



THE UNIVERSITY *of* EDINBURGH

Using the channel profile analysis tool, built by the University of Edinburgh Land Surface Dynamics group

Part A: Extracting the channel network from a DEM

Simon M. Mudd, Mikaël Attal, David T. Milodowski, Stuart W.D. Grieve and Declan A. Valters

School of GeoSciences, University of Edinburgh

Contact: simon.m.mudd_at_ed.ac.uk

Table of Contents

1. Quick guide	1
2. Overview	2
3. Warning.....	2
4. Getting the raw DEM into the toolkit	2
a. Data Sources	2
b. Converting your data into something LSDTopoTools can understand (.flt format)	3
5. Compiling the two driver functions	3
6. Running the channel network extraction	4
a. Writing junctions.....	4
b. Extracting the .chan file	8
c. Format of the .chan file.....	9
Appendix: Installing a compiler.....	10
Installing a compiler on a windows machine	10

1. Quick guide

If you already know more or less what you are doing, but need a quick reminder, here are the steps involved:

- a. Download your DEM.
- b. Project it into a projected coordinate system (we usually use UTM).
- c. Export the DEM in .flt format.
- d. If the programs aren't compiled, make them with:
`chi_step1_write_junctions.make`
`chi_step2_write_junctions.make`
- e. Run the program `chi1_write_junctions.exe` on your DEM.
- f. Import the junction raster (*.Jl.flt) into a GIS and pick a junction (this is easiest if you also import the stream order (*.SO.flt) and hillshade (*.HS.flt).
- g. Run `chi2_write_channel_file.exe` to get the .chan file. Once you do this you are ready to move on to section two: running the chi analysis!

2. Overview

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. (2013, draft manuscript). For background into the strengths of the integral method of channel profile analysis, the user should read Perron and Royden (2013, ESPL):

<http://mit.edu/perron/www/files/PerronRoyden13.pdf>

This document guides the user through the installation process, and explains how to use the model. You will need a c++ compiler for this tutorial. If you have no idea what a c++ compiler is, see the appendix. Visualisation of the model results is performed using Python scripts. We recommend installing Python(x,y) (<https://code.google.com/p/pythonxy/>) and running the scripts within Spyder (which is installed with Python(x,y)). Both the recommended compiler and Python(x,y) are open source: you do not need to buy any 3rd party software (e.g., Matlab) to run our topographic analysis!

3. Warning

This code is for research purposes and is under continuous development, so we cannot guarantee a bug-free experience!

4. Getting the raw DEM into the toolkit

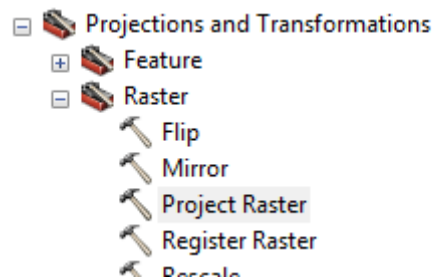
a. Data Sources

- i. The topographic analysis package works from a DEM. You can get these from all sorts of places. You can get data from a number of sources. For LiDAR, two good sources are opentopography (<http://www.opentopography.org/>) and the U.S. Interagency Elevation Inventory (<http://www.csc.noaa.gov/inventory/#>). Other countries are not so progressive about releasing data but you could rummage around the links here: http://en.wikipedia.org/wiki/National_lidar_dataset. For 10m data of the United States you can go to the national map viewer

(<http://viewer.nationalmap.gov/viewer/>). I get ASTER 30m data from the NASA's reverb site (<http://reverb.echo.nasa.gov>), One site that has filled and corrected 90-m DEMs is here: <http://www.viewfinderpanoramas.org/dem3.html>

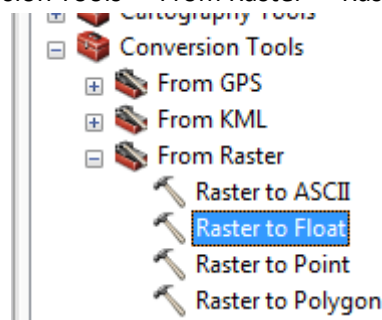
b. Converting your data into something LSDTopoTools can understand (.flt format)

- i. Load your DEM into ArcMap. DEMs come in all sorts of formats, and it is very likely you will have to do some modification before you can start working on the DEM with the Edinburgh topographic analysis toolkit. Firstly, you will have to make sure the DEM is in projected coordinates. That is, distances should be measured in something like metres rather than degrees. You can do this in ArcMap by going into the toolbox and using Data Management Tools -> Projections and Transformations -



> Raster -> Project Raster

- ii. The projection should be into a 'projected' (not geographic) coordinate system. I usually use UTM, using the WGS 1984 standard. You can look up the UTM zone of your site here: <http://www.dmap.co.uk/utmworld.htm>
- iii. Once you have a projected DEM, the next step is to convert it into float format. The Edinburgh software reads **ONLY** float and ascii format DEMs, and ascii DEMs are **huge** and are not recommended. We have retained this format only for the purposes of bug-checking. To convert a DEM into float format in ArcMap use the Conversion Tools -> From Raster -> Raster to Float tool:



- iv. The float DEM is comprised of 2 files: one with the extension flt and one with the extension hdr. You will need both of these files.
- v. Note that LSDTopoTools can also work with ascii DEM but this is not preferred since they take up so much space compared to .flt files.

5. Compiling the two driver functions

The tools for topographic analysis are distributed as c++ source code. Before you run these tools you will have to compile them. Compiling means that you translate from source code, which looks a bit like English, to machine code, which is all 1s and 0s. To do this you use a compiler. If you have a Linux

system all you really need to do is navigate to the folder containing the source files in a terminal window. If you have a Windows computer follow the instructions in the appendix to get a compiler installed. Then you compile using a command prompt (you can find this in windows 7 by typing 'command prompt' into the search for programs and files field in the start menu).

- a. This should be fairly easy if you have your compiler installed. If you are working in the Cygwin environment (see appendix) you will also need the make utility. First you compile the junction selection tool. Do this by typing:

```
make -f chi_step1_write_junctions.make
at the command line and then hitting enter.
```

- b. Now for the next tool, which creates something called a .chan file (you don't need to worry about what this is yet). The next tool is compiled with

```
make -f chi_step2_write_junctions.make
```

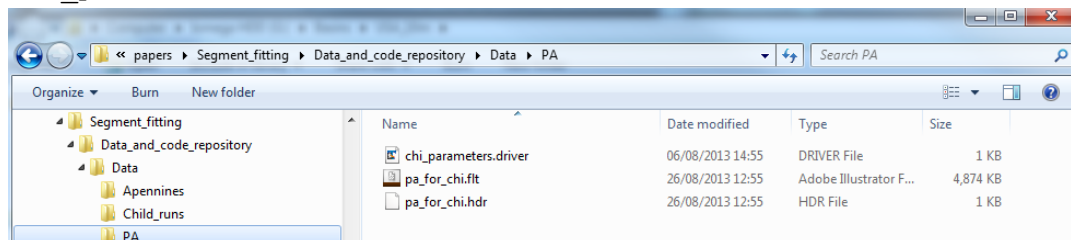
- c. The source code is now compiled on your system!

6. Running the channel network extraction

The channel extraction code requires two steps. In the first step, the toolkit takes the raw DEM and prints several derived datasets from it. The main dataset used for the next step is the junction index dataset. The second step involves selecting a junction from which the chi analysis proceeds.

a. Writing junctions

- i. First, create a folder for your DEM. Make sure both the .flt and the .hdr file are in this folder. Then you need to create a file that tells the analysis the name of the DEM and a few parameters. You can name this file anything you like but I have called mine `chi_parameters.driver`:



- ii. The driver file must contain three lines. The first line is the name of the DEM **without the extension**. In this example the name is 'ganga'. The next line is a minimum slope for the fill function. The default is 0.0001. The third line is the threshold number of pixels that contribute to another pixel before that pixel is considered a channel. You can play with these numbers a bit, in this example, I've set the threshold to 300 (it is a 10m DEM so in the example the threshold drainage area is $3 \times 10^5 \text{ m}^2$). Here are the first 3 lines of the file:

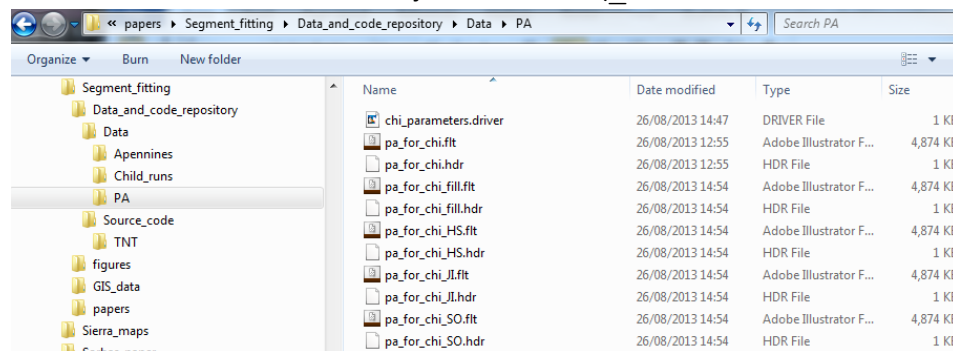
```
chi_parameters.driver
pa_for_chi
0.0001
300
```

- iii. Once you have done this, you need to run the driver program. The driver program is called `chi1_write_junctions.exe`. It takes 2 arguments. The first is the path name into the folder where your data is stored, and the second is the name of the driver file. To run the program, just type the program name and then the path name and driver

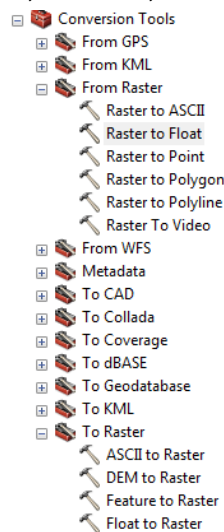
file name. The path has to end with a '/' symbol. If you are working in Linux, then the program name should be proceeded with a './' symbol. Here is a typical example:

```
./chil_write_junctions.exe
/home/smudd/papers/Segment_fitting/Data_and_code_repository/
Data/PA/ chi_parameters.driver
```

- iv. In later sections you will see that the driver file has the same format for all steps, but for this step only the first three lines are read. The driver file has a bunch of parameters that are described later but there is a file in the distribution called `Driver_cheat_sheet.txt` that has the details of the parameter values.
- v. This is going to churn away for a little while. If you've used incorrect filenames the code should tell you. The end result will be a large number of new files: The code prints a filled DEM (with `_fill` in the filename), a hillshade raster (with `_HS` in the filename), information about the stream orders (file with `_SO` in the filename) and a file with information about the junctions (`_JI` in the filename).



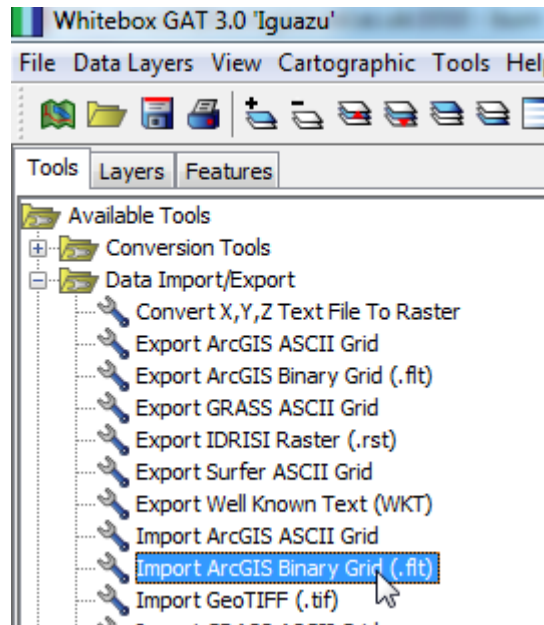
- vi. You will then need to load these files into arcmap and look at them. You can't load the .FIPickle file but you can load all the other files. You'll have to convert them from .flt format, however, so in ArcMap use the Conversion Tools -> To Raster -> Float to Raster



tool:

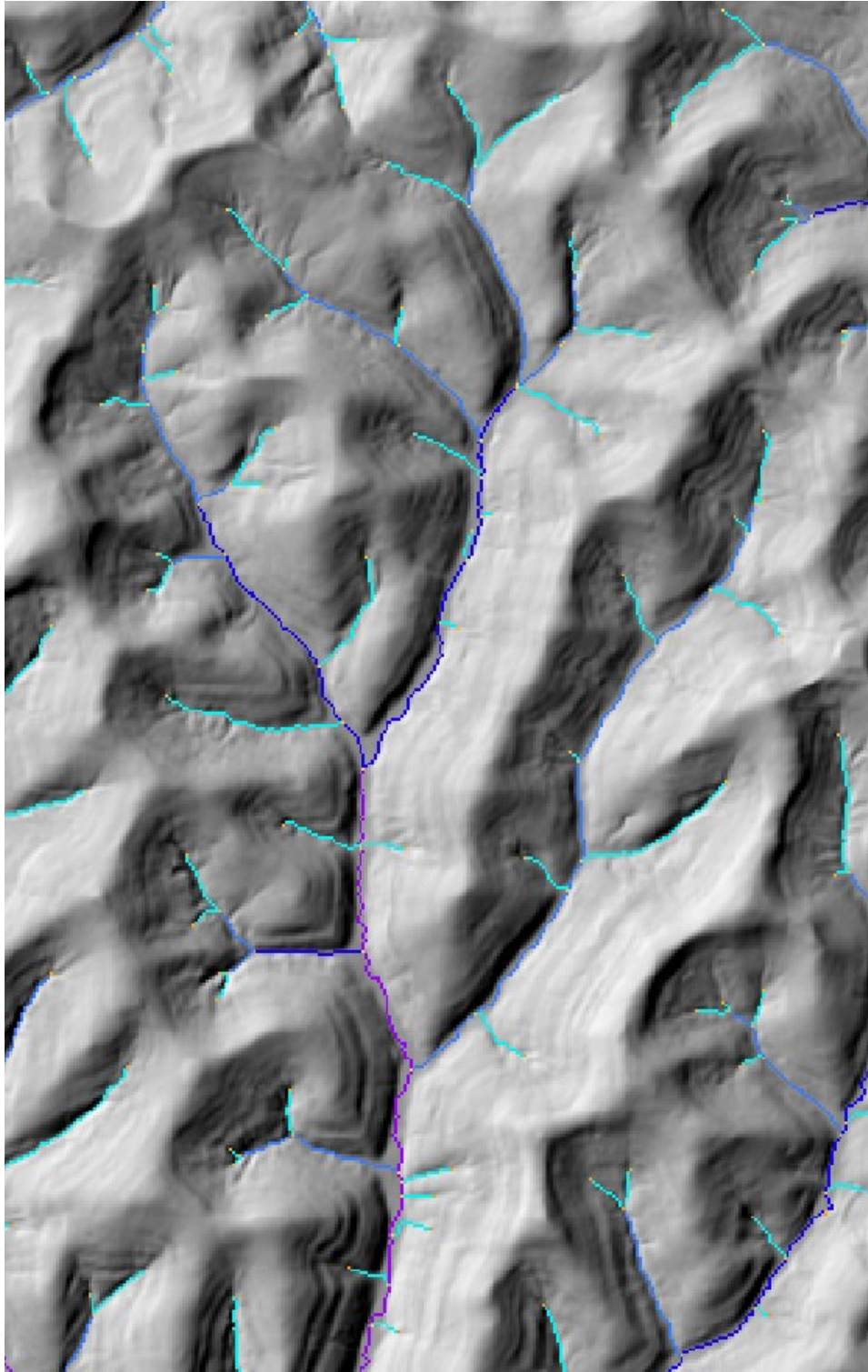
- vii. An alternative to ArcMap is Whitebox (<http://www.uoguelph.ca/~hydrogeo/Whitebox/>) which has the advantage of being open source. You can import .flt files using the


import/export data tool menu:



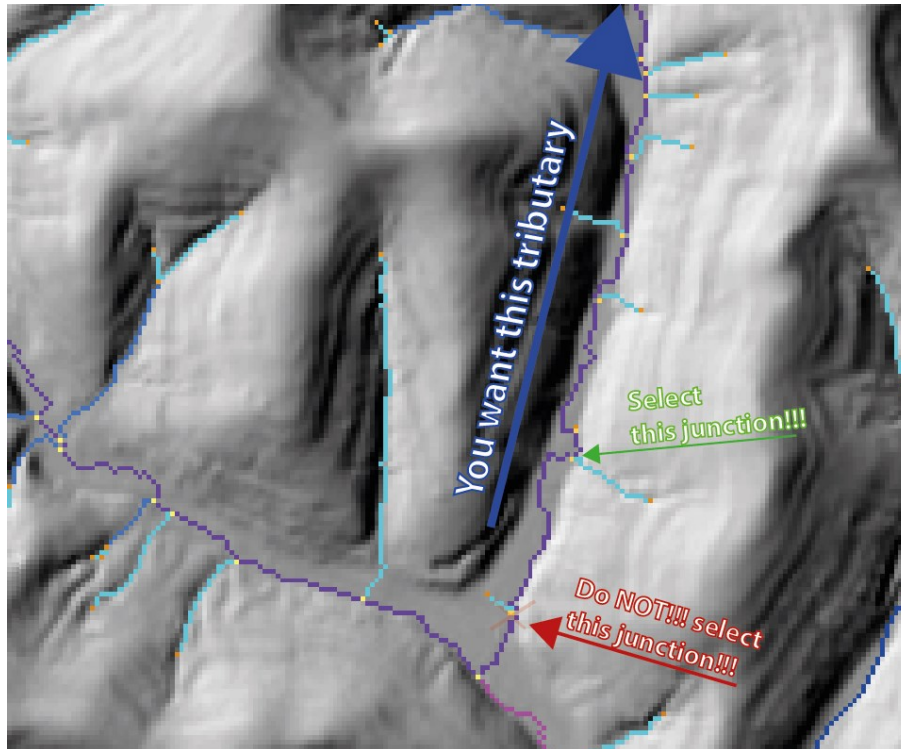
- viii. You want to look at the channel network and junctions. So at a minimum you should import the hillshade raster, the stream order raster (_SO in filename) and the junction index raster (_JI in filename) into your preferred GIS. The stream order raster will display the channel network, with each channel having a stream order. **The junction index file is the key file, you will need information from this file for the next step.** In the image below, the channel network is in cool colours and the junctions are in warm colours.

Each junction has a unique integer value, called the junction index.

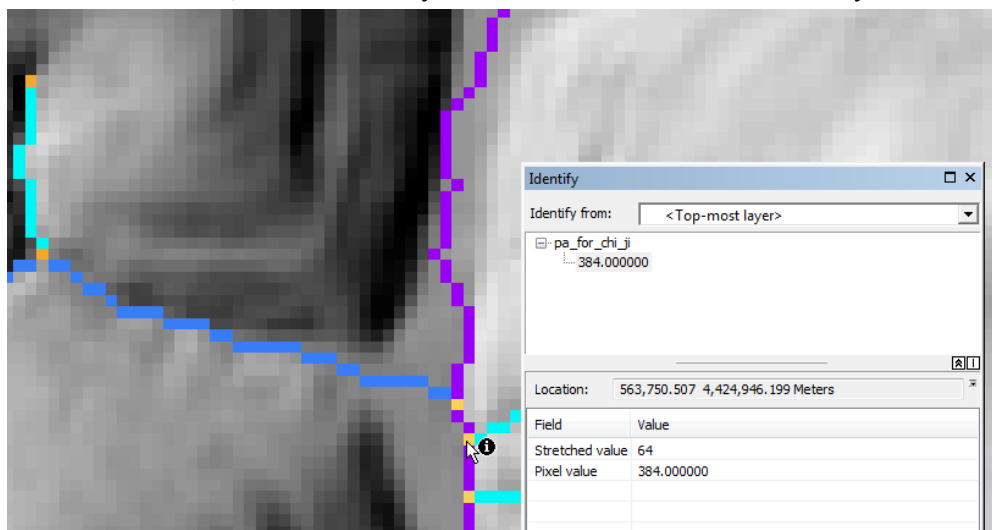


- ix. Now, find the part of the map where you want to do the chi analysis. You need to choose the junction at the downstream end of the channels where you will do your analysis. Use the inspection tool  to get the number of the junction that you want to use as the lowest junction in the channel network.
- x. **THIS IS EXTREMELY IMPORTANT!!!** Due to some details in data organization which are somewhat dull, when you select a junction the algorithm will trace down to the NEXT DOWNSTREAM junction and then work up from there. **HOWEVER if you are working**

up from a tributary that intersects to the main stem you need to select the junction 2 JUNCTIONS UPSTREAM from the mainstem. If you don't do this the network code will follow the mainstem as a tributary!! See the image:



- xi. In the below case, I have found junction number 384. Record this junction number.

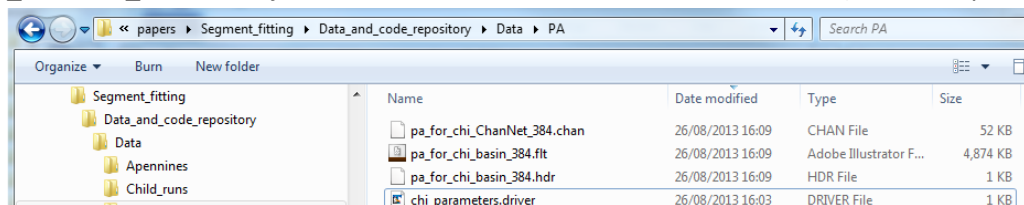


b. Extracting the .chan file

- Now that you have the junction number, you need to run the second program. Before you run this program, you need to write a file that contains the parameters for the chi analysis.
- The first 3 lines of this file **MUST** be the same as the driver file in step 1.* The code doesn't check this so you need to make sure on your own this is the case.
- The next two rows of the driver file are the junction number from which you want to extract the network (see above, this is actually the 2nd most downstream junction in the network due to some nuances of data organization which are too tedious to describe

here) and something that controls how the channel network is ‘pruned’. This is the ratio in area between the main stem and a tributary that must be exceeded for a tributary to be included in the analysis. If this number is 1 you only get the main stem. The smaller the number the more tributaries you get. A reasonable number seems to be ~ 0.02 .

- iv. There can be more information in the driver file (for example, parameters for a chi analysis), but the channel network extraction program will ignore these; it only looks at the first 5 lines of the driver function.
- v. From here you run the program `chi2_write_channel_file.exe`. You need to include the path name and the name of the chi parameter file. In Linux the program should be proceeded with `./`. Here is an example:
`./chi2_write_channel_file.exe`
`/home/smudd/papers/Segment_fitting/Data_and_code_repository/`
`Data/PA/ chi_parameters.driver`
- vi. This will generate a DEM with `_basin_` in the filename. Immediately before the `.flt` extension the junction number will also be listed. In addition a `.chan` file (with `_ChanNet_` and the junction number in the filename, see below) will be printed:



- vii. The `_basin_` file is a raster containing the outline of the contributing pixels to the basin drained by the extracted channel network.

c. Format of the `.chan` file

The segment fitting algorithm (part B) works on a ‘channel’ file (we use the extension `.chan` to denote a channel file). The channel file starts with six lines of header information that is used to reference the channel to a DEM. If the channel is not generated from a DEM these six rows can contain placeholder values. The six rows are

```
Nrows <- number of rows
Ncols <- number of columns
Xllcorner <- location in the x coordinate of the lower left corner
Yllcorner <- location in the y coordinate of the lower left corner
Node_spacing <- the spacing of nodes in the DEM
NoDataVal <- the value used to indicate no data
```

This header information is *not* used in the segment analysis; it is only preserved for channel data to have some spatial reference so that scripts can be written to merge data from the channel files with DEM data.

The rest of the channel file consists of rows with 9 columns.

- The first column is the channel number (we use c++ style zero indexing so the main stem has channel number 0).
- The second column is the channel number of the receiver channel (the channel into which this channel flows). The mainstem channel flows into itself, and currently the code can only

handle simple geometries where tributaries flow into the main stem channel *only*, so this column is always 0.

- The third column is the node number on the receiver channel (which, recall, **must be the main stem**) into which the tributary flows. The main stem is defined to flow into itself. Suppose the main stem has 75 nodes. The third column would then be 74 for the main stem (because of zero indexing: the first node in the main stem channel is node 0. Nodes are organized from upstream down, so the most upstream node in the main stem channel is node zero. Suppose tributary 1 entered the main stem on the 65th node of the main stem. The third column for tributary 1 would be 64 (again, due to 0 indexing).
- The 4th column is the node index that refers back to the LSDFlowInfo object.
- The 5th column is the row in a DEM the node occupies.
- The 6th column is the column in a DEM the node occupies.
- The 7th column is the flow distance from the outlet of the node. It should be in metres.
- The 8th column is the elevation of the node. It should be in metres.
- The 9th column is the drainage area of the node. It should be in metres squared.

Many of these columns are not used in the analysis but are there to allow the user to refer the channel file back to a DEM. Columns are separated by spaces so rows will have the format

```
Chan_number receiver_chan receiver_node node_index row col flow_dist elev drainage_area
```

Here are the first few lines of an example file:

```
2907
3473
548517
4.40339e+06
10
-9999
0 0 1793 1619544 735 720 59819.3 408.117 30000
0 0 1793 1622700 736 719 59805.1 406.679 30300
0 0 1793 1625857 737 718 59791 404.598 31000
0 0 1793 1629014 738 717 59776.8 402.726 43900
0 0 1793 1632173 739 717 59766.8 400.542 45900
0 0 1793 1635333 740 717 59756.8 399.258 47100
```

Appendix: Installing a compiler

This section describes how to install the software on your Windows computer

Installing a compiler on a windows machine

There are several ways to install a compiler on a Windows machine but at Edinburgh we use the Cygwin environment (<http://www.cygwin.com/>).

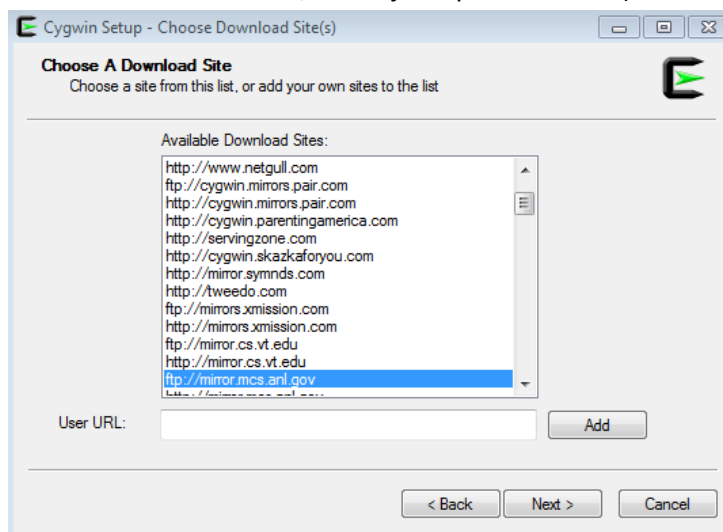
1. Go to the Cygwin website and download setup.exe:

Current Cygwin DLL version

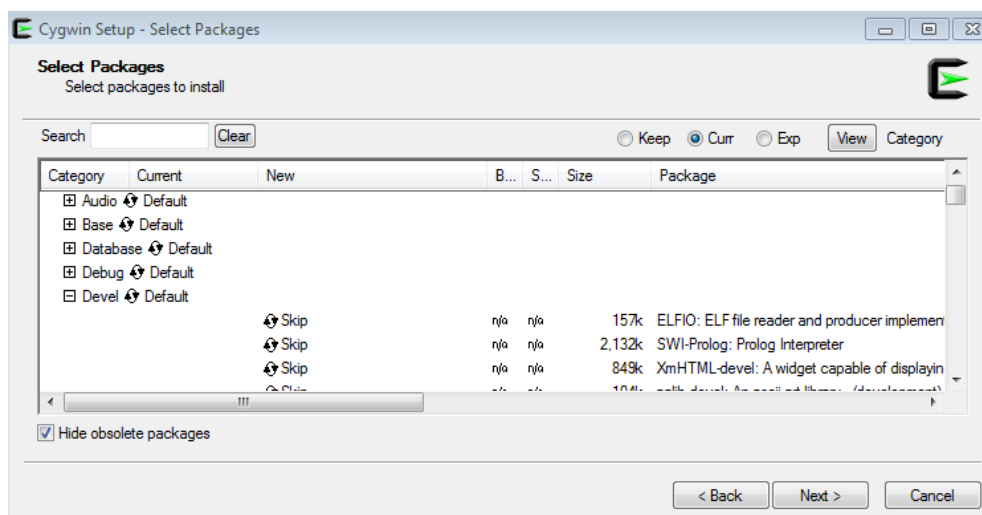
The most recent version of the Cygwin DLL is [1.7.16-1](#). Install it by running [setup.exe](#).



2. Once you download setup double click to run it:
3. You will get a bunch of warnings, but just keep clicking next until you get to a screen to choose a 'mirror' site, and just pick a site (it doesn't really matter which one):



4. You will get a window that asks you what packages you want to install. Then expand the 'devel' list of packages:



5. You toggle between installing and not installing by clicking these buttons: . The things that are essential to install are the g++ compiler and the 'make' utility.

3.4.4-999 Keep n/a ☐ 7,829k gcc-g++: C++ compiler

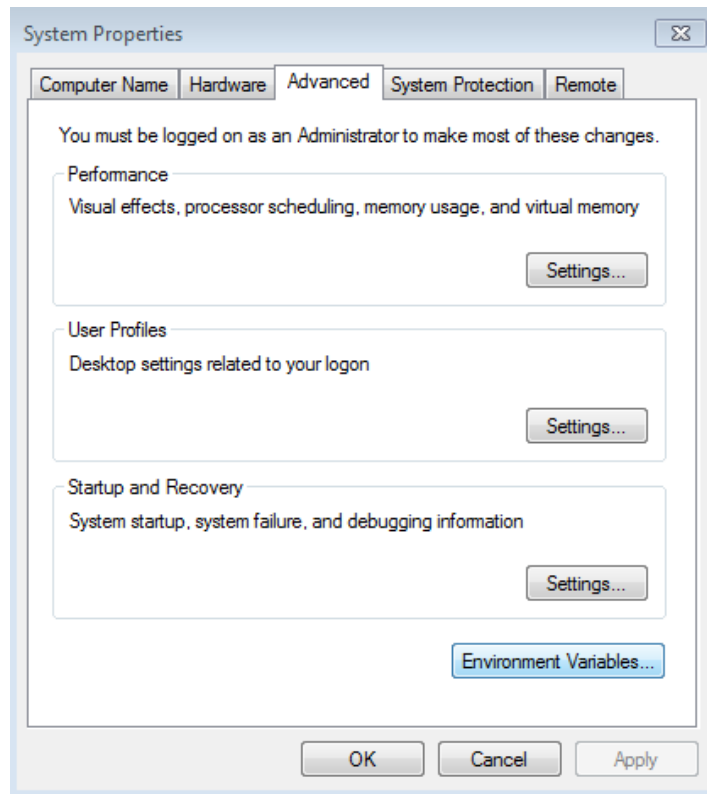
3.82.90-1 Keep n/a ☐ 442k make: The GNU version of the 'make' utility

The GNU debugger might also come in handy later.

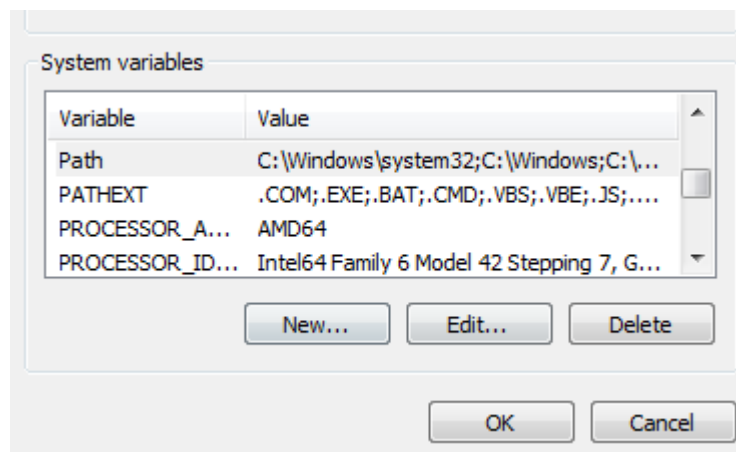
7.3.50-3 Keep n/a ☐ 5,600k gdb: The GNU Debugger

Note that as you learn about these tools you can always run setup again and install more stuff.

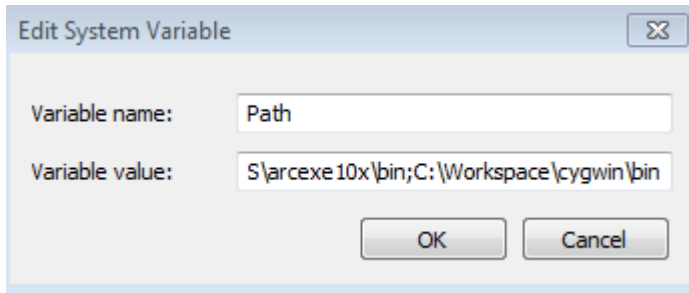
6. Once you have got g++ and make, just click next a few times and things will install.
7. Once installation is complete, you need to edit the PATH environment variable on your machine so it sees all the cygwin programs. If you are using windows 7, you can just type 'environment variables' into the search for program bar, and then click on the link to edit the variables. You will get this screen:



8. Click on the environment variables button and then click on path and then edit



9. Append the path with a semicolon and then the path to the cygwin bin folder:



The screenshot shows a standard Windows 'Edit System Variable' dialog box. The title bar reads 'Edit System Variable'. Inside the dialog, there are two text input fields. The first field, labeled 'Variable name:', contains the text 'Path'. The second field, labeled 'Variable value:', contains the text 'S\arcexe10x\bin;C:\Workspace\cygwin\bin'. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel'. A small icon in the top right corner of the dialog box represents a broken link.