# User guide to chi analysis

Simon M. Mudd, School of GeoSciences, University of Edinburgh

# Table of Contents

# Preface by Simon Mudd

This is the documentation for the chi analysis tool. It is the second generation of the software reported in Mudd et al. 2014. Right now there is not very much documentation but that will change in the near future.

# Chapter 1. Introduction

This section explains the theory behind analysis of channels using both slope-area analysis and chi analysis. If you just want to run the software skip ahead to the section Getting the software.

## 1.1. Background

In the late 1800s, G.K. Gilbert proposed that bedrock channel incision should be proportional to topographic gradients and the amount of water flowing in a channel.

> We have already seen that erosion is favored by declivity. Where the declivity is great the agents of erosion are powerful; where it is small they are weak; where there is no declivity they are powerless. Moreover it has been shown that their power increases with the declivity in more than simple ratio.
>
> — G.K. Gilbert, Geology of the Henry Mountains 1877

Since then, many geomorpholgists have attempted to extract information about erosion rates from channel profiles. Chi analysis is a method of extracting information from channel profiles that attempts to compare channels with different discharges first proposed by Leigh Royden and colleagues at MIT. LSDTopoTools has a number of tools for performing chi analysis.

This document gives instructions on how to use the segment fitting tool for channel profile analysis developed by the Land Surface Dynamics group at the University of Edinburgh. The tool is used to examine the geometry of channels using the integral method of channel profile analysis. For background to the method, and a description of the algorithms, we refer the reader to Mudd et al. (2014). For background into the strengths of the integral method of channel profile analysis, the user should read Perron and Royden (2013, ESPL).

## 1.2. Background information

These formulations can be generalised into the stream power incision model:

where $E$ is the long-term fluvial incision rate, $A$ is the upstream drainage area, $S$ is the channel gradient, $K$ is the erodibility coefficient, which is a measure of the efficiency of the incision process, and $m$ and $n$ are constant exponents.  In order to model landscape evolution, equation () is often combined with detachment-limited mass balance, where:

where $z$ is the elevation of the channel bed, $t$ is time, $x_d$ is the distance downstream, and $U$ is the rock uplift rate, equivalent to the rate of baselevel lowering if the baselevel elevation is fixed \citep{howard_channel_1983,howard_detachment-limited_1994,whipple_dynamics_1999}.

In order to examine fluvial response to climatic and tectonic forcing, equation () is often rearranged for channel slope, assuming uniform incision rate \citep{hack_stream-profile_1973,flint_stream_1974,sklar_river_1998,whipple_dynamics_1999,wobus_tectonics_2006}:

$S = k_{sn} A^{-\theta}$

where $\theta = m/n$, and represents the concavity of the channel profile, and $k_{sn} = (E/K)^{\frac{1}{n}}$, and represents the steepness of the profile. $\theta$ and $k_{sn}$ are referred to as the concavity and steepness indices respectively. Equation () therefore predicts a power-law relationship between slope and drainage area, which is often represented on a logarithmic scale (Figure BLAH). The concavity and steepness indices can be extracted from plots of slope against drainage area along a channel, where $\theta$ is the gradient of a best-fit line through the data, and $k_{sn}$ is the y-intercept. These slope-area plots have been used by many studies to examine fluvial response to climate, lithology and tectonics \citep[e.g.][]{flint_stream_1974,tarboton_scaling_1989,kirby_quantifying_2001,wobus_tectonics_2006}.

However, there are limitations with using these plots of slope against drainage area in order to analyse channel profiles. Topographic data is inherently noisy, either as a result of fine-scale sediment transport processes, or from processing of the data in the creation of DEMs. Furthermore, this noise is amplified by the derivation of the topographic surface in order to extract values for channel gradient \citep{perron_integral_2013}. This noise leads to significant scatter within the profile trends, potentially obscuring any deviations from the power law signal which may represent changes in process, lithology, climate or uplift. In order to circumvent these problems, more recent studies have turned to the `integral method' of slope-area analysis, which normalises river profiles for their drainage area, allowing comparison of the steepness of channels across basins of different size \citep{royden_evolution_2000,harkins_transient_2007,perron_integral_2013,mudd_statistical_2014}.
The integral method only requires the extraction of elevation and drainage area along the channel, and is therefore less subject to topographic noise than slope-area analysis. The technique involves integrating equation ~\ref{eq:stream_power_slope}, assuming spatially constant incision equal to uplift (steady-state) and erodibility:

$z(x) = z(x_b) + \Bigg(\frac{E}{K}\Bigg)^{\frac{1}{n}} \int_{x_b}^{x} \frac{dx}{A(x)^{\frac{m}{n}}}$

where the integration is performed upstream from baselevel ($x_b$) to a chosen point on the river channel, $x$. The profile is then normalised to a reference drainage area ($A_0$) to ensure the integrand is dimensionless:

z(x) = z(x_b) + \Bigg(\frac{U}{K{A_0}^m}\Bigg)^{\frac{1}{n}} \chi

where the longitudinal coordinate $\chi$ is equal to:

$\chi = \int_{x_b}^{x} \Bigg(\frac{A_0}{A(x)}\Bigg)^{m/n} dx$

The longitudinal coordinate $\chi$ has dimensions of length, and is linearly related to the elevation $z(x)$. Therefore, if a channel incises based on the stream power incision model, then its profile should be linear on a plot of elevation against $\chi$ according to equation (). As well as providing a method to test whether channel profiles obey common incision models, $\chi$-plots also provide means of testing the appropriate $\theta$ for a channel

\citep{perron_integral_2013,mudd_statistical_2014}. If the integral analysis is performed for all channels within a basin, the correct value of $\theta$ can be determined by identifying at which value all of the channels are both linear in $\chi$-elevation space, and collinear, where main channel and tributaries all collapse onto a single profile (Figure \ref{fig:chi_plot}).

# Chapter 2. Getting the software

We have attempted, to the best of our ability, to streamline the process of installing our software on any platform (Windows, Linux, MacOS). You will need to install a bit of software. If you have a decent internect connections this should take under an hour. The main rate limiting step is downloading files: You will need to download ~2Gb of software. This requirement is substantially reduced if you want to install on a a native Linux machine.

## 2.1. Starting up on your own Linux machine

If you work on a Linux operating system we have a python script for installing everything. You will need to be connected to the internet, and have python installed. You can skip ahead to the section Installing the code and setting up directories.

## 2.2. Starting up on a non-Linux operating system (Windows, MacOS)

Below are quick instructions. If you have trouble, follow the link above and then go through the complete instructions.

**Quick Instructions for using Vagrant for LSDTopoTools**

1. Download and install virtualbox.
2. Download and install vagrant. You might have to restart your computer after this.
3. If you are on Windows, download putty.exe. If you are on Linux or MacOS you can skip this (they have built-in equivalents).
4. Make a directory for your vagrant box. We tend to put ours in a directory called `VagrantBoxes`.
5. Inside that directory make two new directories: `LSDTopoTools` and `Ubuntu`. The second directory's name doesn't matter, it is just for holding a vagrant file (see below). However you **MUST** have the `LSDTopoTools` directory. The directory name is **case sensitive**!
6. Download one of our vagrantfiles: https://github.com/LSDtopotools/LSDTT_vagrantfiles into the `Ubuntu` directory (again, the name of the directory doesn't matter).
7. Rename the vagrantfile from the repo (either `Vagrantfile_32bit_FFTW` or `Vagrantfile_64bit_FFTW`) simply `vagrantfile`. Your operating system is almost certainly 64 bit, but on most computers you need to select 32 bit because the default setting is to disable 64 bit guest operating systems. This can be changed but only by expert users.
8. Open a terminal or powershell window and navigate to the directory with the vagrantfile.
9. Run `vagrant up` from the command line.

   **WARNING** If you are running `vagrant up` for the first time it can take some time to download the base box. They are several hundred Mb each!

10. Run `vagrant provision` after the box has started.
11. If on Windows, you should now be able to use `putty.exe` to ssh into your LSDTopoTools server. The host name is almost always `127.0.0.1` and the port is almost always `2222`.
12. On Windows, you will need to give a username and password after connecting using `putty.exe`. The machine is running locally on your computer so nothing is being sent over the internet. The username is always `vagrant` and the password is also `vagrant`.
13. If you are on MacOS or Linux you do not need `putty.exe`; all you need to do is type `vagrant ssh` into the command line. See the vagrant instructions.

## 2.3. Troubleshooting a vagrant server

There are a few common problems people have when running a vagrant server.

- If you are on an old computer, sometimes vagrant times out before the virtual machine boots. This most frequently happens the first time you boot a vagrant machine. The most effective way to fix this is with the canonical IT solution: turning it off and on again. To do that run `vagrant halt` and `vagrant up` in succession.
- If vagrant hangs up in the powershell or terminal window and does not give you back the command prompt, turn it off and on again by typing ctrl-c and then running the vagrant command again.
- If your files are not synching across your host and vagrant machine, it is probably because there is some misspelling in your `LSDTopoTools` folder on the host machine. Make sure that folder is in the correct place and is spelled correctly (remember it should be case sensitive!!).

## 2.4. Installing the code and setting up directories

### Easy setup quick start

If you just want to get started with the standard setup without any details, do the following:

1. Go into a Linux terminal. If you are on your own computer and it is not Linux, you will need to start a Vagrant session. You will need python but this is now standard on most Linux distributions so we will assume you have it.
2. If you used our vagrantfile the setup tool will already be on your vagrant box. It is sitting in the LSDTopoTools directory. You can go there with `cd /LSDTopoTools`.
3. If you are on your own linux machine you can get the setuptool with:

```
$ wget
https://raw.githubusercontent.com/LSDtopotools/LSDAutomation/master/LSDTopoToolsSetup.py
```

4. Run the setup tool:

```
$ python LSDTopoToolsSetup.py -id 0 -MChi True
```

5. You are ready to go!!

**NOTE** **For native Linux users**: The `-id` flag tells LSDTopoToolsSetup.py where to put the files. In vagrant it always starts in the root directory (this is for file syncing purposes). However, if your native operating system is Linux, then you can either install in your home directory (`-id 0`) or in the directory where you called LSDTopoToolsSetup.py (`-id 1`).

1. The first thing you need to do is get our python program **LSDTopoToolsSetUp.py**. It lives here: https://github.com/LSDtopotools/LSDAutomation/blob/master/LSDTopoToolsSetup.py

   You will need to run this file in your Linux environment, so in a terminal window type:

   ```
   $ wget
   https://raw.githubusercontent.com/LSDtopotools/LSDAutomation/master/LSDTopoToolsSetup.p
   y
   ```

   > **NOTE** If `wget` doesn't work, you can follow the link: https://raw.githubusercontent.com/LSDtopotools/LSDAutomation/master/LSDTopoToolsSetup.py
   > Copy the text, paste it into a text editor and save it as `LSDTopoToolsSetup.py`.

2. Now, in your terminal window run this script. It has some options:

   a. For the most basic setup, type:

   ```
   $ python LSDTopoToolsSetup.py -id 0  -MChi True
   ```

   In our Vagrant setup, this will install everything in the root directory (you can got there with `$ cd /`), which is the default setup generated by our Vagrantfile. If you are not in vagrant the **LSDTopoTools** directories will be in your home directory (you can get there with `$ cd ~`).

   b. If you don't want **LSDTopoTools** in your home directory, you can install it in your current directory with:

   ```
   $ python LSDTopoToolsSetup.py -id 1 -MChi True
   ```

3. It turns out the `LSDTopoToolsSetup.py` tool has a number of options but we will explain these later. If you want a preview of what it does, you can call its help options:

   ```
   $ python LSDTopoToolsSetup.py -h
   ```

# Chapter 3. Preparing your data.

Use GDAL. And also DEM_preprocessing. They are both great.

# Chapter 4. Running the chi analysis

The chi analysis, and a host of other analyses, are run using the **chi_mapping_tool.exe** program. If you followed the instructions in the section Getting the software you will already have this program compiled.

The program **chi_mapping_tool.exe** runs with two inputs when you call the program:

1. The path to the parameter file. This **MUST** inclde the trailing slash (i.e., `/LSDTopoTools/Topographic_projects/Test_data` is incorrect whereas `/LSDTopoTools/Topographic_projects/Test_data/` is correct).

2. The name of the parameter file.

So if the parameter file is located at `/LSDTopoTools/Topographic_projects/Test_data` and it is called `test_chi_map.param`, you run the program with:

```
$ ./chi_mapping_tool.exe /LSDTopoTools/Topographic_projects/Test_data test_chi_map.param
```

As we will see momentarily, the data an the parameter file can be in different locations, although in general it might be sensible to place the parameter file in the sample directory as the data.

The code is run using a parameter file, within which users can set the data they want to print to file. Regardless of which data they choose to print to file, a file will be printed with the extension `_Input.param` which prints out all the parameters used in the analysis (including the default parameters). This file can be used to verify if the correct parameters have been used in the analysis.

## 4.1. The parameter file

The parameter file has keywords followed by a value. The format of this file is similar to the files used in the `LSDTT_analysis_from_paramfile` program, which you can read about in the section [Running your first analysis].

**NOTE** The parameter file has a specific format, but the filename can be anything you want. We tend to use the extensions `.param` and `.driver` for these files, but you could use the extension `.MyDogSpot` if that tickled your fancy.

The parameter file has keywords followed by the `:` character. After that there is a space and the value.

### Chi mapping parameter file format

1. Lines beginning with `#` are comments.
2. Keywords or phrases are followed by a colon (`:`).
3. The order of the keywords do not matter.
4. Keywords are not sensitive, but must match expected keywords.
5. If a keyword is not found, a default value is assigned.

## 4.2. Parameter file options

Below are options for the parameter files. Note that all DEMs must be in ENVI `bil` format (**DO NOT** use ArcMap's bil format: these are two different things. See the section [What data does LSDTopoToolbox take?] if you want more details). The reason we use `bil` format is because it retains georeferencing which is essential to our file output since many of the files output to `csv` format with latitude and longitude as coordinates.

*Table 1. File input and output options.* **These do not have defaults and MUST be declared**.

| Keyword | Input type | Description |
|---|---|---|
| write path | string | The path to which data is written. **The code will NOT create a path: you need to make the write path before you start running the program.** |
| read path | string | The path from which data is read. |
| write fname | string | The prefix of rasters to be written **without extension**.<br>For example if this is `Test` and you have selected `bil` format then a fill operation will result in a file called `Test_Fill.bil`. |
| read fname | string | The filename of the raster to be read without extension. For example if the raster is `MyRaster.bil`, read fname will be `MyRaster`. |
| channel heads fname | string | The filename of a channel heads file. You can import channel heads. If this is set to `NULL` then the channels will be calculated using a pixel threshold. |

*Table 2. Options for determining which channels and basins to analyse, including settings for the fill function.*

| Keyword | Input type | Default value | Description |
|---|---|---|---|
| min_slope_for_fill | float | 0.001 | The minimum slope between pixels for use in the fill function. |
| threshold_contributing_pixels | int | 1000 | The number of pixes required to generate a channel (i.e., the source threshold). |
| minimum_basin_size_pixels | int | 1000 | The minimum number of pixels in a basin for it to be retained. This operation works on the baselevel basins: subbasins within a large basin are retained. |
| test_drainage_boundaries | bool (true or 1 will work) | false | A boolean that, if set to true, will eliminate basins with pixels drainage from the edge. This is to get rid of basins that may be truncated in a DEM (and thus will have incorrect chi values). |
| only_take_largest_basin | bool (true or 1 will work) | true | If this is true, a chi map is created based only upon the largest basin in the raster. |

*Table 3. Parameters for calculating the chi coordinate.*

| Keyword | Input type | Default value | Description |
|---|---|---|---|
| A_0 | float | 1000 | The $\mathbf{A_0}$ parameter (which nondimensionalises area) for chi analysis. This is in $m^2$. |
| m_over_n | float | 0.5 | The **m/n** paramater (sometimes known as the concavity index) for calculating chi. |
| threshold_pixels_for_chi | int | 1000 | The number of contributing pixels above which chi will be calculated. The reason for the threshold is to produce chi plots that do not extend onto hillslopes; this helps visualisation of chi differences between nearby headwater channels. |

*Table 4. Parameters for calculating the segments of similar chi slope (M_{\chi}). More details on the use fo these parameters can be found in [Mudd et al., JGR-ES 2014](#).*

| Keyword | Input type | Default value | Description |
|---|---|---|---|
| n_iterations | int | 20 | The number of iterterations of random sampling of the data to construct segments. The sampling probability of individual nodes is determined by the skip parameter. |

| Keyword | Input type | Default value | Description |
| --- | --- | --- | --- |
| target_nodes | int | 80 | The number of nodes in a segment finding routine. Channels are broken into subdomains of aroung this length and then segmenting occurs on these subdomains. |
| minimum_segment_length | int | 10 | The minimum length of a segment in sampled data nodes. The actual length is approxamately this parameter times (1+skip). |
| skip | int | 2 | During Monte Carlo sampling of the channel network, nodes are sampled by skipping nodes after a sampled node. The skip value is the mean number of skipped nodes after each sampled node. For example, if skip = 1, on average every other node will be sampled. Skip of 0 means every node is sampled (in which case the n_iterations should be set to 1, because there will be no variation in the fit between iterations). |
| sigma | float | 10.0 | This represents the variability in elevation data (if the DEM has elevation in metres, then this parameter will be in metres). It should include both uncertainty in the elevation data as well as the geomorphic variability: the size of roughness elements, steps, boulders etc in the channel that may lead to a channel profile diverging from a smooth long profile. |
| basic_Mchi_regression_nodes | int | 11 | This works with the basic chi map: segments are not calculated. Instead, a moving window, with a length set by this parameter, is moved over the channel nodes to calculate the chi slope. This method is very similar to methods used to calculate normalised channel steepness ($\mathbf{k_{sn}}$). |

*Table 5. Keywords for setting which analyses to be preformed and which files to print.* **These are all booleans!** ***Defaults are all false so these parameters must be set to true to perform analyses and print to file.*

| Input type | Description |
| --- | --- |
| only_check_parameters | If this is true, the program simply prints all the parameters to a file and does not perform any analyses. This is used for checking if the parameters are set correctly and that the keywords are correct. |
| print_stream_order_raster | If true, prints a raster of the stream orders. |
| print_junction_index_raster | If true, prints a raster with the junction indices. |
| print_fill_raster | If true, prints a filled raster |
| print DrainageArea_raster | If true, prints a raster of the draiange area in $m^2$. |
| print_chi_coordinate_raster | If true, prints a raster with the chi coordinate (in m). Note that if you want to control the size of the data symbols in a visualisation, you should select the `print_simple_chi_map_to_csv` option. |
| print_simple_chi_map_to_csv | If true, prints a csv file with latitude, longitude and the chi coordinate. Can be converted to a shapefile or GeoJSON with our python mapping scripts. This options gives more flexibility in visualisation than the raster, since in the raster data points will only render as one pixel. |
| print_simple_chi_map_to_csv | If true, prints a csv file with latitude, longitude and the chi coordinate. Can be converted to a shapefile or GeoJSON with our python mapping scripts. This options gives more flexibility in visualisation than the raster, since in the raster data points will only render as one pixel. |
| print_segmented_M_chi_map_to_csv | If true, prints a csv file with latitude, longitude and a host of chi information including the chi slope, chi intercept, drainage area, chi coordinate and other features of the drainage network. The M_{\chi} values are calculated with the segmentation algorithm of Mudd et al. 2014. |
| print_basic_M_chi_map_to_csv | If true, prints a csv file with latitude, longitude and a host of chi information including the chi slope, chi intercept, drainage area, chi coordinate and other features of the drainage network. The M_{\chi} values are calculated with a rudimentary smoothing window that has a size determined by the parameter `basic_Mchi_regression_nodes`. |

### 4.2.1. Example parameter file

Below is an exaple parameter file. This file is included in the repository along with the driver functions.

```
# Parameters for performing chi analysis
# Comments are preceeded by the hash symbol
# Documentation can be found here:
#
http://lsdtopotools.github.io/LSDTT_book/#_chi_analysis_part_3_getting_chi_gradients_for_
the_entire_landscape

# These are parameters for the file i/o
# IMPORTANT: You MUST make the write directory: the code will not work if it doens't
exist.
read path: /LSDTopoTools/Topographic_projects/Test_data
write path: /LSDTopoTools/Topographic_projects/Test_data
read fname: Mandakini
channel heads fname: NULL

# Parameter for filling the DEM
min_slope_for_fill: 0.0001

# Parameters for selecting channels and basins

threshold_contributing_pixels: 200000
minimum_basin_size_pixels: 50000
test_drainage_boundaries: false

# Parameters for chi analysis
A_0: 1000
m_over_n: 0.45
threshold_pixels_for_chi: 20000

n_iterations: 20
target_nodes: 80
minimum_segment_length: 10
sigma: 10.0
skip: 2

# The data that you want printed to file
only_check_parameters: true
print_stream_order_raster: false
print_DrainageArea_raster: false
print_segmented_M_chi_map_to_csv: true
```

### 4.2.2. Output data formats

Data is written to either rasters or csv files. The rasters are all in `bil` format, which you can read about in the section: [What data does LSDTopoToolbox take?]

The **csv** files are comma seperated value files which can be read by spreadsheets and GIS software. These files all have labeled columns so their contents can be easily views. All of the files contain **latitude** and **longitude** columns. These columns are projected into the **WGS84** coordinate system for ease of plotting in GIS software.

### Viewing data and converting to GIS ready formats

If the user has opted to print data in `csv` format, they can use our visualisation tools to convert the data into GIS-ready formats.

Users should first clone the mapping tools respoitory:

```
$ git clone https://github.com/LSDtopotools/LSDMappingTools.git
```

In this repository the user needs to get a helping script called `LSDOSystemTools.py`. You can fetch this script using the `wget` tool:

```
$ wget https://github.com/LSDtopotools/LSDAutomation/raw/master/LSDOSystemTools.py
```

The user can then run the script `TestMappingToolsPoint.py`, activating the `TestMappingToolsLassoCSV` function:

```
if __name__ == "__main__":
    TestMappingToolsLassoCSV()
```

and changing the target directory to the directory storing the csv files:

```
def TestMappingToolsLassoCSV():
    DataDirectory = "C://VagrantBoxes//LSDTopoTools//Topographic_projects//Test_data//"
    LSDP.ConvertAllCSVToGeoJSON(DataDirectory)
```

Note that this is if your run the python script within windows. If you run it within your agrant Linux machine the directory would be:

```
def TestMappingToolsLassoCSV():
    DataDirectory = "/LSDTopoTools/Topographic_projects/Test_data/"
    LSDP.ConvertAllCSVToGeoJSON(DataDirectory)
```

You can convert all `csv` files into either shapefiles or GeoJSON files.

```
LSDP.ConvertAllCSVToGeoJSON(DataDirectory)
LSDP.ConvertAllCSVToShapefile(DataDirectory)
```

These files can then be read by your favourite GIS.

# Chapter 5. Visualisation

We use python, since it is much better than the alternatives. Don’t ever use Matlab.