

## Optimizing and smoothing contour curves for automated cutting using Rhino

Contour curve data can be of highly variable quality depending on the original source. Much of the time the raw data is not suitable for automated cutting, as it can be locally rough (jagged), contain too few or too many points, or have other problems.

Some software have built-in smoothing and fixing routines that may work, but the results should always be examined carefully. Rhino has native functions such as Smooth, Rebuild and FitCrv for fixing individual curves, but as a contour curve file can contain hundreds of curves that have different sizes and numbers of points, it may be difficult to find the correct settings for optimizing a whole file at once.

The following tutorial covers a script for optimizing contour curve data for automated cutting developed by Mitch Heynick at the EPFL. It is written in Python and will run in Rhino V5 or V6 for Windows or Mac. If Rhino is in French, the script prompts will be as well, otherwise it defaults to English.

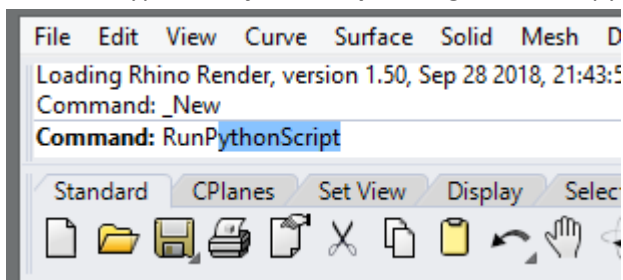
The script works on curves in 2D or 3D. The input curves can be polylines, polycurves or splines. It will ignore single lines as well as pure circles, arcs and ellipses. It has options to reduce points within tolerance, smooth the curves or not (various degrees) and output either splines or polylines.

All curves are converted to polylines for processing. The polyline reduction algorithm can reduce excess points but does not alter the curve more than the tolerance specified. Activating the smoothing part will change the curve more (according to the degree chosen); spline output will rebuild the result into spline curves – much smoother, but with the possibility of overlaps and more radical local changes.

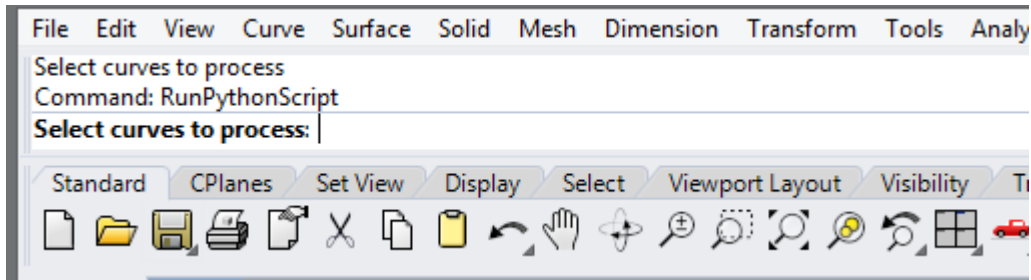
Curve characteristics (layer, color, etc.) are preserved when running the script. If your contour line curves are already joined to the outer rectangular borders (as in post-treatment of a typical cutting file), then smoothing (see below) must be set to 0 – otherwise all the outer rectangles will move significantly and the file will be unusable. It is therefore recommended to apply point reduction and smoothing to the overall contour line file (all curves, no borders) **before** creating your individual cutting files.

### Using the script:

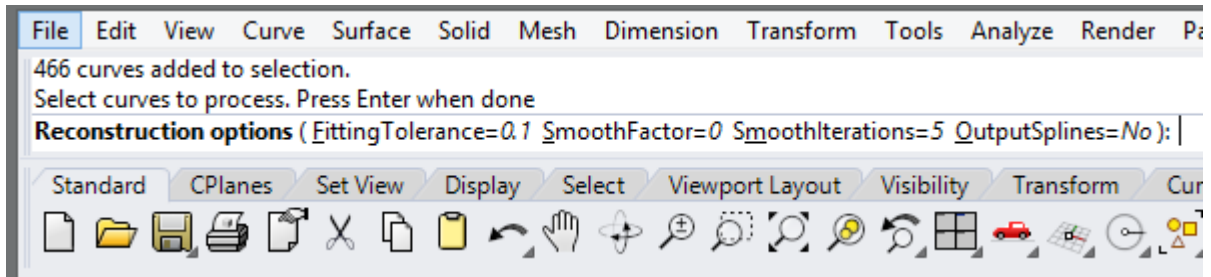
1. Your file should already be in **mm** and at model scale (i.e. correct size for cutting). If this is not the case, it will be difficult to determine the tolerance factor for the reconstruction.
2. Copy the script file « **CombinedFixContourCrvsFE.py** » to your desktop.
3. In Rhino, type **RunPythonScript**, navigate to the .py file on the desktop and hit « Open ».



4. The script will run. First it will ask you to select a curve collection to process.



5. Once the curves are selected, you will be offered four options on the command line :



- a. **FittingTolerance:** This is the maximum distance (in file units/mm) that curves are to move during point reduction. A value of 0.1 to 0.2 will usually not even be visually noticeable in the cut model, but can still make a large reduction in the point count.  
*Note: this tolerance will only be held if the SmoothFactor=0 (no smoothing). Using a smoothing factor of more than 0 (next option) will likely cause the curves to change shape by more than the FittingTolerance value.*
- b. **SmoothFactor:** A value between 0 and 1. The default setting is 0, which means no smoothing is applied, just point reduction. If this is set larger than 0, a degree of smoothing will be applied. 1.0 is the maximum value. Smoothed curves become less jagged locally, while the overall shape of the curve is preserved. You can choose to apply smoothing one or more times (iterations), next option.
- c. **SmoothIterations:** This is the number of times a smoothing operation is applied to each curve. The default is 5. Generally a larger number of operations at a smaller smooth factor results in more local smoothing and less global smoothing than a larger smooth factor and fewer iterations, but you will need to experiment.
- d. **OutputSplines:** As stated above, all curves are first converted into polylines for point reduction. With the setting No (default), the output stays as polylines. When set to Yes, the output is converted to splines afterward. This will result in very smooth curves, but with an increased possibility for loops and overlaps with neighboring curves.

6. Once the options are set, press Enter to run the script. In Windows Rhino, you will see a progress meter as the script calculates.

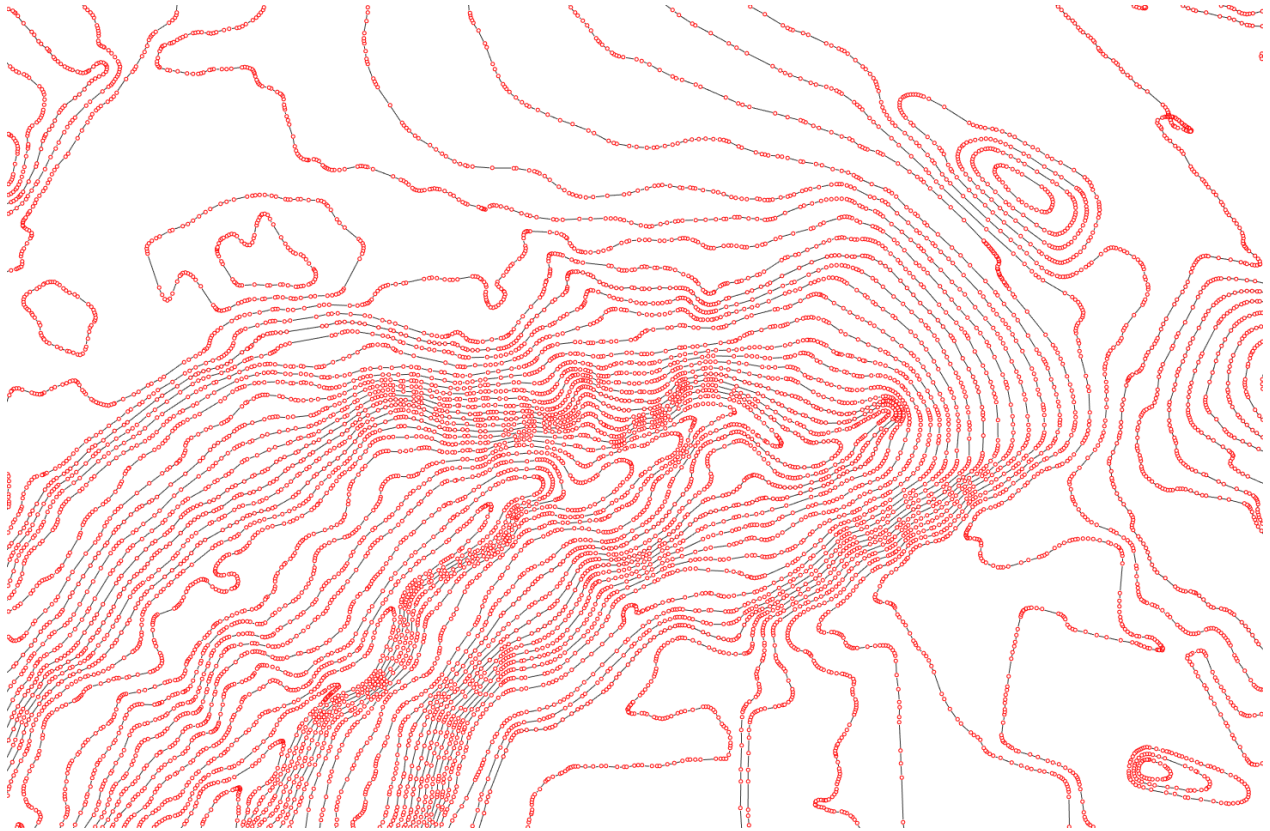
Once done, check areas with tight curvature and dense contour line spacing carefully for overlaps or loops after running the operation. Excessive smoothing can also cause your model to look too « soft » by removing some of the essential terrain features. The script is relatively rapid, so if the results are not as desired, Undo and re-run the script with a different set of parameters.

The script changes the original curves – if you need the originals, copy them to another layer and then run the script on the copies. This will also allow you to see the difference between the originals and the rebuilt curves.

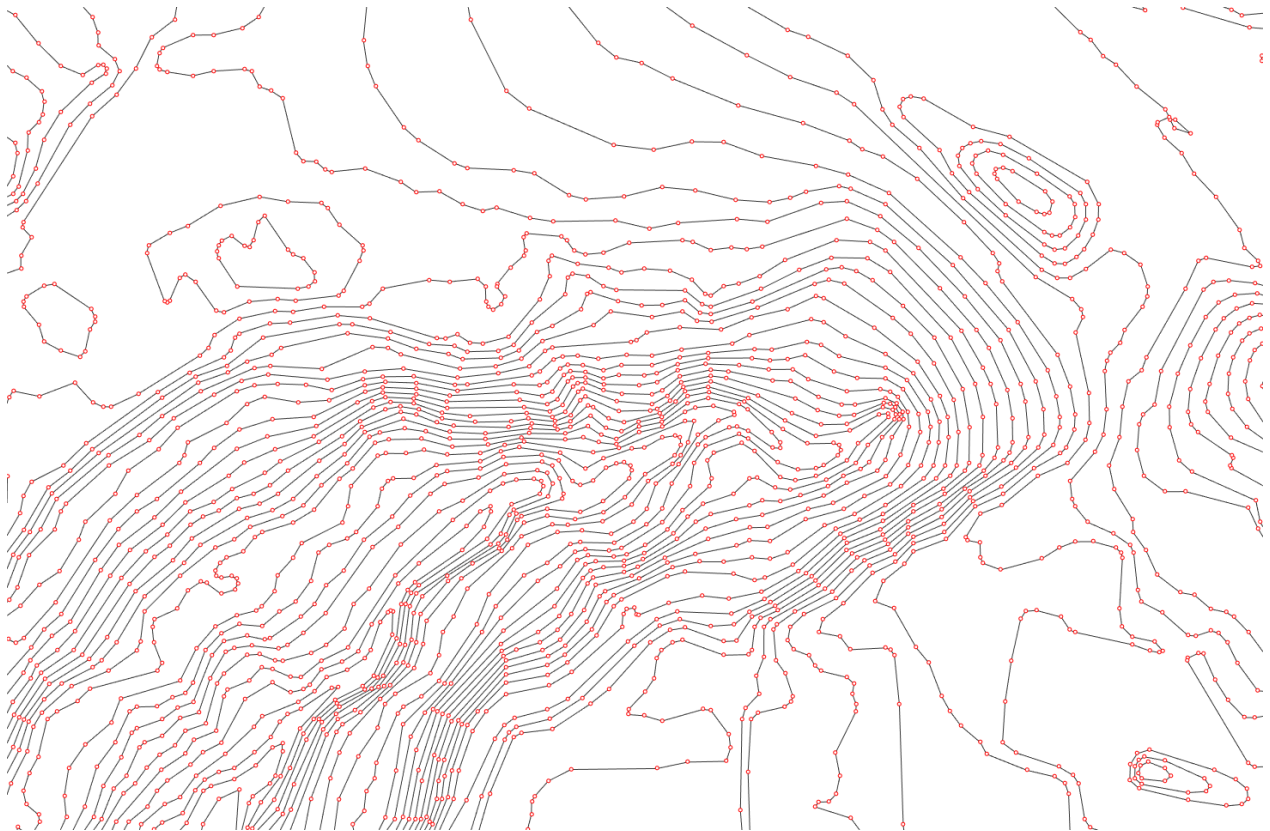
Original file 1200mm x 800mm – 1160 curves; 429,850 points



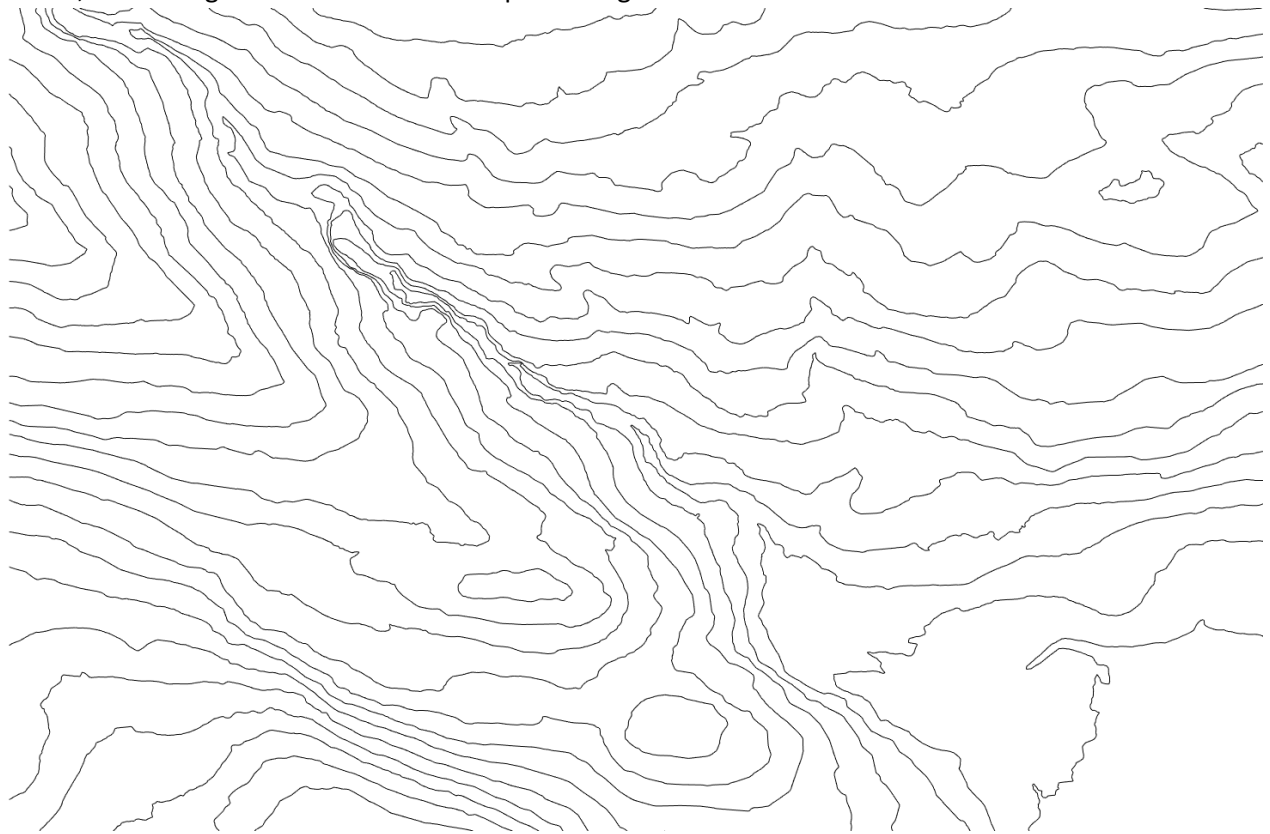
Detail: curve control points before reduction (429,850 points)



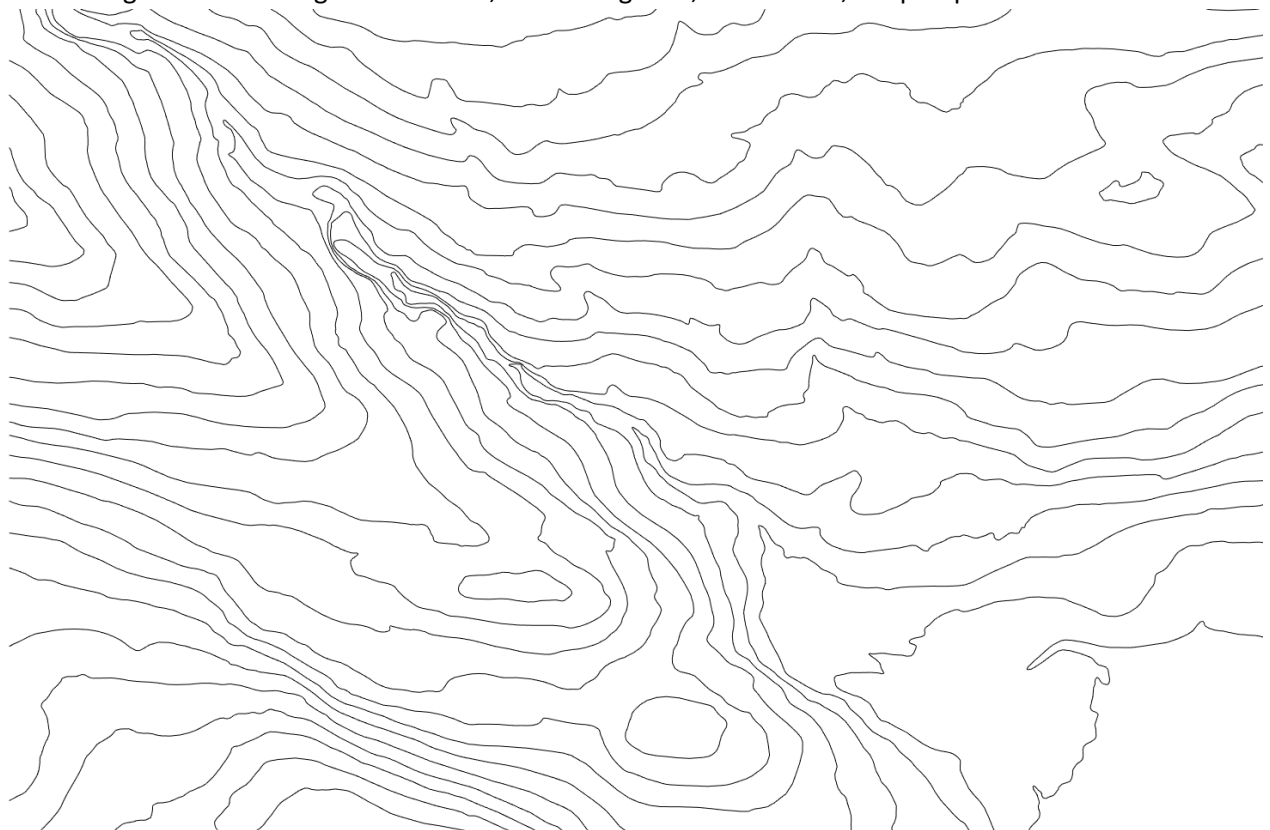
Curve control points after reduction: FittingTolerance 0.2, 0 smoothing – 86750 points (80% reduction)



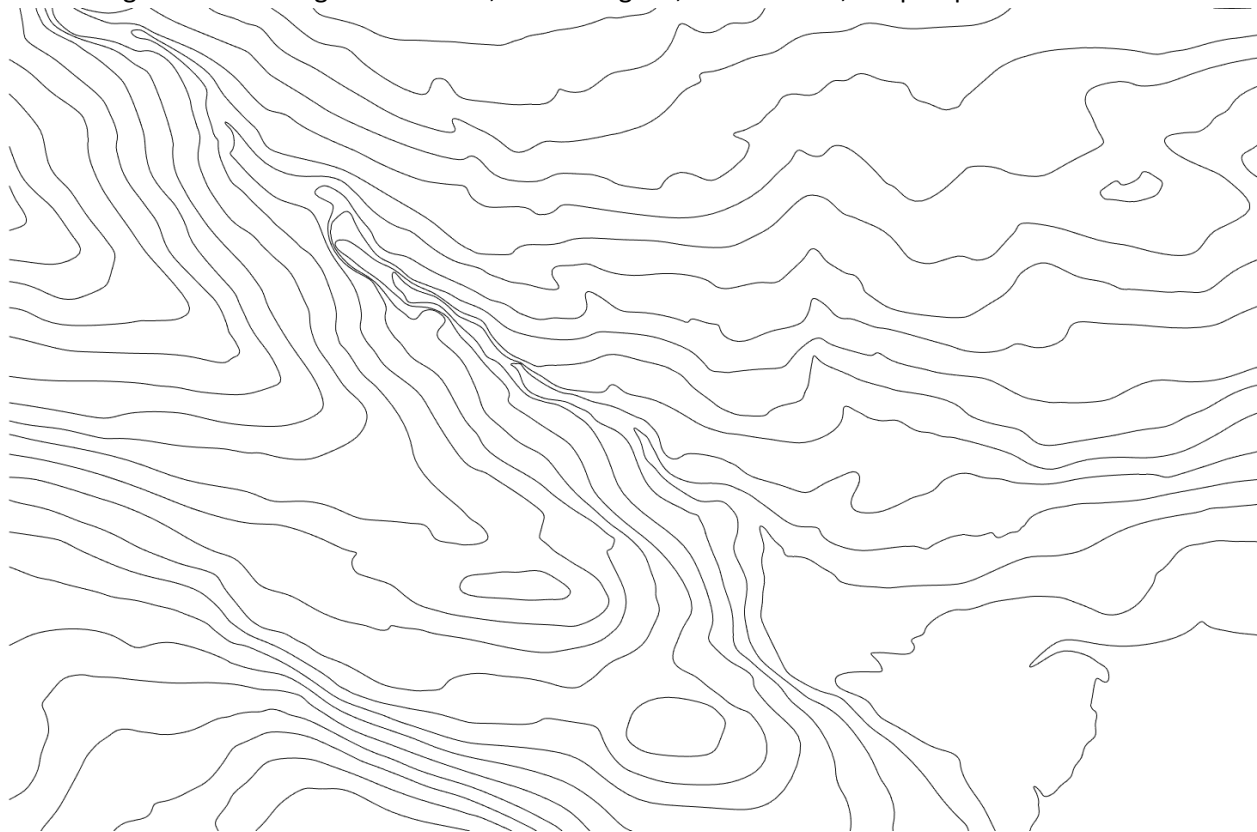
Detail, smoothing effects – curves before processing



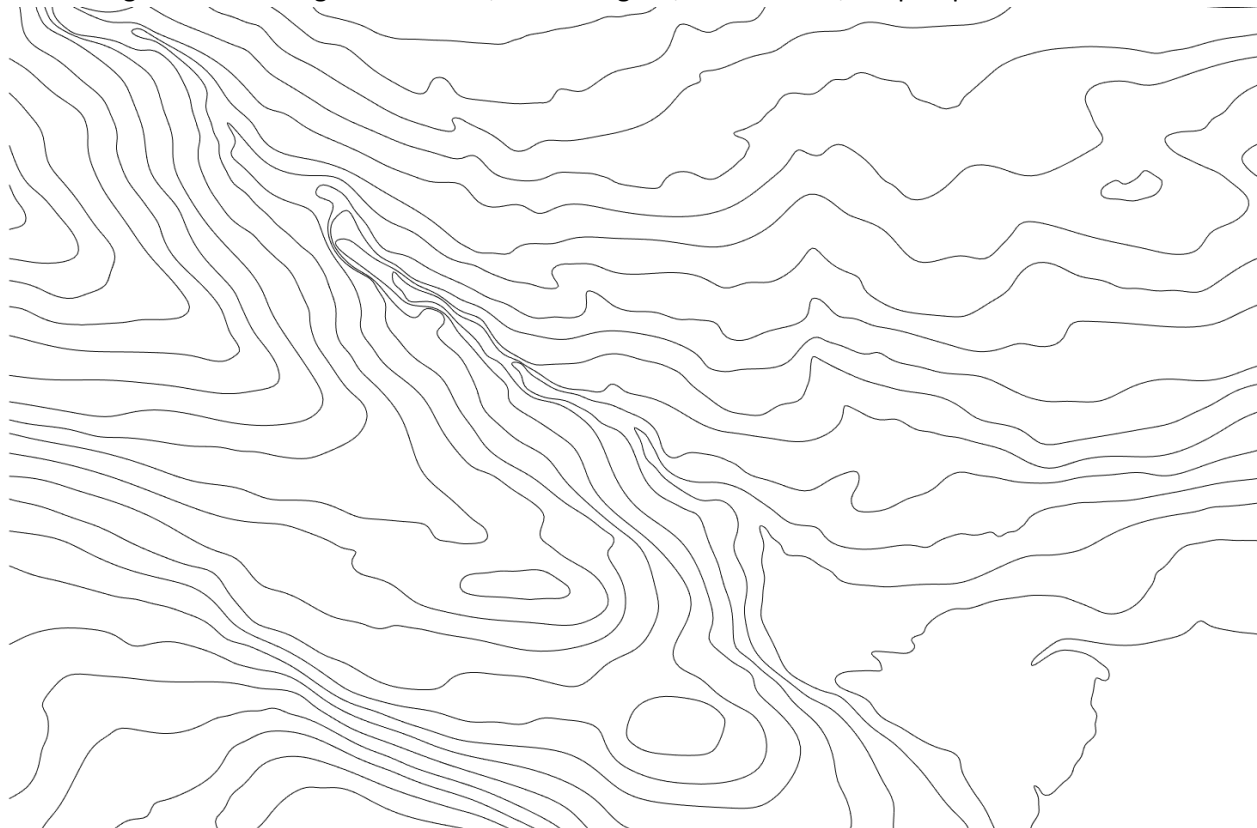
Smoothing effects – FittingTolerance 0.2, Smoothing 0.25, 5 iterations, OutputSplines=No



Smoothing effects – FittingTolerance 0.2, Smoothing 0.5, 10 iterations, OutputSplines=No



Smoothing effects – FittingTolerance 0.2, smoothing 0.5, 10 iterations, OutputSplines=Yes





Smoothing effects – FittingTolerance 1.0, smoothing 1.0, 10 iterations, OutputSplines=Yes



Smoothing effects - overlaps caused by excessive smoothing

