THE UNIVERSITY OF TEXAS AT AUSTIN

NHD cross-section feature extraction

Richard Carothers

8 May 2015

CE 397 – Flood Forecasting

Contents

Figures and tables

1 Introduction

The driving goal of the National Flood Interoperability Experiment (NFIE) is to connect national flood hydrology with local emergency response in an understandable and actionable way. The five components of the NFIE serve as a roadmap to achieve this outcome and a structure for data storage and transmission, Figure 1.1.

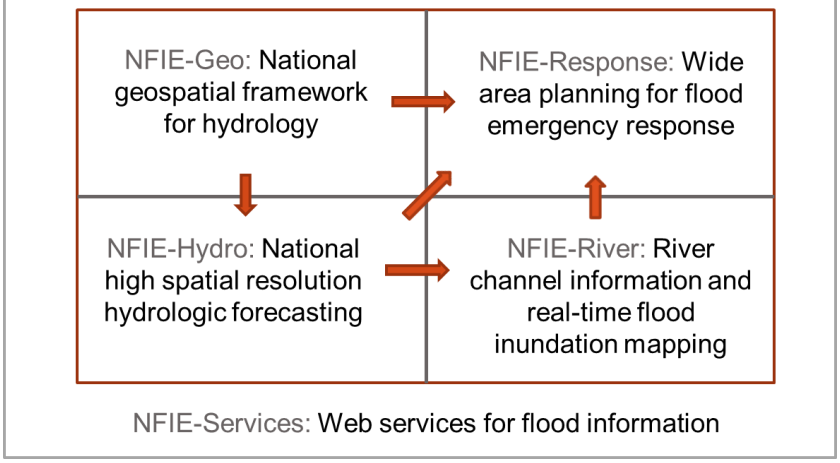

Figure 1.1 NFIE component framework (figure adapted from Dr. David Maidment)

Of the components that directly inform the NFIE-Response component and thus the emergency services community, both the NFIE-Geo and NFIE-Hydro have the required data in place such that they can be processed into actionable information (an exception to this is the limited national reach of the National Flood Hazard Layer). However, the detailed local channel information required for river modeling and inundation mapping is sparse and disparate in source when available. Local governments have models for select reaches, but major information gaps are present.

The most basic data shortfall is a lack of river channel geometry in the form of channel cross-sections for modeling input. Conventionally, curvilinear coordinate systems are required for these cross-sections. This address consists of a location of the cross-section within the reach (the reach station), a specific location along the cross-section itself (the cross-section station), and an elevation. These coordinates are shown as “s”, “n”, and “z”, in Figure 1.2, respectively. Generally not explicitly included in river models are the cross-section Cartesian coordinates, helpful for modeling and more general application.

2 Objectives

In order to address the need within the NFIE-River component for high resolution river cross-section data in local reaches, the goal of this project is to create a suite of tools that facilitate the creation of two cross-section feature classes within the ArcGIS framework based on river flow lines and a digital elevation model (DEM) and containing both curvilinear and Cartesian coordinates. The tools should be simple to operate and minimally parameterized.

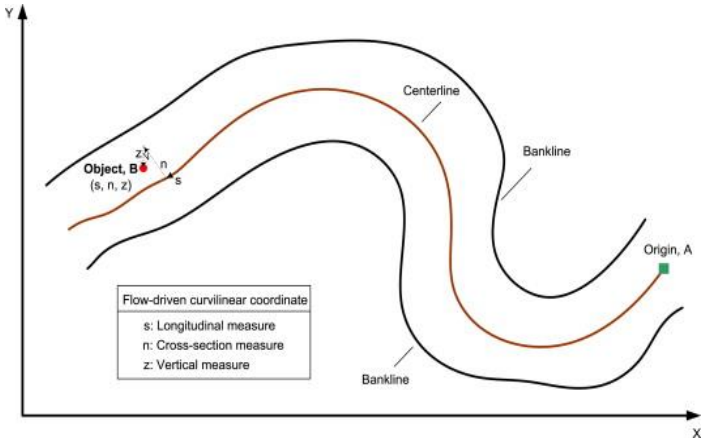

Figure 1.2 Reach curvilinear coordinate system (figure from Kim et al, http://www.sciencedirect.com/science/article/pii/S1364815214003570)

Furthermore, the required inputs should be intuitive and readily available while the outputs should be easily understandable and deliver the required information.

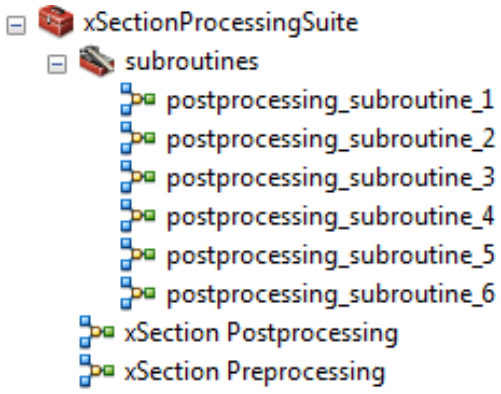

Figure 3.1 ArcGIS cross section processing toolbox

2.1 Inputs

The first input data are the NHD Plus dataset 100k resolution flow lines for the given area. This data is easily acquirable online and carries pertinent information about individual reaches such as the common identifier (COMID) and the reach connectivity. Also necessary as an input are digital elevation data for the desired region of analysis. This may come in the form of a high resolution Lidar based DEM when available or may be selected from the coarser National Elevation Dataset (NED). Regardless, topographic information is required in order to determine the z-component of the cross-sections.

2.2 Outputs

The primary feature class required is a point feature class that represents the cross-sections as a series of points. The necessary data contained within are at minimum the reach COMID and the curvilinear and Cartesian coordinates for each given point. Furthermore, the curvilinear coordinates need to maintain accepted convention for proper modeling as described before. In addition to this information, a unique cross-section ID (xSecID) will provide a more intuitive label of the cross-section within the context of the output dataset and likewise help with processing.

In addition to this point feature class, a line feature class representing the cross-sections as cut lines is helpful for visualization. The line feature class contains the COMID, xSecID, and reach station of each cross-section line.

3 Toolbox, models, and processes

The general methodology for developing cross-sections involves three steps: preprocessing to prepare the data, striking of the cross-sections, and post-processing the cross-sections to assign the required attribute information and clean up the output feature classes. The preprocessing and post-processing steps are completed in ArcGIS using the xSectionProcessingSuite toolbox, Figure 3.1. The striking of the cross-section is done using an external MATLAB code heavily modified from a subroutine in the GeoNet2.0 feature extraction toolbox[1]. The overall inputs and outputs are those described in the Objectives. The user inputs the NHD flow lines in conjunction with a DEM and the process outputs the line and point cross-section feature classes.

[1] https://sites.google.com/site/geonethome/home

3.1 Preprocessing

As the general procedure was initially designed around the adapted GeoNet code, the preprocessing tool within the ArcGIS toolbox prepares the data be processed within that code. Figure 3.2 shows the parameter menu for running this tool. A page view of the ArcGIS model builder model is provided in Figure 3.3.

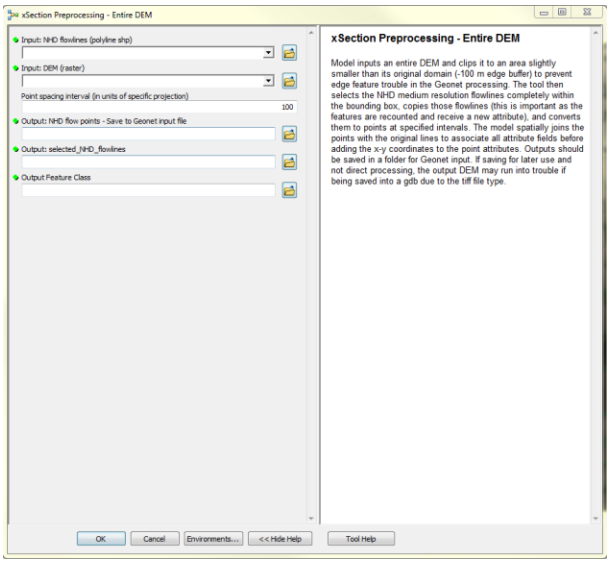

Figure 3.2 Preprocessing tool filepath and parameter menu

Both preprocessing and post-processing tools are dependent on the conversion of lines to consecutive points. To accomplish this, a script for ArcGIS was found online[2]. The script utilizes ArcGIS features that derived from toolbars and thus unable to be plugged directly into the model builder as normal tools are. Currently, it is a requirement to download this script and add this tool in order for the processing suite to function.

3.1.1 Inputs and parameters

As the initial step of processing, the inputs for the preprocessing tool are the overall inputs for the toolbox. Processing requires a raster DEM covering the area and the regional NHD flow lines feature

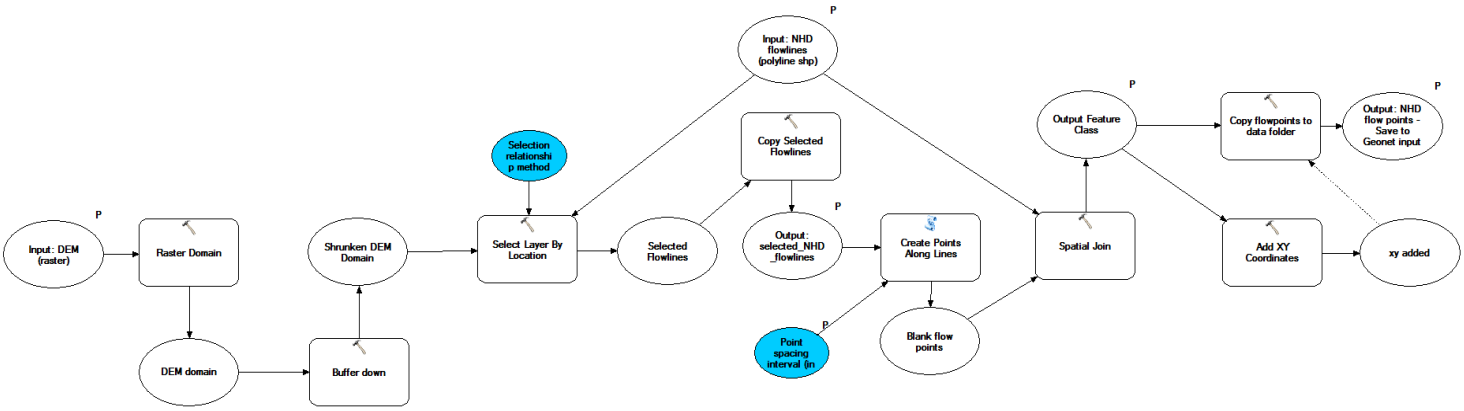

Figure 3.3 ArcGIS workflow model of data preprocessing

class for the area to be processed. In order to run the tool, both the flow lines and the DEM need to be in the same file geodatabase (gdb) and in the same projection. Furthermore, the projection should be in the units desired for output. UTM projections are recommended.

The only parameter required for the preprocessing run is a point interval. The flow lines are converted into a series of consecutive flow points at this specified interval. In the case of Figure 3.2, 100 m was

[2] http://www.arcgis.com/home/item.html?id=a2a41c8345e24ab6a9dd2ae215710b39

chosen. In this way, cross-sections can be struck at every 10 flow points (in the following GeoNet processing) allowing for a cross-section at every 1 km.

3.1.2 Processing and Outputs

In the current version of the tool, a polygon mask of the DEM is temporarily created. The mask is buffered down 100 m, and the flow lines completely within that mask are selected based on their location. The buffer down is required to prevent processing errors when determining cross-section elevations along the edge of the DEM. The selected flow lines are copied over again for later use, ideally in the same gdb. They are also converted to points at the desired interval. These flow point features are spatially joined with the flow lines so that each point shares all attributes with the flow lines. Finally, the flow point feature class is written in the gdb for later use and also as a shapefile in the selected GeoNet folder for continued processing.

3.2 GeoNet MATLAB code

Currently, cross-sections are actually struck within an external MATLAB code using a heavily modified version of a subroutine in the GeoNet2.0 feature extraction toolbox. The code comes in the form of an executable file and a run file (currently titled xSectionExecTest.m and xSectionRunTest.m, respectively). The executable file calls the run file, and, therefore, only it needs user modification. Using the shapefile prepared by the preprocessing tool, the cross-sections are drawn at specific intervals for and lengths based on user inputs. The resulting cross-section endpoints are printed out in csv format.

3.2.1 Inputs and parameters

In order to read shapefiles directly using MATLAB, the MATLAB mapping toolbox or some variant is required[3]. However, as the toolbox has a non-trivial cost associated with it, the processing code the shapefile reading features of the free M_Map package[4]. Download and installation of this package is required to run the code. Slight syntactical changes can be applied to the code if the mapping toolbox is already owned.

```
%% Point to source DEM data
%  Edit this to point to your source TIFF - the path is deduced from it
Parameters.demDataFileName = 'travisDEM';    % File name fo filtered DEM
Parameters.channelDataFileName = 'travisFlowPoints';

%%% Input/Output file locations. Replace string with filepath to desired output
%%% folder.
% Input filepaths
Parameters.demDataFilePath = fullfile...
    ('C:\Users\Richard\Documents\classes\CE 397 - Flood Forecasting\termProject\matlab\data');
Parameters.channelDataFilePath = fullfile...
    ('C:\Users\Richard\Documents\classes\CE 397 - Flood Forecasting\termProject\matlab\data');

% Output filepaths. If no such folder exists, one will be created.
Parameters.fileOutputPath = fullfile...
    ('C:\Users\Richard\Documents\classes\CE 397 - Flood Forecasting\termProject\matlab\results'...
    , Parameters.channelDataFileName);
```

Figure 3.4 Filepath and filename inputs in the executable file

With the M_Map package installed, the code can correctly import the shapefile from the location specified in the preprocessing phase. This input filepath and filename for the flow point shapefile are specified in the executable file seen in Figure 3.4. Likewise, there is a filepath for the output csv that needs to be entered. In the current manifestation of this code, there are also filepaths and filenames associated with the DEM. These are not used, but cannot be left blank as processing errors will occur. In future variants of the code, these will be removed and the executable file streamlined.

[3] http://www.mathworks.com/products/mapping/
[4] http://www.eos.ubc.ca/~rich/map.html

The four user input parameters are also identified in the executable file, Figure 3.5. The first parameter, Parameters.skipPixels, indicates how many flow points to skip before the process of striking cross-sections begins. This is important depending on the value of the third parameter, Parameters.reultantVector. The resultant vector parameter indicates how the angle of the cross-section will be struck. The orthogonal angle to the line formed by the flow points on each side of the flow point where the cross-section will be struck is used as the path of the cross-section, assuming a value of one as indicated in the figure. Were the value of resultant vector increased to two, the flow point two points away from the cross-section flow point on either side would be used instead. Considering again the pixels skipped, the resultant vector cannot be calculated unless there are more pixels skipped than used for the resultant, or processing errors will occur.

```
%%% xSection parameters
% Skip pixels: tells how many initial shp pts to skip.
% Cross section gap: how many shp points to skip between xSections drawn
% Resultant Vector: shp points used to determine orthogonal angle for xSect
% xSection Length: Actually half the length. This counts point out in each
% direction from centerline point.
Parameters.skipPixels = 2;          % Rec keep as 2 or greater
Parameters.crosssectiongap = 10;
Parameters.resultantVector = 1;     % Must be less than skipPixels
Parameters.crosssectionLength = 50;  % Cross section half length
```

Figure 3.5 Executable file parameters

Regarding the second and fourth parameters, Parameters.crosssectiongap and Parameters.crosssectionLength, these are more straight-forward than the others. The gap simply indicates how many flow points will be in between each cross-section. As discussed in the parameters of the preprocessing method (Section 3.1.1), 10 has been chosen as the skip pixel value in this situation so that a cross-section is struck every 1 km. The cross-section length actually indicates half of the length of the cross-section. The units are those of the projection used. The code actually counts the desired length along the orthogonal out in each direction from the cross-section flow point.

A final feature of the executable file is the ability of to write several different output files based on a series of output switches, Figure 3.6. There are currently three binary switches indicating the desired program csv output. Parameters.printXSections will produce a csv with all points along (at each meter or projected unit) each cross-section, Parameters.printMidpointX will print the points along the cross-section only at the middle of each reach (reaches are determined by COMID), and Parameters.printEndpoints will print just the endpoints of each cross-section. If .printMidpointX and .printEndpoints are turned on, a separate file of the endpoints for the mid reach cross-sections will be generated. Activating the .printXSections switch will significantly increase processing time due to the increased number of points that are required to be written (from about 30 seconds for endpoints only to 7 minutes for the Travis County dataset with these parameters). Furthermore, the post-processing tool currently uses only the endpoint csv file (reach midpoints or all cross-sections) to produce the final outputs.

```
%%% Print output switches
%%% NOTE: If midpoints and endpoints are both selected, a 4th csv will be
%%% printed (filename_MidpointXEnds) with the midpoint xSection endpoints
% Switch to print xSections
Parameters.printXSections = 0;

% Switch to return only xSections at reach midpoints in a separate file
Parameters.printMidpointX = 0;

% Switch to print xSection endpoints again in a separate file
Parameters.printEndpoints = 1;
```

Figure 3.6 Executable file output switches

3.2.2 Processing and outputs

The general processing has basically been described in the description of the parameters. The code reads the flow point shapefile input from the preprocessing output and marches

along each individual reach using the rules put forth by the user prescribed parameters. Csv files are written based on the user identified switches.

	A	B	C	D	E	F	G	H
1	FID	COMID	HydroID	xSECT NUI	xSECT PT	POINT_X	POINT_Y	POINT_Z
2	1	5671165	5671165	1	1	634766	3378315	0
3	1	5671165	5671165	1	101	634727	3378407	0
4	1	5671165	5671165	2	1	633783.7	3378276	0
5	1	5671165	5671165	2	101	633767.1	3378374	0
6	2	5671187	5671187	1	1	632748	3378044	0
7	2	5671187	5671187	1	101	632652.9	3378075	0
8	2	5671187	5671187	2	1	632100.9	3377351	0

Figure 3.7 Endpoint csv output file from GeoNet processing

There are, however, two protocols that are worth mentioning. Part of the processing procedure is to run through each reach and calculate how many cross-sections will be required based on the number of flow points and a calculation based on the .skipPixels and .crosssectiongap parameters. If the number of cross-sections is calculated as zero or negative, no cross-sections will be drawn, and the reach will effectively be skipped. If that value is one, the cross-section will automatically be drawn at the middle of the reach instead of two points above the reach discharge.

The output csv file resulting from processing the data with this code is seen in Figure 3.7. The FID and COMID both refer to the specific reach, while the xSECT NUM and xSECT PT refer to locations of the cross-section within the reach and individual point within the cross-section. As this version of the output csv refers to an endpoint output file, the xSECT PT locations are the first and last points within the cross-section. Output coordinates refer to the projected coordinate system of the input flow points. The presence of POINT_Z is an artifact of when DEM processing occurred within this code as opposed to the post-processing as it is currently handled. Finally, HydroID is a separate identifier that, in this case at least, is the same as the COMID. More generally, the presence of the HydroID in the output file represents a place holder for any other information from the input shapefile that the user may want to maintain through the cross-section striking process.

3.3 Post-processing

The post-processing represents an ArcGIS model combining the outputs of both the pre and GeoNet processing phases. This includes three inputs from the preprocessing and the csv generated in the GeoNet processing. With two parameters, the model produces the desired two feature classes as outputs, Figure 3.8.

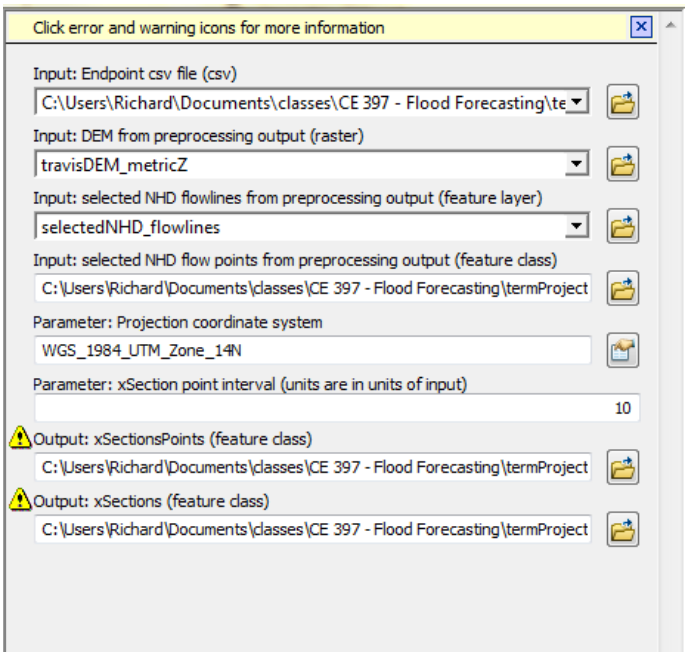

Figure 3.8 Post-processing tool filepath and parameter menu

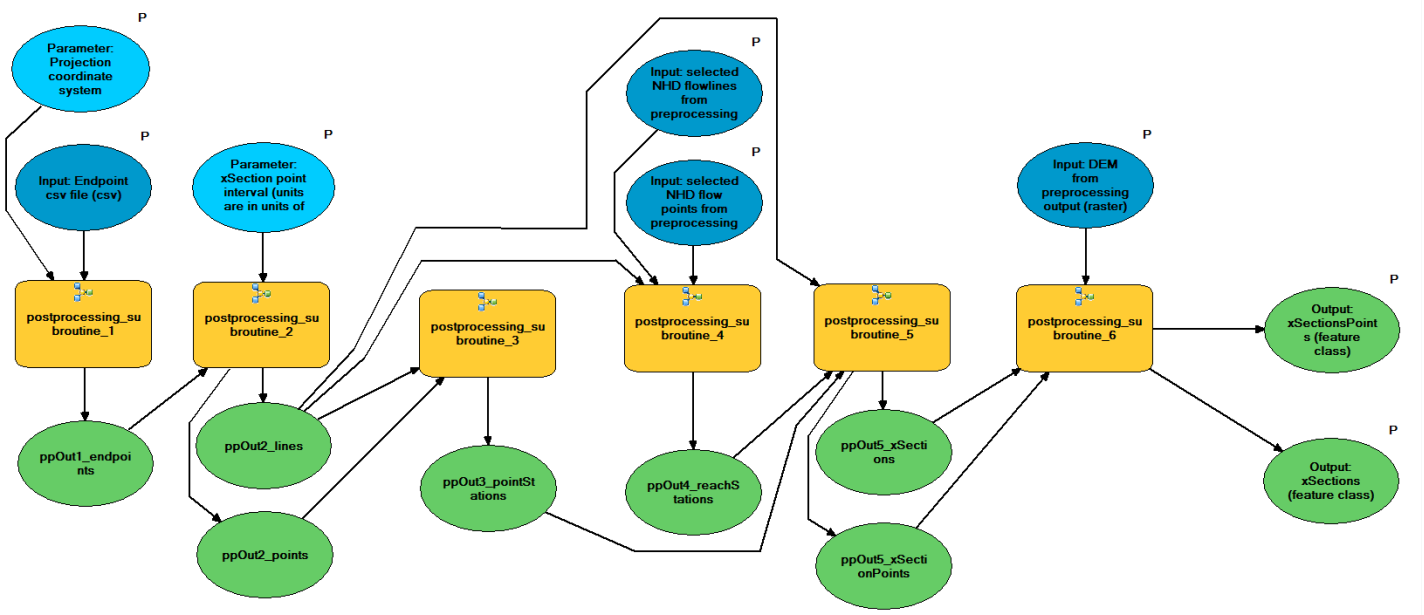

Figure 3.9 ArcGIS workflow model of combined post-processing

As seen in the model workflow, Figure 3.9, it is actually composed of six individual subroutines. In this workflow, external inputs or parameters are represented in shades of blue and placed above the yellow subroutines while the subroutine generated inputs and outputs are shown in green below. The two feature classes farthest to the right are the primary model outputs if the cross-section lines and points. Individual subroutine models can be seen in Appendix A in the order in which they are processed**.**

3.3.1 Inputs and parameters

The primary input for the combined post-processing model is the csv generated during the GeoNet processing phase. This file identifies the endpoints of all cross-sections in the selected system. This is combined with three of the preprocessing outputs: the DEM (actually not a preprocessing output in the current version, but still within the gdb from initial processing), the selected flow lines from within the DEM mask, and the flow points generated at intervals along the flow lines (see Section 3.1.2 for reference).

Two parameters are required at this point: the projection used for the flow lines to be reapplied to the cross-section endpoints, and the interval at which cross-section points are to be generated. The second of these is an important consideration as it will heavily influence the required processing time and the resolution of the cross-sections.

3.3.2 Subroutine processing

Table 3.1 Post-processing subroutine overview

Post-processing Subroutine	General Process Description
1	Converts the csv to an endpoint feature class
2	Converts the endpoints to cross-section lines and points
3	Determines the point stations along each cross-section
4	Determines the reach stations for each cross-section
5	Condenses information and cleans up feature class attributes
6	Adds Cartesian coordinates (x,y,z) to the point feature class

The six post-processing subroutines are generally expansive in workflow (again, see Appendix A for workflows). The specifics of each process can be complex, but the general ideas of each are summarized in Table 3.1. A more detailed analysis of each subroutine is not especially beneficial in this synopsis.

3.3.3 Outputs

The combined post-processing model creates the cross-section point and line feature classes described in the Objectives. The line feature class, "Output: xSections" in the workflow Figure 3.9, represents the actual cross-section lines as they are struck along each reach. As seen in Figure 3.10, the attributes used to define each cross-section are the xSecID, COMID, and the Station_in_Reach (along with several general and mandatory ArcGIS feature descriptors). Again, the xSecID is a unique cross-section ID that describes the reach within the feature class (the integer part) and the cross-section within that reach (the decimal). The COMID refers to the reach as it is identified in the NHDPlus. Finally, the Station_in_Reach represents half of the curvilinear coordinates and, in accordance with convention, identifies how far upstream the cross-section is from the reach discharge in the units of the projection being used.

xSections

OBJECTID *	Shape *	xSecID	COMID	Station_in_Reach	Shape_Length
1	Polyline Z	1.001	5671165	129.849477	100.000071
2	Polyline Z	1.002	5671165	1129.849576	100.000039
3	Polyline Z	2.001	5671187	164.980677	100.00002
4	Polyline Z	2.002	5671187	1164.980577	100.000066
5	Polyline Z	2.003	5671187	2164.980577	100.000053
6	Polyline Z	3.001	5671185	117.042477	99.999888
7	Polyline Z	4.001	5671181	147.736977	100.000058
8	Polyline Z	4.002	5671181	1147.736976	100.000007
9	Polyline Z	5.001	5671189	106.709877	99.999943
10	Polyline Z	5.002	5671189	1106.709877	100.000092
11	Polyline Z	5.003	5671189	2106.709877	100.00004
12	Polyline Z	5.004	5671189	3106.709777	100.000052

Figure 3.10 Cross section line feature class attribute table

The point feature class, "Output: xSectionsPoints" in the workflow Figure 3.9, details the Cartesian location of every point in each cross-section as well as the entire curvilinear location of the cross-section point within the specific reach. Figure 3.11 shows the attribute table for this feature class. The xSecID, COMID, and Station_in_Reach are as they were described previously. Additionally, the Station_in_xSection identifies the remaining curvilinear coordinate, the station of each point along the individual cross-section. Likewise following convention, this station refers to the distance along the cross-section from the left hand side while looking downstream with the flow. The units are those of the

xSectionPoints

OBJECTID *	Shape *	xSecID	COMID	Station_in_xSection	Station_in_Reach	POINT_X	POINT_Y	POINT_Z
1	Point	1.001	5671165	0	129.849477	634766.032	3378314.5802	191.873053
2	Point	1.001	5671165	10.000003	129.849477	634762.126303	3378323.785933	191.809623
3	Point	1.001	5671165	20.000006	129.849477	634758.220606	3378332.991667	191.782464
4	Point	1.001	5671165	30.000009	129.849477	634754.314908	3378342.1974	192.652999
5	Point	1.001	5671165	40.000012	129.849477	634750.409211	3378351.403134	190.995388
6	Point	1.001	5671165	50.000015	129.849477	634746.503514	3378360.608867	202.531643
7	Point	1.001	5671165	60.000018	129.849477	634742.597817	3378369.814601	194.310985
8	Point	1.001	5671165	70.00002	129.849477	634738.692119	3378379.020334	193.041729
9	Point	1.001	5671165	80.000023	129.849477	634734.786422	3378388.226068	188.772021
10	Point	1.001	5671165	90.000026	129.849477	634730.880725	3378397.431801	190.523007
11	Point	1.001	5671165	100.0001	129.849477	634726.975028	3378406.637535	196.834005
12	Point	1.001	5671165	100.0001	129.849477	634726.975	3378406.6376	196.833836
13	Point	1.002	5671165	0	1129.849576	633783.7419	3378275.5254	196.864802
14	Point	1.002	5671165	9.999996	1129.849576	633782.081061	3378285.386516	192.703008
15	Point	1.002	5671165	19.999992	1129.849576	633780.420221	3378295.247632	193.094887
16	Point	1.002	5671165	29.999988	1129.849576	633778.759382	3378305.108749	195.195537
17	Point	1.002	5671165	39.999985	1129.849576	633777.098543	3378314.969865	191.551883
18	Point	1.002	5671165	49.999981	1129.849576	633775.437703	3378324.830981	195.268478
19	Point	1.002	5671165	59.999977	1129.849576	633773.776864	3378334.692097	197.054999
20	Point	1.002	5671165	69.999973	1129.849576	633772.116024	3378344.553213	198.277582
21	Point	1.002	5671165	79.999969	1129.849576	633770.455185	3378354.41433	195.687316
22	Point	1.002	5671165	89.999965	1129.849576	633768.794346	3378364.275446	198.242207
23	Point	1.002	5671165	100	1129.849576	633767.133506	3378374.136562	198.786561
24	Point	1.002	5671165	100	1129.849576	633767.1335	3378374.1366	198.786616
25	Point	2.001	5671187	0	164.980677	632748.0407	3378044.4216	201.827615
26	Point	2.001	5671187	9.999998	164.980677	632738.522702	3378047.488799	201.931569

Figure 3.11 Cross section points feature class attribute table

projection. Finally, the Cartesian coordinates. POINT_X, POINT_Y, and POINT_Z, are in units of the projection.

4 Travis County test case results

The suite of tools described previously was applied to Travis County, Texas. The Basin 12 NHD flow lines were taken from the NHDPlusV2 website[5]. The DEM was developed by Cassandra Fagan based on a conglomeration of CAPCOG LAS datasets from throughout the county and has a resolution of roughly 3 m (10 ft). Both datasets were loaded into a default gdb and projected in as UTM 14N. Before running the processing tools, the DEM elevation value was also converted from feet to meters.

4.1 Preprocessing results

Running the preprocessing results using these two input datasets and a flow point interval of 100 m (see Figure 3.2 for preprocessing input menu). Figure 4.1 shows a map of the resulting flow points and flow lines enlarged in S Central Austin. The inset shows greater Travis County with the DEM and flow lines selected accordingly.

Total preprocessing time for the Travis County dataset was 4 minutes and 33 seconds. The longest step sin the process was the DEM masking and buffering. This length of time resulted in part from the sheer size of the DEM (23,067 by 22,114 pixels), but likely the greatest delay came from masking and buffering the irregular borders.

[5] http://www.horizon-systems.com/NHDPlus/NHDPlusV2_data.php

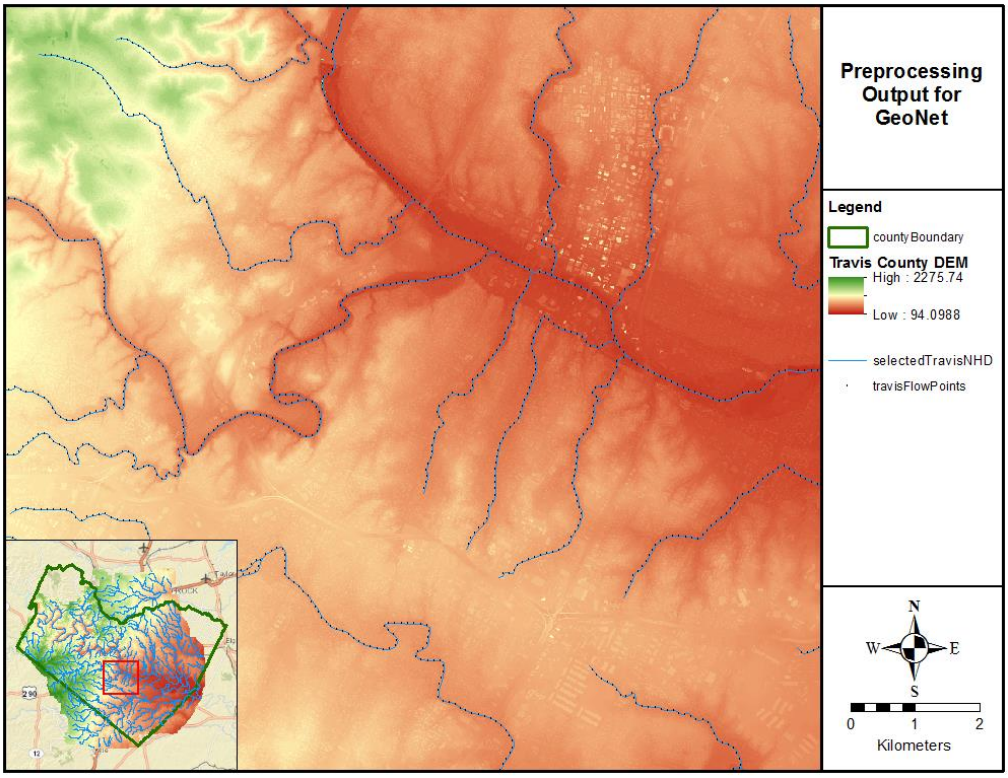

Figure 4.1 Preprocessing output flow points and flow lines in S Central Austin, TX

4.2 GeoNet processing results

The flow points mapped in Figure 4.1 were also written as a shapefile for GeoNet inputting. Figure 3.7 shows an excerpt of the actual GeoNet endpoint csv output for the Travis County dataset run. Total run time was 32.6 seconds, with the loading of the 17,300 flow points accounting for the majority of this time. Were other output files such as the entire cross-section flow points written, the processing time would take significantly longer. Likewise, altering the flow point spacing would increase or decrease the processing time depending on the resulting number of flow points.

4.3 Post-processing results

Post-processing was performed on the GeoNet csv results combined with the selected flow lines, flow points, and DEM output from preprocessing (see Figure 3.8 for post-processing run window). Cross-section point intervals were chosen to be 10 meters. Total processing time was 3 minutes and 33 seconds. No specific process was a limiting factor in the overall processing time.

The results of the first post-processing step, subroutine 1, can be seen in Figure 4.2. This really represents a feature class of the GeoNet output csv, the cross-section endpoints. Subsequent subroutines connect the endpoints resulting in the cross-section line feature class, Figure 4.3, and the cross-section point feature class, Figure 4.4, and they pass and calculate necessary attributes for each feature class. The real significance of the line and point cross-section feature classes lies in these attributes, tables of which were seen previously in Figure 3.10 and Figure 3.11, respectively. The attributes provide accessible addresses for each point in Cartesian coordinates for mapping and for each

point and line in curvilinearly referenced coordinates for conventional modeling applications. The entire suite of processes was completed county-wide with less than 9 minutes of computational time, Figure 4.5.

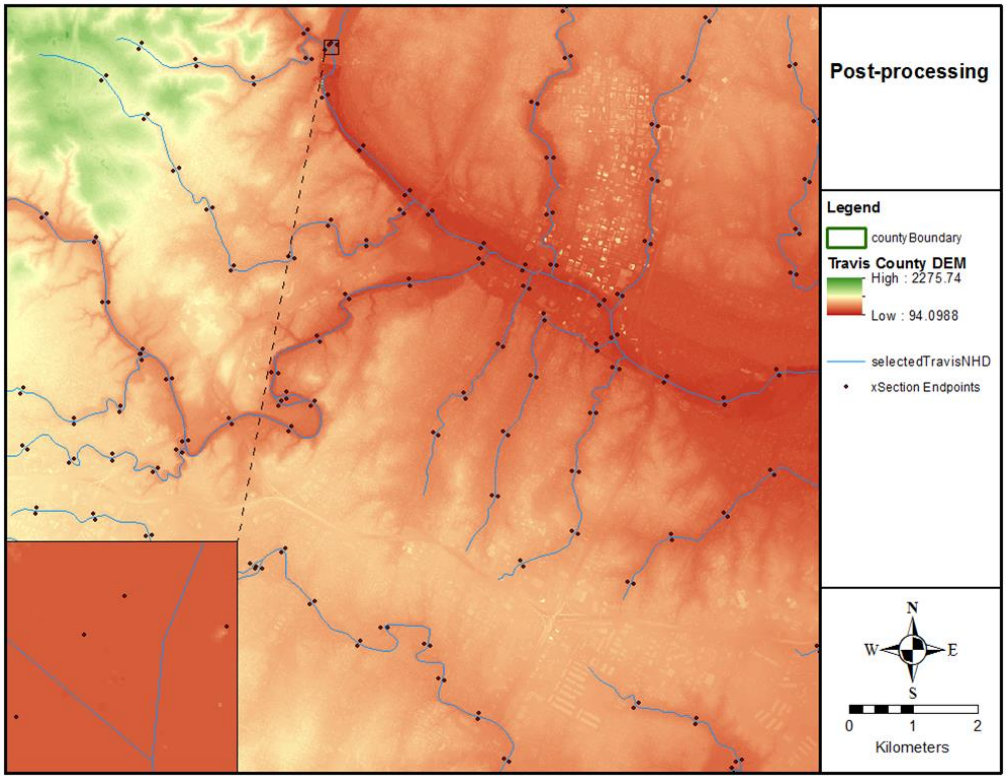

Figure 4.2 Post-processing subroutine 1 results, cross section endpoint feature class

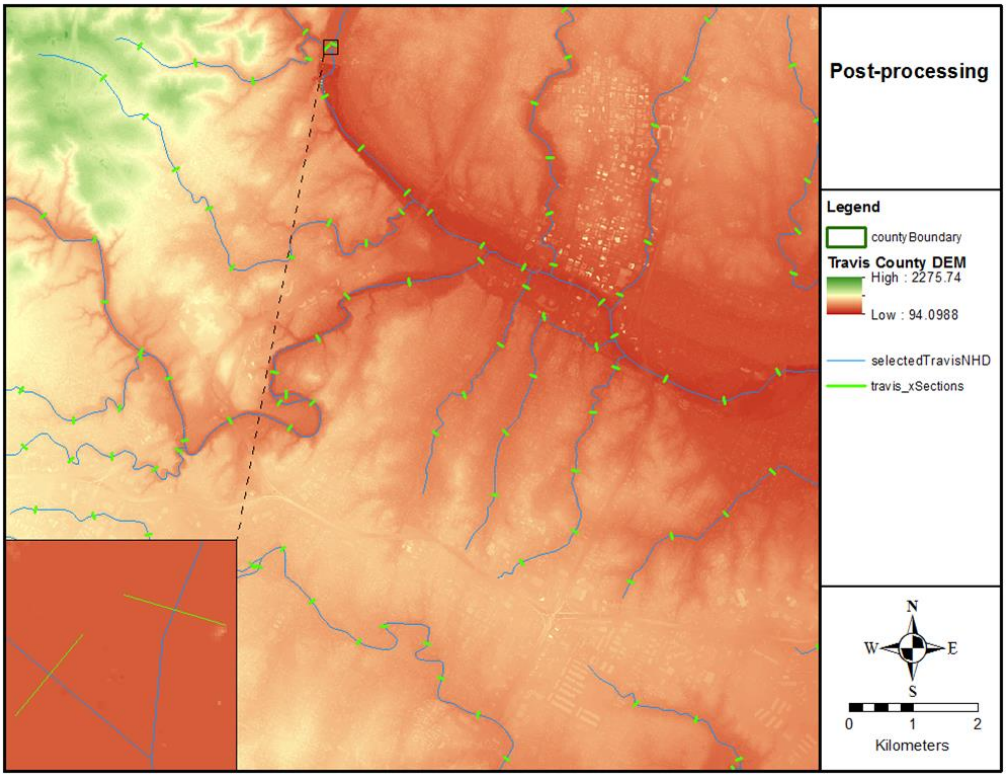

Figure 4.3 Post-processing results, cross section line feature class

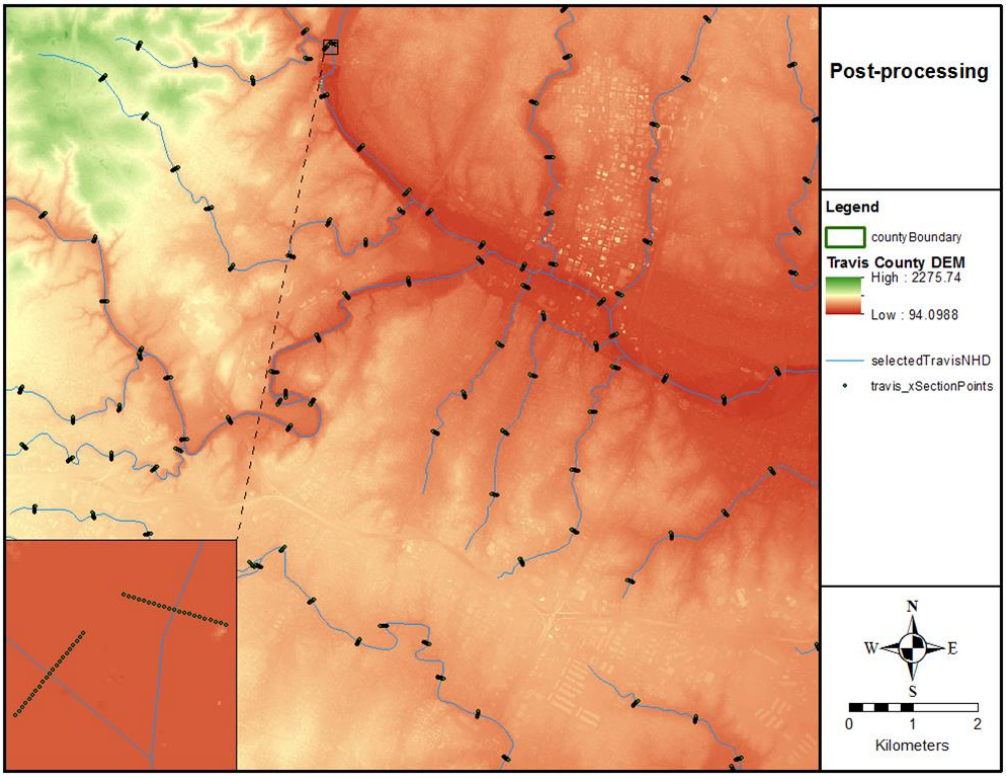

Figure 4.4 Post-processing results, cross section point feature class

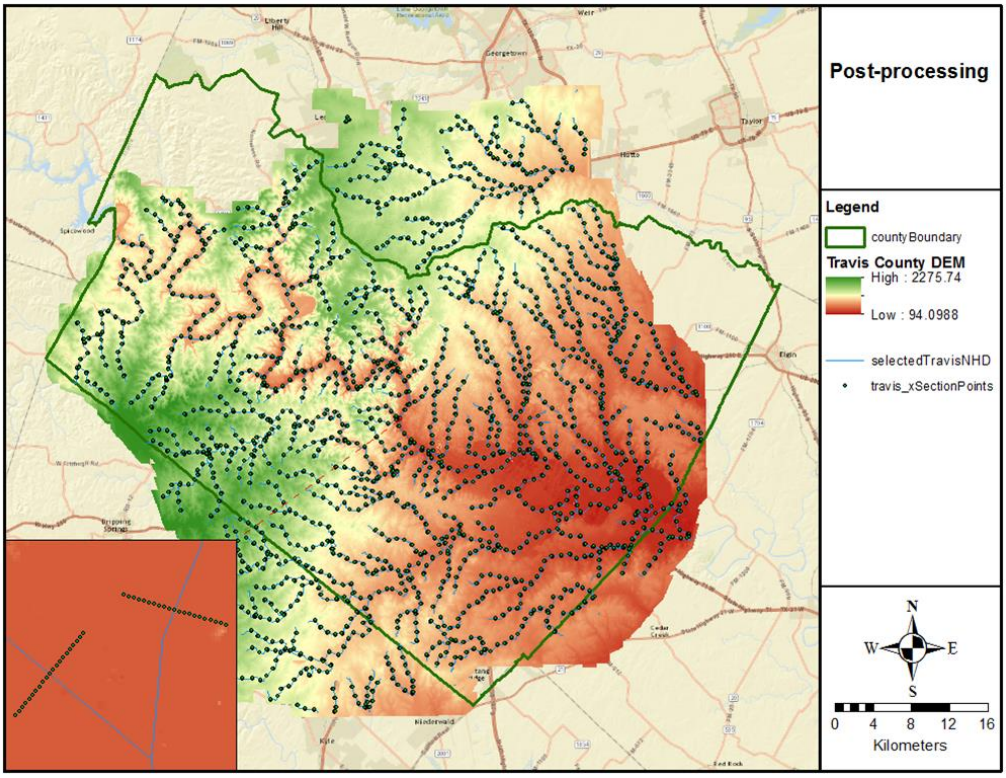

Figure 4.5 Complete cross section point feature class results for Travis County DEM

5 Discussion and recommendations

5.1 Current results and process structure

As a combined process and methodology, the suite of cross-section processing tools represented by the ArcGIS processing toolbox and the GeoNet code work well as a first attempt at automated cross-section feature extraction. The desired feature classes are made in fairly efficient time, and these features are located in the landscape in both practical Cartesian coordinates and technically applicable curvilinear coordinates of reach and cross-section station, all according to appropriate conventions. Furthermore, the processing inputs and outputs are few, easy to understand, and, as far as the inputs, readily available. Given that the code and the toolbox need significant clean up in the form of proper parameter labels, parameter descriptions, and tool descriptions, the objectives set forth are accomplished in this test scenario.

Previous versions of the pre, GeoNet, and post-processing steps have been tested on several datasets with varying cross-section intervals and widths across a range of DEM sizes and resolutions. At this point, the current suite version has only been tested with the Travis County dataset presented here.

The most glaring shortcoming and stumbling block is the current need for the processing external to ArcGIS, the modified GeoNet code. In itself, this produces two issues. In the current form, the code is still bound within the larger GeoNet processing context. That is, an extensive number of unused GeoNet parameters and processes are present in both the executable and run files making them cumbersome

and more confusing than need be. Because the two codes are heavily modified and rely on no proprietary GeoNet processes, they should be removed from this context entirely and combined into a single run file.

The extrication of the code from the GeoNet context and combination of the files into a single script would go a long way to facilitating a solution to the second major issue, that there is an external process at all. Having to break up the overall cross-section feature extraction into three steps is convenient for troubleshooting, but the requirement of additional software and operational knowledge is cumbersome and unnecessary. With a single script developed from the union of the executable and run files, the code can be rewritten in similar python code and brought into ArcGIS as an arcpy script. Were this to be accomplished, the entire cross-section feature extraction could be combined into one process with two inputs, the NHD flow lines and a DEM, and two outputs, the cross-section line and point feature classes.

5.2 Approach redefinition

It is worth considering this solution as a general experiment in the larger framework cross-section feature extraction. Many techniques utilized in this process may not work well when expanded to a more applicable scale. For example, accurate modeling around complex features such as bridges and culverts require many cross-sections in close proximity. This raises two particular concerns with regards to the cross-section extraction process proposed here.

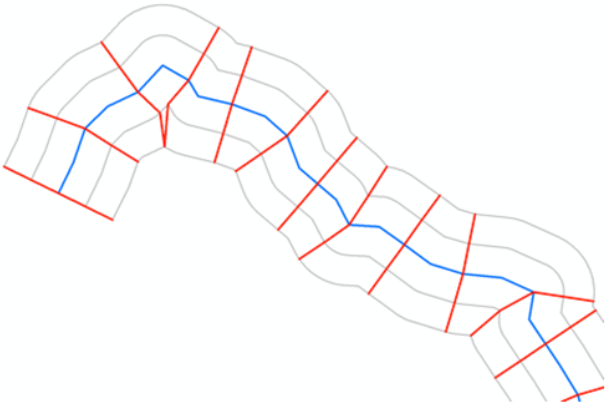

Figure 5.1 Cross section extraction approach developed by Chris Franklin and Brian Chastain of UT Dallas (Figure adapted from their slideshow)

The increased cross-section density required in proximity of complex features would, using this tool, increase the number of total cross-sections dramatically. As cross-sections are created with this tool set at a uniform interval, the required spacing at complex features would set the standard for the entire analyzed area. Because of this, the processing time and sheer size of the features classes would become overly cumbersome. A potential solution to this would be to incorporate complex features into the cross-section extraction process. The presence of a complex feature along a reach could change the reach cross-section interval or trigger a more dynamic cross-section extraction process that places extra cross-sections directly in proximity to those features.

The other concern involves a combination of the complex features and large floodplain areas. In such areas where the overbanks are wide and flat, the cross-sections may need to be much wider than the 100 m used in this example. However, when expanding the cross-section width when cross-sections are in close proximity increases the likelihood that cross-sections will overlap. This is one of the prohibitions of cross-section construction for modeling. A solution to this would be an approach such as that taken by Chris Franklin and Bryan Chastain of UT Dallas, Figure 5.1. Their cross-section extraction method uses

available ArcGIS tools to develop cross-sections that are not simply straight. By following their method, wide cross-sections can be developed without the cross-sections overlapping.

6 Conclusions

The initial solution to cross-section feature extraction represented by the suite of tools presented here works well. Given a DEM and river flow lines, both readily available online, cross-sections of user defined width at user defined intervals can be struck and addressed with curvilinear and Cartesian coordinates in relatively little time. However, there is much room for improvement. The process can be streamlined and unified within ArcGIS by rewriting the GeoNet derived code in python. Furthermore, were the process to be applied in real modeling situations, cross-sections would need to be densified around complex features such as bridges, and processing troubles may arise. A more dynamic cross-section striking solution is likely achievable. Regardless, this experiment shows that GIS processing of a DEM with NHD flow lines allows for a solution to some data gaps within the NFIE-River component.

7 Appendix A: Post-processing subroutine workflows

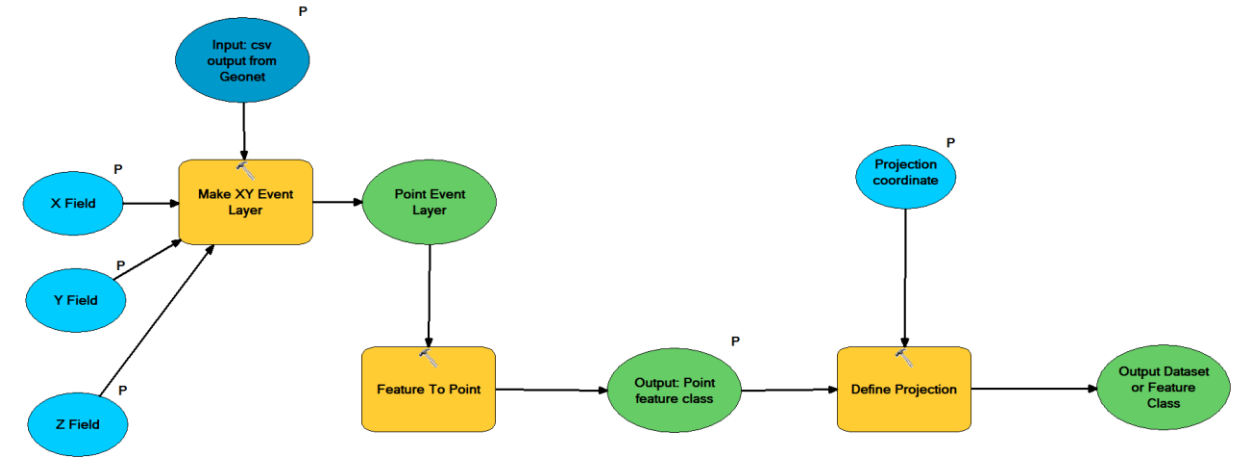

Figure 7.1 Subroutine 1 workflow: Converts the csv to an endpoint feature class

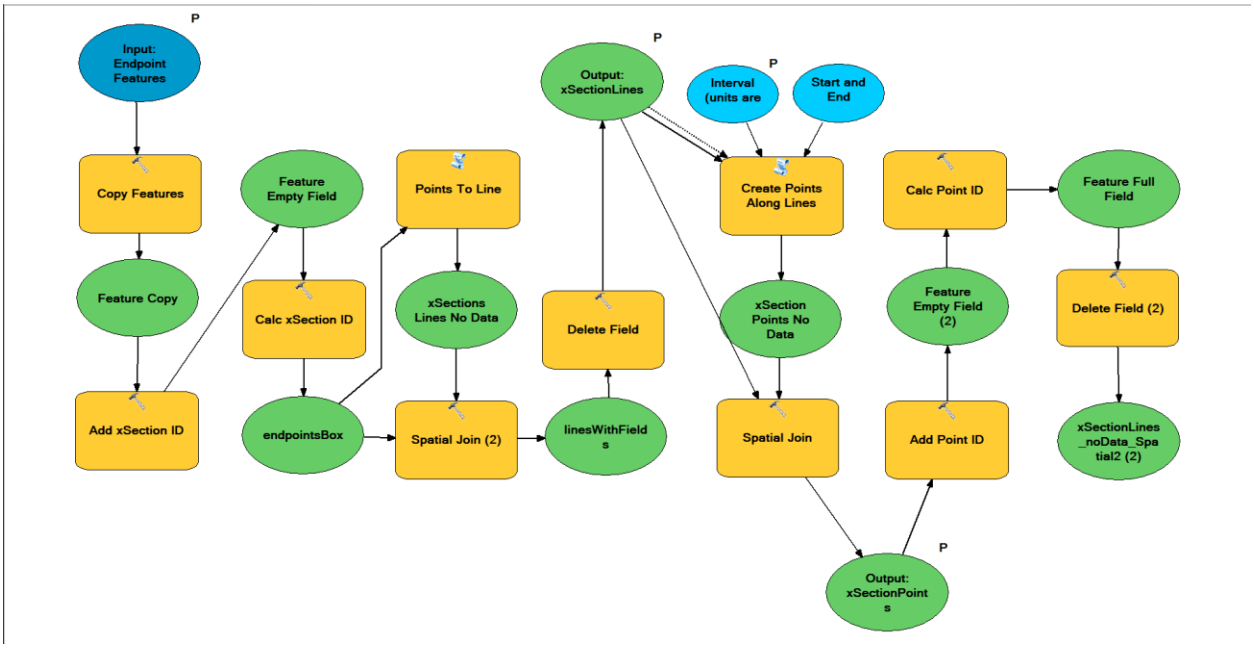

Figure 7.2 Subroutine 2 workflow: Converts the endpoints to cross section lines and points

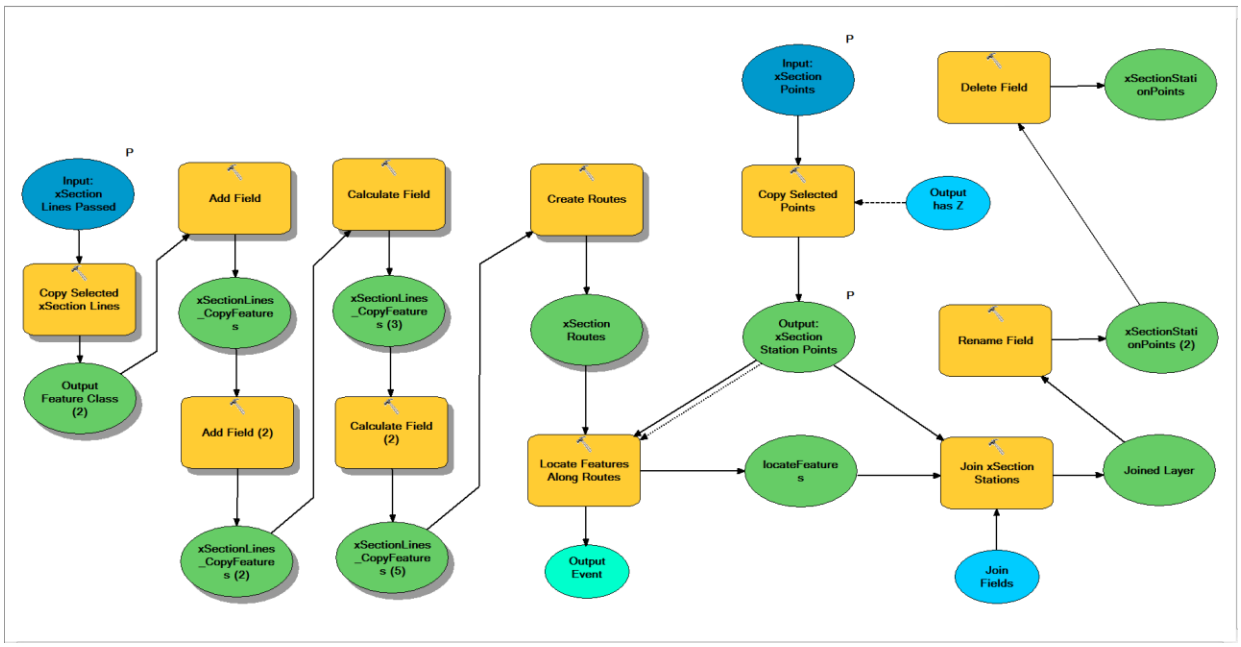

Figure 7.3 Subroutine 3 workflow: Determines the point stations along each cross section

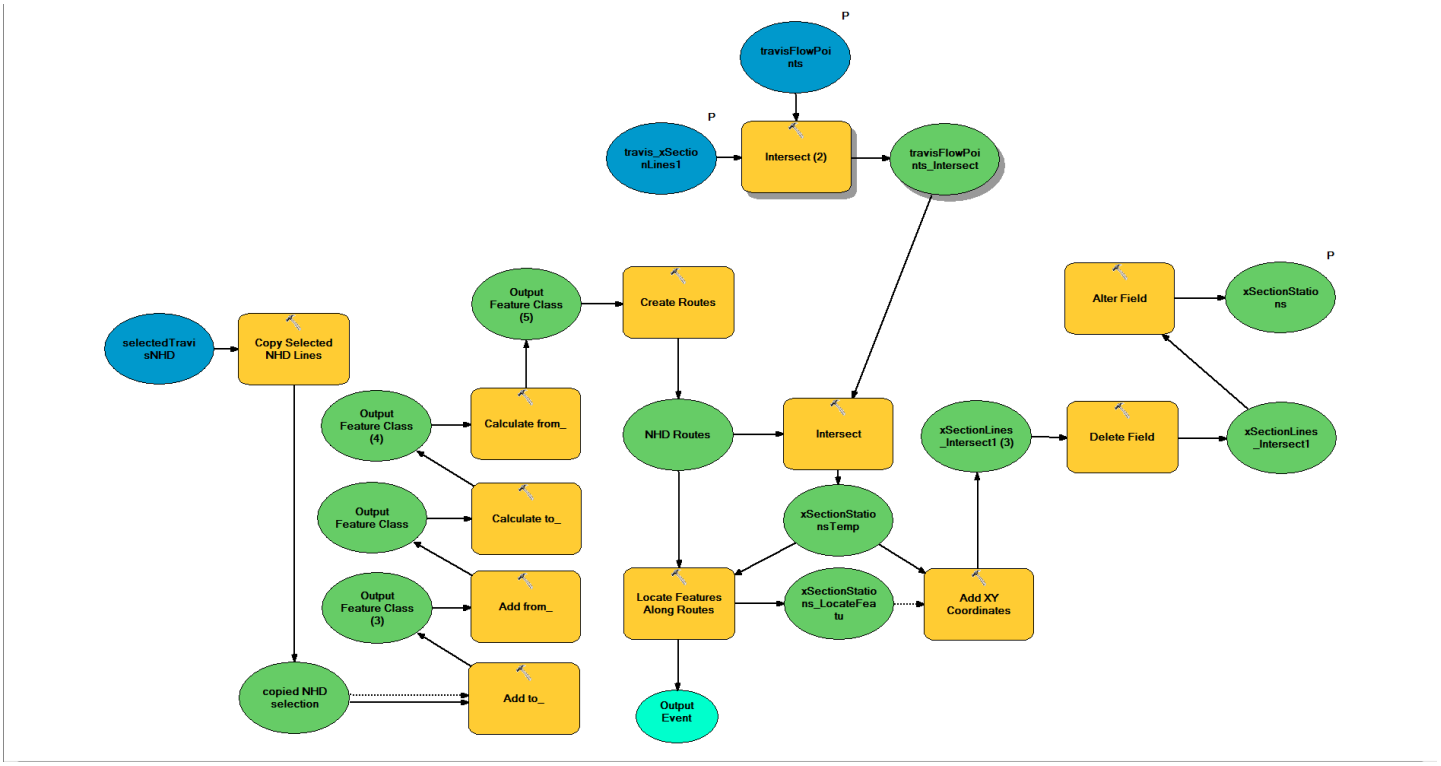

Figure 7.4 Subroutine 4 workflow: Determines the reach stations for each cross section

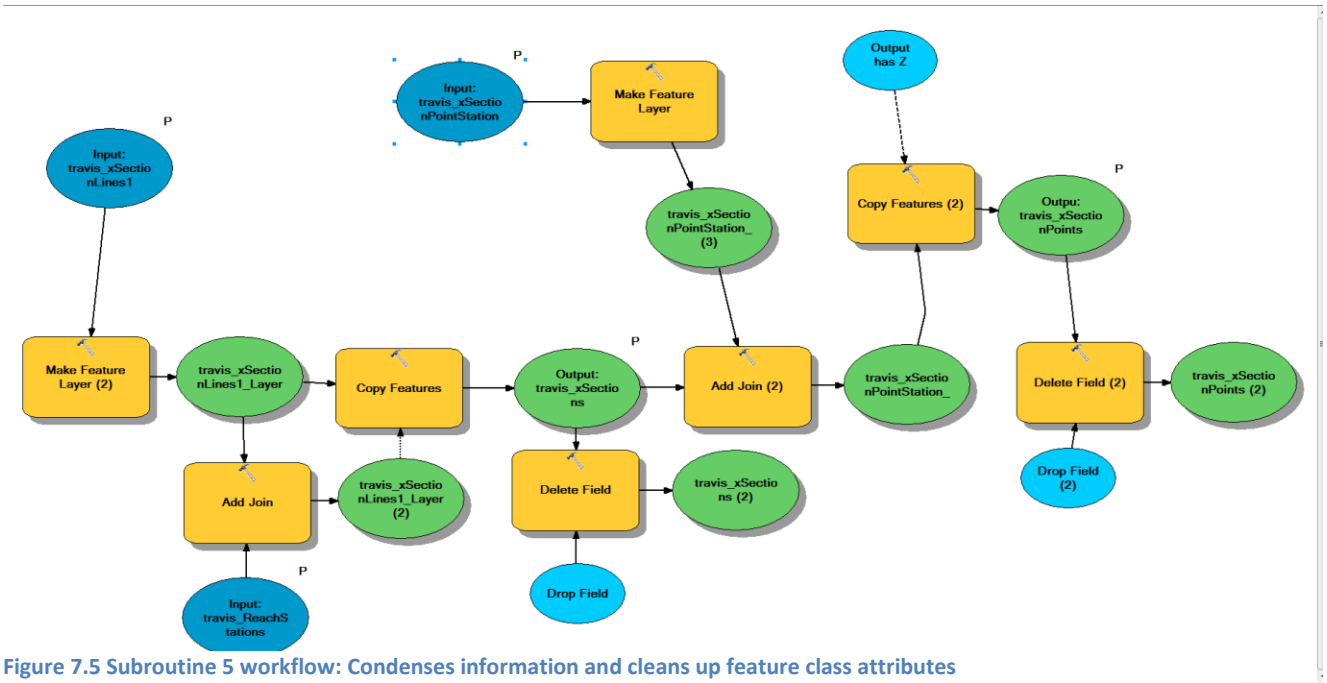

Figure 7.5 Subroutine 5 workflow: Condenses information and cleans up feature class attributes

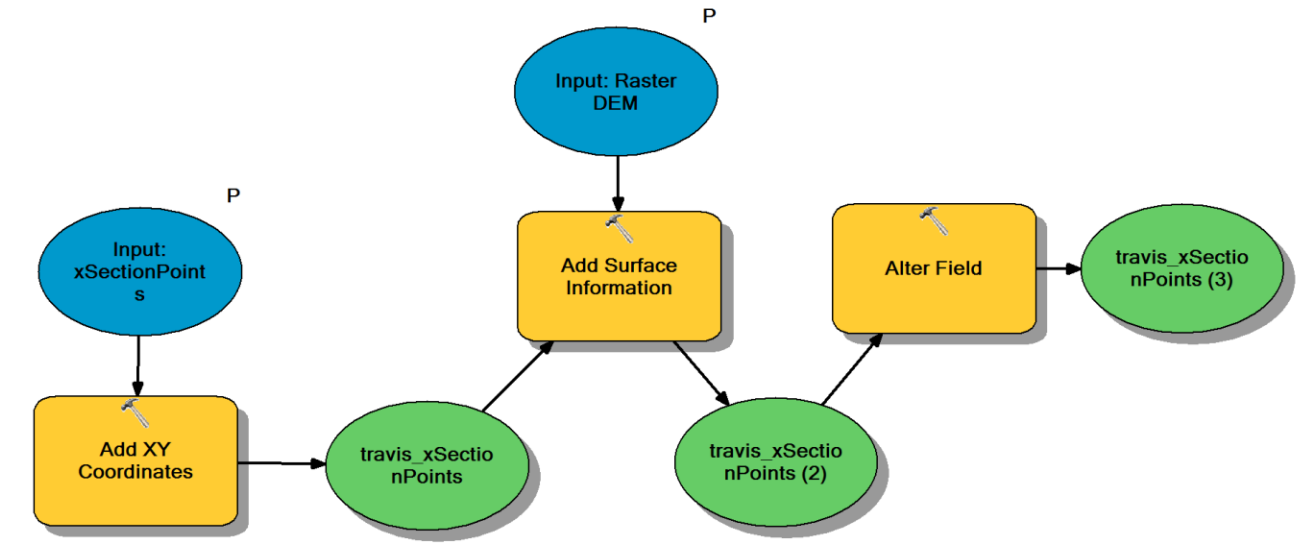

Figure 7.6 Subroutine 6 workflow: Adds Cartesian coordinates (x,y,z) to the point feature class