.

  To get help for a sub command use: malstroem subcommand --help

  Examples:
  malstroem complete -r 10 -r 30 -filter "volume > 2.5" -dem dem.tif -outdir ./outdir/
  malstroem filled -dem dem.tif -out filled.tif

Options:
  --version            Show the version and exit.
  -v, --verbosity LVL  Either CRITICAL, ERROR, WARNING, INFO or DEBUG
  --help               Show this message and exit.

Commands:
  accum     Calculate accumulated flow.
  bspots    Label bluespots.
  complete  Quick option to complete process.
  depths    Calculate bluespot depths.
  filled    Create a filled (depressionless) DEM.
```

(continues on next page)

(continued from previous page)

```
  flowdir   Calculate surface water flow directions.
  network   Calculate stream network between bluespots.
  pourpts   Determine pour points.
  rain      Calculate bluespot fill and spill volumes for...
  wsheds    Calculate bluespot watersheds.
```

Help for a given subcommand is available by invoking `malstroem subcommand --help`. For example:

```
$ malstroem accum --help
Usage: malstroem accum [OPTIONS]

  Calculate accumulated flow.

  The value in an output cell is the total number of cells upstream of that
  cell. To get the upstream area multiply with cell size.

Options:
  -flowdir PATH        Flow direction file  [required]
  -out PATH            Output file (accumulated flow)  [required]
  -v, --verbosity LVL  Either CRITICAL, ERROR, WARNING, INFO or DEBUG
  --help               Show this message and exit.
```

6.1.1 General considerations

malstroem makes the following assumptions regarding the input

- DEM horisontal and vertical units are meters.
- All subsequent processing steps assume input data as output by the former processing step of malstroem.

malstroems generally does not do very much checking that input makes sense together.

6.1.2 Vector output options

malstroem uses OGR for writing vector data. Output vector data can be tweaked using OGR specific parameters *format*, *lco*, and *dsco*.

Example writing to GeoPackage format from `malstroem pourpts`:

```
$ malstroem pourpts -bluespots bluespots.tif -depths depths.tif -watersheds wsheds.
↪tif -dem dem.tif -format gpkg -out dbfile.gpkg -layername pourpoints
```

For documentation of OGR features see the documentation of OGR formats.

6.1.3 Raster output options

malstroem uses GDAL for writing raster data. All raster data are written in GeoTiff format.

6.1.4 malstroem complete

The `complete` subcommand gives you fast-track processing from input DEM to output rain incidents including most intermediate datasets. It basically collects the subcommands `filled`, `depths`, `flowdir`, `accum`, `bspots`,

`wsheds`, `pourpts`, `network` and `rain` into one single subcommand. See these subcommands to learn more about what happens or see *Complete chain of processes*.

Arguments:

- `dem` is a raster digital elevation model. Both horisontal and vertical units must be meters.
- `r` or `rain` One or more rain incidents to calculate. In mm. `-r value` can be specified multiple times.
- If `accum` is specified the accumulated flow is calculated. This takes some time and is not strictly required.
- If `vector` is specified the bluespots and watersheds are vectorized. This takes some time and is not required.
- `filter` allows ignoring bluespots based on their area, maximum depth and volume. Format: `area > 20.5 and (maxdepth > 0.05 or volume > 2.5)`. Bluespots that do not pass the filter are ignored in all subsequent calculations. For instance their capacity is not taken into account.
- `outdir` is the path to the output directory where all output files are written. This directory must exist and be empty.

Example:

```
$ malstroem complete -r 10 -r 30 -filter "volume > 2.5" -dem dem.tif -outdir ./outdir/
```

6.1.5 malstroem filled

The `filled` subcommand creates a filled (depressionless) DEM.

In a depressionless terrain model each cell will have at least one non-uphill path to the raster edge. This means that a depressionless terrain model will have flat areas where it has been filled.

Arguments:

- `dem` is a raster digital elevation model. Both horisontal and vertical units must be meters.

Outputs:

- The filled DEM to a new raster

Example:

```
$ malstroem filled -dem dem.tif -out filled.tif
```

6.1.6 malstroem depths

The `depths` subcommand calculates bluespot depths.

Depths are calculated by subtracting the original DEM from the filled DEM

Arguments:

- `dem` is the raster digital elevation model.
- `filled` is the filled version of the input DEM.

Outputs:

- A new raster with the bluespot depth in each cell. Cells not in a bluespot will have the value 0.

Example:

```
$ malstroem depths -dem dem.tif -filled filled.tif -out depths.tif
```

6.1.7 malstroem flowdir

The `flowdir` subcommand calculates surface water flow directions.

This is a two step process:

Step 1: Fill depressions in the DEM in a way which preserves a downward slope along the flow path. This is done by requiring a (very) small minimum slope between cells. This results in flow over filled areas being routed to the nearest pour point.

Step 2: Flow directions for each cell. Uses a D8 flow routing algorithm: At each cell the slope to each of the 8 neighboring cells is calculated. The flow is routed to the cell which has the steepest slope. If multiple cells share the same maximum slope the algorithm picks one of these cells.

Flow direction from a cell is encoded: *Up=0*, *UpRight=1*, . . . , *UpLeft=7*, *NoDirection=8*

Arguments:

- `dem` is the raster digital elevation model.

Outputs:

- A new raster where the flow direction from each cell is encoded.

Example:

```
$ malstroem depths -dem dem.tif -out flowdir.tif
```

6.1.8 malstroem accum

The subcommand `accum` calculates accumulated flow.

The value in an output cell is the total number of cells upstream of that cell.

Arguments:

- `flowdir` is the flow direction raster.

Outputs:

- A raster where the value in each cell is the number of cells upstream of that cell.

Example:

```
$ malstroem accum -flowdir flowdir.tif -out out.tif
```

6.1.9 malstroem bspots

The `bspots` subcommand identifies and labels all cells belonging to each bluespot with a unique bluespot ID.

Note:

- The unique ID is an integer in the range from 1 to the number of bluespots in the dataset. So bluespot IDs are NOT unique across different datasets.
- IDs are not necessarily assigned the same way between different runs on the same dataset.

- The ID 0 (zero) is used for cells which do not belong to a bluespot.

Bluespots with certain properties can be ignored by specifying a filter expression. Available properties are

`maxdepth` which is the maximum depth of the bluespot. `area` which is the area of the bluespot in m2. `volume` which is the bluespot volume (or water capacity) in m3.

Allowed operators are <, >, ==, !=, >=, <=, `and` and `or`. Parenthises can be used to make the expression more readable.

An example of a valid *filter*:

```
maxdepth > 0.05 and (area > 20 or volume > 0.5)
```

Note:

- Bluespots that do not pass the filter are ignored in all subsequent calculations. For instance their capacity is not taken into account.

Arguments:

- `depths` is a raster with bluespot depths
- `filter` allows ignoring bluespots based on their area, maximum depth and volume. Format: `area > 20.5 and (maxdepth > 0.05 or volume > 2.5)`. Bluespots that do not pass the filter are ignored in all subsequent calculations. For instance their capacity is not taken into account.‘

Outputs:

- A raster with bluespot IDs. The ID 0 (zero) is used for cells which do not belong to a bluespot.

Example:

```
$ malstroem bspots -depths depths.tif -filter "maxdepth > 0.05 and (area > 20 or␣
↪volume > 0.5)" -out bluespots.tif
```

6.1.10 malstroem wsheds

The subcommand `wsheds` determines the local watershed of each bluespot.

All cells in the local watershed is assigned the bluespot ID.

Arguments:

- `bluespots` is the bluespot ID raster.
- `flowdir` is the flow direction raster.

Outputs:

- A raster with bluespot watersheds identified by bluespot ID.

Example:

```
$ malstroem wshed -bluespots bluespots.tif -flowdir flowdir.tif -out wsheds.tif
```

6.1.11 malstroem pourpts

The `pourpts` subcommand determines a pour point for each bluespot.

A pour point is the point where water leaves the blue spot when it is filled to its maximum capacity.

Pour point are determined using one of two methods:

- Random candidate. Requires DEM only
- Maximum accumulated flow candidate. Requires accumulated flow

The output of the two methods only differ when there are more than one pour point candidate (ie multiple threshold cells with identical Z for a given bluespot).

Arguments:

- `bluespots` is the bluespot ID raster.
- `depths` is a raster with bluespot depths.
- `watersheds` is a raster with local bluespot watershed identified by bluespot IDs.
- `dem` the DEM. Only required if `accum` is not used.
- `accum` accumulated flow raster. Required if `dem` is not used.
- `out` output OGR datasource.
- `layername` name of output vector layer within the output datasource.

Outputs:

- Vector Point layer with pour points.

Table 1: **Pour point attributes**

Attribute Name	Description
bspot_id	Bluespot ID
bspot_area	Bluespot area in m2
bspot_vol	Bluespot volume (or capacity) in m3
bspot_dmax	Maximum depth of the bluespot
bspot_fumm	Rain needed to fill up this bluespot with water from local watershed only. In mm.
wshed_area	Area of local bluespot watershed. In m2.
cell_row	Raster row index of pour point location
cell_col	Raster column index of pour point location

Example:

```
$ malstroem pourpts -bluespots bluespots.tif -depths depths.tif -watersheds wsheds.
↪tif -dem dem.tif -out shpdir/ -layername pourpoints
```

6.1.12 malstroem network

The subcommand `network` calculates the stream network between bluespots.

Streams are traced from the pour point of each bluespot using the flow directions raster.

A stream ends:

- when it first enters the next downstream bluespot.

- when it merges with another stream

When two or more streams merge a new node of type `junction` is inserted and a new stream is traced downstream from the node.

Streams stop at the border of the bluespot because routing within the bluespot will otherwise happen on a synthetic surface sloping towards the pour point. This has nothing to do with the real flow of the water.

Arguments:

- `bluespots` bluespots ID raster.
- `flowdir` flow direction raster.
- `pourpoints` OGR vector datasource with pour points.
- `pourpoints_layer` layer name within *pourpoints* datasource. Needed when datasource can have multiple layers (eg. a database).
- `out` output OGR datasource.
- `out_nodes_layer` layer name for output *nodes* layer within the output datasource.
- `out_streams_layer` layer name for output *streams* layer within the output datasource

Outputs:

- Nodes vector Point layer establishing a network
- Streams vector LineString layer

Table 2: **Nodes attributes**

Attribute Name	Description
nodeid	Integer ID for each node.
nodetype	`pourpoint` or `junction`.
dstrnodeid	`nodeid` of the next downstream node.
bspot_id	Bluespot ID. NULL for nodes of type `junction`.
bspot_area	Bluespot area in m2. 0 (zero) for nodes of type `junction`.
bspot_vol	Bluespot volume (or capacity) in m3. 0 (zero) for nodes of type `junction`.
wshed_area	Area of local bluespot watershed. In m2. 0 (zero) for nodes of type `junction`.
cell_row	Raster row index of pour point location
cell_col	Raster column index of pour point location

Table 3: **Streams attributes**

Attribute Name	Description
nodeid	Integer ID for starting node of the stream.
dstrnodeid	`nodeid` of the next downstream node.

Note:

- As streams end at the border of the downstream bluespot they do not form a complete geometric network.
- The network can be established by using the `nodeid` and `dstrnodeid` attributes.

Example:

```
$ malstroem network -bluespots bluespots.tif -flowdir flowdir.tif -pourpoints shpdir/
↪pourpoints.shp -out shpdir/ -out_nodes_layer nodes -out_streams_layer streams
```

6.1.13 malstroem rain

The subcommand `rain` calculates bluespot fill and spill volumes for specific rain events.

For each rain event bluespot fill and spill volumes are calculated for all nodes and spill is propagated downstream.

Arguments:

- `nodes` OGR datasource containing nodes layer.
- `nodes_layer` layer name within *nodes* datasource. Needed when datasource can have multiple layers (eg. a database).
- `r` or `rain` is a rain incident in mm. Note that multiple rain incidents can be calculated at once by repeating the '-r' option.
- `out` output OGR datasource.
- `out_layer` layer name for output layer within the output datasource.

Outputs:

- Events Point layer where fill and spill has been calculated for all nodes

Table 4: **Events attributes**

Attribute Name	Description
nodeid	Integer ID for each node.
nodetype	`pourpoint` or `junction`.
dstrnodeid	`nodeid` of the next downstream node.
bspot_id	Bluespot ID. NULL for nodes of type `junction`.
bspot_area	Bluespot area in m2. 0 (zero) for nodes of type `junction`.
bspot_vol	Bluespot volume (or capacity) in m3. 0 (zero) for nodes of type `junction`.
wshed_area	Area of local bluespot watershed. In m2. 0 (zero) for nodes of type `junction`.
cell_row	Raster row index of pour point location
cell_col	Raster column index of pour point location

Table 5: **Events attributes repeated for each rain event of xx mm**

Attribute Name	Description
rainv_xx	Volume of rain falling on the local watershed. In m3.
v_xx	Volume of water in the bluespot. (Sum of water falling on local watershed and water flowing in from upstream). In m3.
pctv_xx	Percentage of bluespot volume (capacity) filled.
spillv_xx	Volume of water spilled downstream from the bluespot. In m3.

Example:

```
$ malstroem rain -nodes shpdir/ -nodes_layer nodes -r 10 -r 20 -out shpdir/ -out_
↪layer nodes
```

6.1.14 Complete chain of processes

The complete process from DEM to fill and spill volumes for a rain event can be calculated with the `malstroem complete` subcommand (see *malstroem complete*). If you need greater control than offered by this command, you need to do the processing in steps using the other subcommands.

The below series of process calls will produce the same results as `malstroem complete`:

```
$ malstroem filled -dem dem.tif -out filled.tif
$ malstroem depths -dem dem.tif -filled filled.tif -out depths.tif
$ malstroem flowdir -dem dem.tif -out flowdir.tif
$ malstroem accum -flowdir flowdir.tif -out accum.tif
$ malstroem bspots -filter "maxdepth > 0.05 and (area > 20 or volume > 0.5)" -depths␣
↪depths.tif -out bspots.tif
$ malstroem wsheds -bluespots bspots.tif -flowdir flowdir.tif -out wsheds.tif
$ malstroem pourpts -bluespots bspots.tif -depths depths.tif -watersheds wsheds.tif -
↪dem dem.tif -out shpdir/
$ malstroem network -bluespots bspots.tif -flowdir flowdir.tif -pourpoints shpdir/ -
↪out shpdir
$ malstroem rain -nodes shpdir/ -r 10 -r 20 -out shpdir/
```

This workflow utilizes default OGR output format and layer names. Both formats and layer names can be controlled by parameters.

CHAPTER 7

API documentation

7.1 malstroem package

7.1.1 Subpackages

malstroem.algorithms package

Subpackages

malstroem.algorithms.speedups package

Module contents

Submodules

malstroem.algorithms.dtypes module

malstroem.algorithms.fill module

malstroem.algorithms.flow module

malstroem.algorithms.label module

malstroem.algorithms.net module

Module contents

malstroem.scripts package

Submodules

malstroem.scripts.bluespot module

malstroem.scripts.cli module

CHAPTER 8

Indices and tables

- genindex
- modindex
- search