MGS Stacked Cross Sections: Convert Xsec to Traditional Display

# Summary

This script takes cross section data that was created with stacked display coordinates and converts it to true elevation (traditional) coordinates. All attributes from the original file get transferred to the converted file.

The script was written in Python 3 using the Spyder Python IDE. Comments throughout the code explain processes, and additional information is in this document. Throughout the code, some comments begin with “#%%”. The addition of the “%%” turns a chunk of code into a cell that can be run from the Spyder Python IDE. Sample input and output data is saved in the “SampleData” folder. This tool runs the most efficiently if all data is stored on a local drive.

# Code Structure

### #%% 1 Import Modules and define functions

This section imports necessary Python modules. Two print statement functions are defined (message and error), and a function is defined to check if a file exists.

### #%% 2 Set parameters to work in testing and compiled geoprocessing tool

This section defines parameters that are used in the script. The block within if (len(sys.argv) > 1): will execute if the script is run inside an ArcGIS Pro script tool. The else block is executed if the script is run outside of ArcGIS Pro in a Python editor. To run the script outside of Arc, modify the parameters in the else block. Do not modify the parameters in the if (len(sys.argv) > 1): block unless you are also modifying and updating the script tool. Parameter descriptions are in the code comments.

### #%% 3 Determine geometry of input feature class

This section determines the geometry of the input feature class (point, polyline, or polygon) and defines the shape variable. The county\_relief variable is also defined, which is used to calculate from the stacked display.

### #%% 4 Check for mn\_et\_id field

This section checks that the input feature class and the cross section line feature class both have the mn\_et\_id field. This field is necessary for calculating from the stacked display. If one of the feature classes does not have mn\_et\_id, it will join the field from the other one. If neither of them have the mn\_et\_id field, an error message will print and the tool will fail.

### #%% 5 Add unique ID field to input so join works correctly later

This section adds a unique\_id field to the input feature class. It is calculated based on the OBJECTID or FID field, and is used to join attributes at the end of processing.

### #%% 6 Create a blank output feature class

Creates a blank output feature class and adds fields: cross section ID (user input parameter), unique\_id, and mn\_et\_id.

### #%% 7 Convert point data

If the shape variable is point, this section executes. It uses SearchCursor to loop through cross section line file one line at a time. It defines et\_id and mn\_et\_id variables. It makes a list of x and y coordinates of each line vertex, and prints an error message if there are more than two vertices, or if the lines do not precisely follow UTM latitude. The minimum x value is defined in the min\_x variable, which is used in the coordinate calculation.

Then, SearchCursor is used to loop through the input feature class, using a where clause to query only the points along the current cross section. The variable new\_x\_raw is equal to the x coordinate in stacked display (the same as true x coordinate) minus the minimum x value of the cross section line. This ensures that the cross section display will start at zero on the left. This raw x value is divided by vertical exaggeration and converted to feet to create vertical exaggeration. The y coordinate is calculated by reversing the stacked coordinate display calculation. Finally, a geometry object is created from the calculated values, and a new point is added to the output feature class.

### #%% 8 Convert line data

If the shape variable is polyline, this section executes. It uses SearchCursor to loop through cross section line file one line at a time. It defines et\_id and mn\_et\_id variables. It makes a list of x and y coordinates of each line vertex, and prints an error message if there are more than two vertices, or if the lines do not precisely follow UTM latitude. The minimum x value is defined in the min\_x variable, which is used in the coordinate calculation.

Then, SearchCursor is used to loop through the input feature class, using a where clause to query only the lines along the current cross section. The code reads the xy coordinates of the line vertices. The variable new\_x\_raw is equal to the x coordinate in stacked display (the same as true x coordinate) minus the minimum x value of the cross section line. This ensures that the cross section display will start at zero on the left. This raw x value is divided by vertical exaggeration and converted to feet to create vertical exaggeration. The y coordinate is calculated by reversing the stacked coordinate display calculation. Finally, a geometry object is created from the calculated values, and a new line is added to the output feature class.

### #%% 9 Convert polygon data

If the shape variable is polygon, this section executes. It uses SearchCursor to loop through cross section line file one line at a time. It defines et\_id and mn\_et\_id variables. It makes a list of x and y coordinates of each line vertex, and prints an error message if there are more than two vertices, or if the lines do not precisely follow UTM latitude. The minimum x value is defined in the min\_x variable, which is used in the coordinate calculation.

Then, SearchCursor is used to loop through the input feature class, using a where clause to query only the lines along the current cross section. The code reads the xy coordinates of the polygon vertices. The variable new\_x\_raw is equal to the x coordinate in stacked display (the same as true x coordinate) minus the minimum x value of the cross section line. This ensures that the cross section display will start at zero on the left. This raw x value is divided by vertical exaggeration and converted to feet to create vertical exaggeration. The y coordinate is calculated by reversing the stacked coordinate display calculation. Finally, a geometry object is created from the calculated values, and a new polygon is added to the output feature class.

### #%% 10 Join input fc fields to output

This section uses the unique\_id field to join attribute fields to the output. It lists fields in the input file, and then removes unnecessary or redundant fields from the list before joining to the output.

### #%% 11 Delete join fields from input and output

This section deletes the unique\_id field from the input and outputs.

### #%% 12 Record and print tool end time

Finally, the script calculates processing time and prints the elapsed time.