TATOO – exemplary applications of the workflows

Intention of this document

This document was created to explain the presented workflows for the LARSIM model applying them to a real-case catchment. It shall neither be a full technical documentation of the functions (see the function code comments) nor a description of the functionalities (see publication Mitterer and Disse 2021). Here, I only explain the functionalities which are useful for LARSIM, what requires pre-knowledge about the model (see www.larsim.info). The document is split into four sections: (1) *Preparations and overview* for general issues, (2) *Raster model creation*, (3) *Subcatchment model creation*, and (4) *Hydrological response units*. Section 3 builds for a large share on section 2, as many functions for both model structures are similar or even the same (functions in the common package). The following descriptions explain the functionalities step-by-step using a test data set of an 8-km² catchment in Eastern Bavaria in Germany. The data was very kindly provided by the Bavarian Agency for Digitisation, High-Speed Internet and Surveying, and the Bavarian Environmental Agency. ***The data is only allowed to be used for testing the functionalities of this software.*** Please refer to the license documents for further details.

1 Preparations and overview

1.1 Installing TATOO

TATOO does not require any installation except the availability of the ArcPy package. Nevertheless the path of the packages has to be added to the Python environment you are working in (e.g., Spyder). The following line 39 (same for both example workflows) has to be modified according to your system replacing PATH with the path pointing to the folder with the TATOO packages:

```
sys.path.append((r'PATH')) # set path for packages
```

1.2 Set up of input and output data and paths

For both model designs, raster- and subcatchment-based there are four sources of input data you have to provide:

- high-resolution digital elevation model (DEM) raster
- flow network (FNW) polyline feature class
- land use polygon feature class
- soil parameter feature class

Store all data inside an ESRI file geodatabase, and set the input and output data paths in the control section at the top of the workflow (see the workflow descriptions for details).

1.3 Easy-to-access parameters in the workflows

Table 1 summarizes all possible easy-to-access parameters, which are used in the workflows. The input parameters are explained in detail during the descriptions of the functions.

Parameter	Description	Default	Unit
parameters for high-resolution elevation model processing			
burning_h	deepening value at flow network (float, ≥ 0)	5	m
model_cellsize	model cell size (integer > 0)	200	m
def_sc_area	optimal model elements' area	10^6	m²
def_fl_min_tol	tolerance for flow length mismatch for catchment optimization	50	m
def_a_min_tol	minimum subcatchment area tolerated	50	m²
parameters for runoff concentration processing			
def_sl_min	minimum slope to be retained (float, ≥ 0.0001)	0.0001	dZ/dL

Parameter	Description	Default	Unit
def_fl_strct_mism	default flow length for structural mismatch and negative transition deviations (float, > 1)	2	m
def_fl_upper_lim	upper threshold for realistic flow length (float, > 0)	300	m
def_sl_excl_quant	upper threshold for realistic slope as quantile (float, 0 – 1)	0.999	-
def_zmin_rout_fac	HUT for cells with stream as 50% of stream's low and high point (float, 0 – 1)	0.5	-
def_zmax_fac	High point of runoff concentration within model element elevation distribution with 1 = highest value as defined by Kirpich 1940 (float, 0 – 1)	1	-
def_fl_strct_mism	default flow length for structural mismatch and negative transition deviations (float, > 1)	2	m
parameters for geomorphology			
hq_ch	HQ2 (or MHQ summer) (float, > 0)	-	m^3/s
hq_ch_a	Gauge catchment area (float, > 0)	-	km^2
ser_q_in_corr	ascending order of inflow cell IDs [-] (index) and HQ2 (or MHQ) of inflow cells [m^3/s] (series), pd.Series(np.array([float > 0]), index=[integers > 0], name='q_in')	empty pd.Series	-
parameters for channel estimation			
ch_est_method	string defining channel estimation function, select from 'Allen', 'Krauter' or 'combined'	'combined'	-
def_bx	foreland edge default width (float, ≥ 0)	0	m
def_bbx_fac	foreland default factor with $bbl = bm \cdot def\_bbx\_fac$ and $bbr = bm \cdot def\_bbx\_fac$ (float, > 0)	1	-
def_bnm	channel slope (float, > 0)	1.5 (= 67 %)	dL/dZ
def_bnx	foreland slope (float, > 0)	100 (= 1 %)	dL/dZ
def_bnvrx	foreland boarder slope (float, > 0)	4 (= 25 %)	dL/dZ
def_skm	roughness value river channel (0 – 100)	30 (natural)	$m^{1/3}/s$
def_skx	roughness value foreland (0 – 100)	20 (uneven vegetated)	$m^{1/3}/s$
parameters for hydrological response unit handling			
ctrl_opt_infdyn	control operator to activate dynamic infiltration routine parametrization (False: not active, True: active)	True	-
ctrl_opt_impperc	control operator to activate sealing parametrization in utgb.dat (False: not active, True: active)	True	-
ctrl_opt_capr	control operator to activate capillary rise parametrization in utgb.dat (False: not active, True: active)	False	-
ctrl_opt_siltup	control operator to activate silting-up parametrization in utgb.dat (False: not active, True: active)	True	-
ctrl_hru_aggr	control operator to activate HRU aggregation (False: not active, True: active)	True	-
def_amin_utgb_del	area threshold below which HRUs are deleted (float, ≥ 0)	10^{-8}	m^2
input field names of GIS inputs for HRU calculation			
f_lu_id	landuse class ID numbers in path_lu (string)	'landuse_id'	-
f_impperc	user-defined impervious percent value in path_lu (string)	'imp_perc'	-
arr_lu_mp	assignment table for land use classes and parameters, np.array with the following order: [landuse_id, landuse_name, MPla, MPdi], example: np.array([[1, 'settlement', 0, 0], [2, 'space in settlement', 0, 0], [3, 'impervious land', 0, 0], [4, 'cropland', 75, 30], [5, 'vineyard', 75, 50], [6, 'fruit growing', 100, 50], [7, 'fallow land', 100, 80], [8, 'unvegetated soil', 75, 30], [9, 'intensive pasture', 100, 80], [10, 'wetland', 100, 30], [11, 'extensive pasture', 100, 80], [12, 'grassland with trees', 125, 65], [13, 'conifer forest', 150, 30], [14, 'broadleaf forest', 150, 50], [15, 'mixed forest', 150, 50],	-	-

Parameter	Description	Default	Unit
	[16, 'water', 0, 0]])		
specific land use class IDs			
lu_id_imp	impervious land use class (LARSIM: 3)	3	-
lu_id_water	water land use class (LARSIM: 6)	16	-
cross section definition			
ctrl_rout_cs_file	(de-)activate creating cross section profile parameter file (False: not active, True: active)	True	-
ctrl_rout_cs_fit	(de-)activate fitting cross section profiles (False: not active, True: active)	True	-
parameters for cross section identification			
def_cs_wmax_eval	length of transects incl. both sides of river (float, ≥ 0)	600	m
def_cs_dist	distance between transects (float, > 0)	200	m
def_river_a_in	minimum catchment area defining a river (float, ≥ 0)	1	km²
def_cs_search_rad	search radius for transect identification (float, ≥ 0)	5	m
def_cs_dist_eval	linear distance between two automatically derived cross sections	50	m
parameters for detailled cross section determination			
def_cs_hmax_eval	maximum height of cross section evaluation (float, ≥ 0)	10	m
def_lam_hres	vertical spacing between evaluation lamellae (float, > 0)	0.1	m
def_ch_wmax_eval	maximum width of channel evaluation (float, ≥ 0.1)	40	m
def_ch_hmin_eval	minimum height of channel evaluation (float, > 0)	0.1	m
def_ch_hmax_eval	maximum height of channel evaluation (float, > 0)	3.0	m
def_ch_hmin	minimum channel depth threshold for channel identification (float, ≥ 0.2)	0.2	m
def_ch_vmin	minimum reasonable flow velocity (float, ≥ 0)	0.5	m/s
def_ch_vmax	maximum reasonable flow velocity (float, > def_ch_vmin)	3.0	m/s
def_chbank_slmin	minimum channel bank slope threshold for channel identification (float, > 0)	0.1	dZ/dL
def_ch_w	artificial channel width, added to cross sections (float, ≥ 0)	0.5	m
def_ch_h	artificial channel depth, added to cross sections (float, ≥ 0)	0.5	m
def_ch_wres	horizontal resolution of interpolated points in channel (float, > 0)	0.05	m
def_cs_intp_buf_dist	cross section intersection points' buffer distance (float, ≥ 0)	1	m
def_val_wres	horizontal resolution of interpolated points within valley	10	m
def_flpl_wres	horizontal resolution of interpolated points within flood plain	0.2	m
def_flpl_wmax_eval	estimated maximum flood plain width	50	m
plot outputs			
ctrl_show_plots	Show plot figure windows (False: no plots, True: show plots)	False	-
ctrl_save_plots	Save plots on hard drive (False: do not save, True: save plots)	True	-
parameters for model files (print in header of file)			
catch_name	catchment name (one line string)	-	-
src_geodata	source of used geoinformation (one line string)	-	-
parameters for model element parameter file tgb.dat			
tgbdat_comment	Comment string (one line string)	-	-
hcs_epsg	horizontal coordinate system EPSG number, select from: • DHDN / GK Zones 2-5 (31466-69) • CH1903 (21781) • WGS-84 geogr. 2D (4326) • ETRS89 Zone 32N/33N (25832/33) • Austria GK West/M28 (31254/57)	25833	-
vcs_unit	vertical coordinate system units, one line string, only 'm ue. NN' allowed	'm ue. NN'	-
def_tgb_nodata_val	no data value, max. 3 characters, LARSIM: -1	-1	-
parameters for model element parameter file utgb.dat			
utgbdat_comment	Comment string (one line string)	-	-
def_utgb_nodata_val	no data value, max. 3 characters, LARSIM: -1	-1	-
parameters for model element parameter file profile.dat			
profdat_comment	Comment string (one line string)	-	-
def_profdat_decmax	decimal numbers allowed (integer, ≥ 0)	2	-
def_profdat_nodata_val	no data value, max. 3 characters, LARSIM: -1	-1	-
def_profdat_exit_val	value terminating cross section data block (LARSIM: 999999)	999999	-
command window outputs			
print_out	command line output (False: no output, True: output)	True	-

Table 1: List of parameters for a raster model setup with TATOO (green font color: only for raster model workflow, blue font color: only for subcatchment model workflow).

2 How to create a raster model using TATOO

2.1 Workflow overview

The following section describes the workflow *tatoo_raster_example.py* to create a LARSIM raster model with TATOO. The workflow to create raster models consists of 15 functions including four potential or necessary user interventions (see workflow in Figure 1).

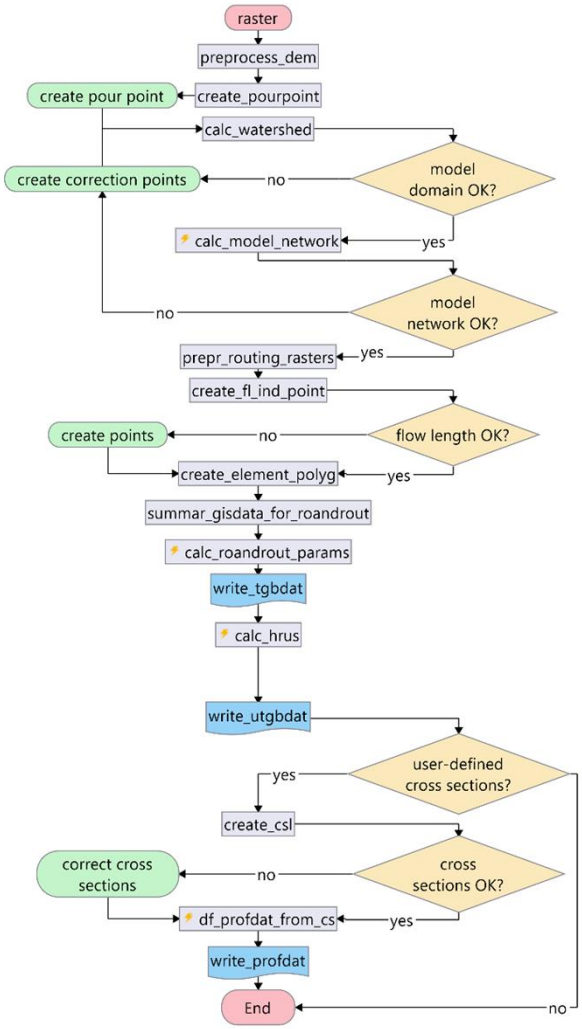

Figure 1: Workflow to create a LARSIM raster model with TATOO

2.2 Set paths and names for input and output data

The following variables have to be set at the beginning defining the inputs and outputs:

- geodatabase (GDB) where inputs are stored (`path_gdb_in`)
- the respective input layer paths of DEM, FNW, land use, and soil (`path_dem_hr, path_fnw, path_lu, path_soil`)
- geodatabase where outputs are stored (`path_gdb_out`)
- output path for model files (`path_files_out`)
- output path for figure files, if storage is activated (`path_plots_out`)

ATTENTION: Keep the '\\' in the end of the path variables!

All other layer and file names can be left as default but can be changed if required.

2.3 Preprocess DEM (*preprocess_dem*)

The calculation starts with the function ***preprocess_dem***. It aggregates the high-resolution DEM (DEM-HR) to an elevation grid with the desired model resolution (DEM-MR) using the parameter `model_cellsize`. Then it converts the FNW, burns it into the DEM-HR using the burning depth `burning_h`, and fills it afterwards. The result is shown in Figure 2 indicating that the detailled flow network is not well represented by the chosen model resolution of 200 m.

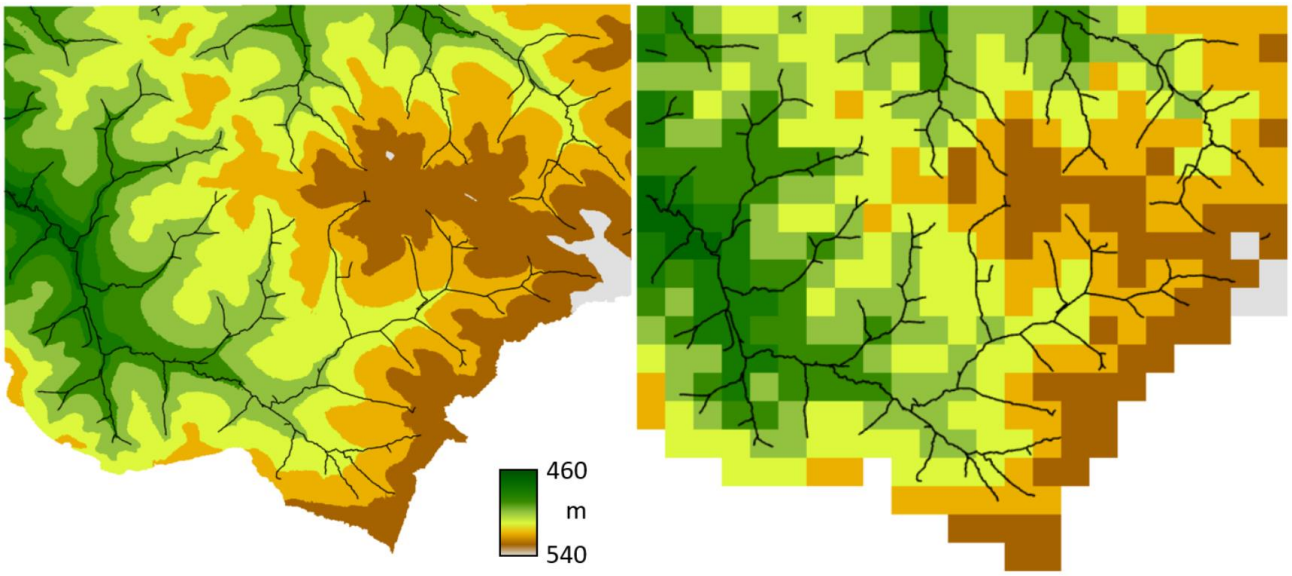

Figure 2: Original DEM-HR and resulting DEM-MR (source: LDBV).

2.4 Create pour point and calculate and correct watershed (*create_pourpoint* and *calc_watershed*)

After preprocessing the DEM, the user may create a pour point feature class identifying the desired model domain using the function ***create_pourpoint***. Creating feature points within that feature class (e.g., using the software ArcGIS Pro) will enable the user to add ID numbers to the subcatchment pour points (positive integers in field with field name defined in `field_pp_ws`) and to exclude subcatchments (negative integers). The latter is especially useful as inflow boundary condition, if another model or gauge input time series exists at the defined inflow point.

```
create_pourpoint(path_fnw, path_gdb_out, name_pp='pp', field_pp='ModelWatershed', print_out=False)
```

We do not need to exclude a subcatchment in this case, wherefore we only set one model domain pour point, which we give the ID number 1.

ATTENTION: Save pending edits and deselect all pour points before you proceed!

HINT: There is no online interaction between your Python environment and ArcGIS Pro. Therefore, you have to refresh the output geodatabase to view recent changes after a TATOO function has been completed.

Consequently, the user can calculate the initial model watershed using ***calc_watershed***. The user has the opportunity to correct the flow directions manually afterwards using point feature class not existing yet as it is created in the initial run. Therefore, the user has to set the following variables for the initial run:

- `initial = True`: This will tell the function not to expect any flow direction correction feature class, which will be created during the initial run of this function,
- `name_fd_p_corr = 'NAME'`: create flow direction correction feature class with the name,
- `path_fd_p_corr = ''`: path of the flow direction correction feature class does not exist yet and is therefore empty.

```
calc_watershed(path_dem_mr_cfnw_f, path_pp, path_gdb_out, name_fd_mr='fd_mr',
    name_fd_mr_corr='fd_mr_corr', name_fa_mr='fa_mr', name_ws_s='ws_s', initial=True,
    name_fd_p_corr='fd_p_corr', path_fd_p_corr='', print_out=False)
```

The function will calculate the flow direction, flow accumulation, snap the pour points to the flow accumulation, and calculate the watershed. Finally, it will exclude subcatchments with negative ID numbers in `field_pp_ws`.

Now, the user can inspect and manually correct the calculated model watershed creating flow direction correction points within the respective feature class `name_fd_p_corr`. This requires the following working steps:

1. Load the initial model watershed polygon feature class `name_ws_s` and the flow direction correction point feature class `name_fd_p_corr` in your GIS software. Other datasets like DEM-HR or the flow direction might also be helpful
2. Create point features in `name_fd_p_corr` defining cells whose flow direction shall be manipulated during the following, second calculation of ***calc_watershed***. These could be (i) cells, which are outside the model domain, but shall be within, or (ii) cells, which are included in the model domain, but shall be excluded.
3. Define integer numbers representing the desired D8 flow direction in the variable 'D8' (E: 1, SE: 2, S: 4, SW: 8, W: 16, NW: 32, N: 64, NE: 128).

In our example, we immediately see mismatches (see Figure 3). On the one hand side, some model elements are added to the model domain, where the DEM-HR coverage is far below 50 %. On the other, a tributary in the northwest is excluded, which according to our reference watershed shall belong to the model domain.

Consequently, we correct the watershed creating points with D8 integer numbers according to the flow direction we intend for the watershed calculation. The points' values will dominate the original flow direction values. The points have to be placed somewhere in the respective model element box.

ATTENTION: Save pending edits and remove the initial model domain from the ArcGIS Pro Map document before you proceed! The watershed polygon feature class is locked during display and the lock will generate an "ERROR 000317: … cannot be deleted. Failed to execute (Delete)".

For the iteration run of ***calc_watershed*** we have to change the mentioned three function variables:

- `initial = False`: This will tell the function to look for the correction feature class,
- `name_fd_p_corr = ''`: the feature class name is given with the path (see below),
- `path_fd_p_corr = 'PATH'`: path of the flow direction correction feature class.

We cannot correct the boundary of the missing tributary in the northwest, wherefore we have to expect a second iteration. Fortunately, it is not necessary, as the remaining boundary is well-drawen. The initial and corrected watersheds can be compared in Figure 3.

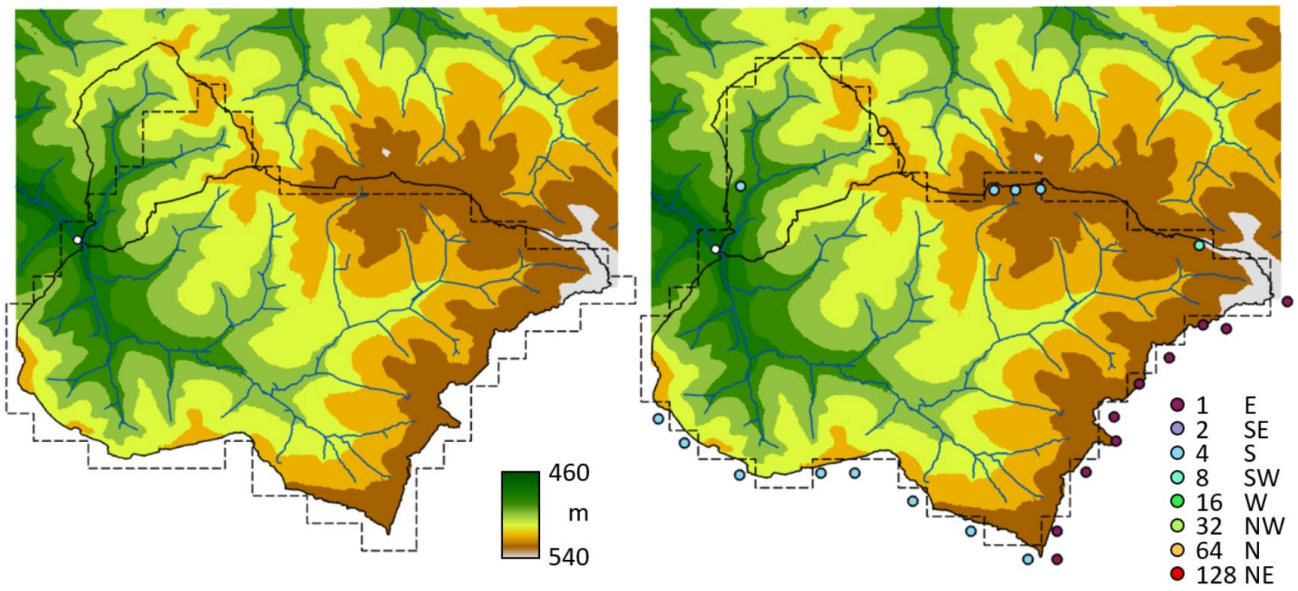

Figure 3: Initial (left) and corrected (right) model domain.

2.5 Calculate and correct model network and polygon raster representation (*calc_model_network* and *create_element_polyg*)

We can calculate the model network now applying ***calc_model_network***. The function will clip the DEMs, flow direction and flow accumulation using the derived model domain and in the next step calculate the model-resolution flow length (`name_fl_mr`). Afterwards it will determine the upstream-downstream cell dependencies and export the resulting model element center point (`name_tgb_p`) and model network polyline feature classes (`name_mnw`). If the input rasters are too large to be handled by ArcPy, then intermediate GeoTIFF files will be created and imported using the GDAL package. Therefore, the function needs an additional path (`path_files_out`) outside of the GDB for intermediate outputs.

```
df_data_tgb_p, df_tgb_up = calc_model_network(path_ws_s, path_fd_mr, path_fa_mr, path_gdb_out,
      path_files_out, name_fl_mr='fl_mr', name_tgb_p='tgb_p', name_mnw='mwn', print_out=False)
```

The function returns two pandas.DataFrames, which make the created information available within the respective Python environment, but which are not necessary for the further model creation process.

The model network (MNW) has to be corrected in the same way as the model domain, as the model resolution is course and does therefore not fit well to the detailled FNW. We can follow this workflow:

1. Load the initial model network polyline feature class `name_mnw` and the flow direction correction point feature class `name_fd_p_corr` in your GIS software. Other datasets like DEM-HR or the flow direction might also be helpful
2. Create point features in `name_fd_p_corr` defining cells whose flow direction shall be manipulated during the following, second calculation of ***calc_model_network***. Tip: You can activate a “snapping” functionality of GIS for start/end points of polylines and then comfortable locate the points at the connection points of the model network polylines.
3. Define integer numbers representing the desired D8 flow direction in the variable 'D8' (E: 1, SE: 2, S: 4, SW: 8, W: 16, NW: 32, N: 64, NE: 128).

To import the corrected flow direction, we have to run both functions ***calc_watershed*** and ***calc_model_network*** again.

ATTENTION: Keep care not to produce intersecting (crossing) flow network connections unintentionally!

ATTENTION: Save pending edits and remove the initial model domain and network feature classes from the ArcGIS Pro Map document before you proceed! The feature classes are locked during display and the locks will generate an "ERROR 000317: … cannot be deleted. Failed to execute (Delete)".

The amount of corrections of course depends on the information the user has at hand as well as the model's required level of detail and the chosen grid resolution.

As soon as the iteration run of the functions ***calc_watershed*** and ***calc_model_network*** is finished, we can compare the results and adjust missed segments iteratively (see Figure 4).

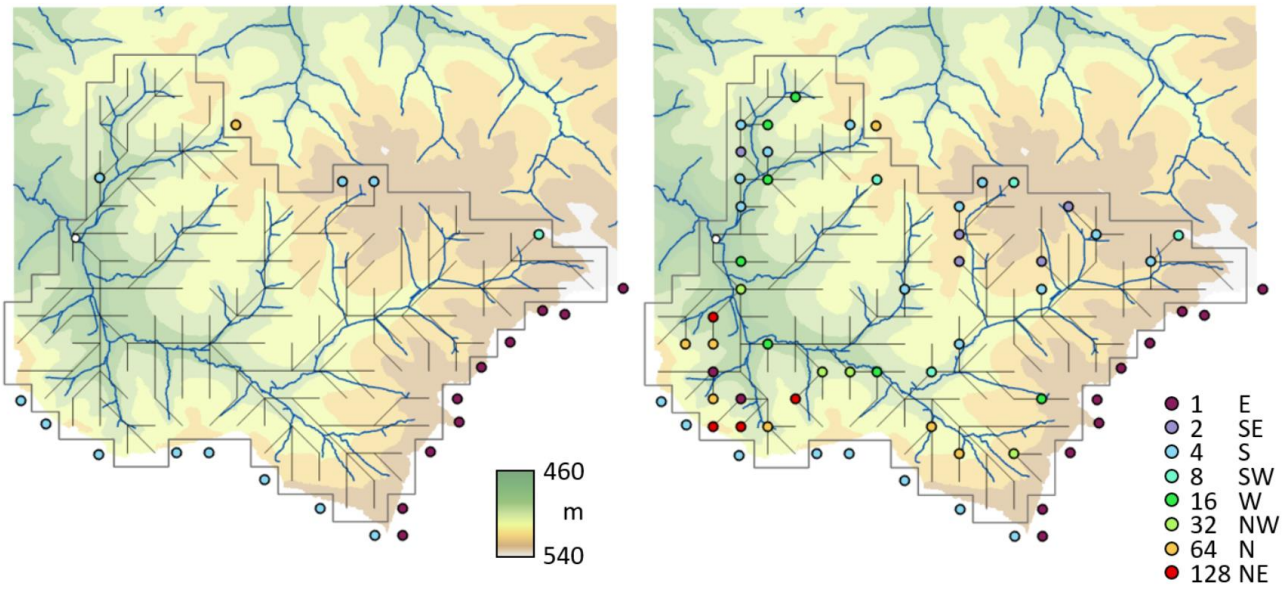

Figure 4: Initial (left) and corrected (right) model network.

Finally, we can produce a polygon feature class representing the raster model elements with ***create_element_polyg***. The function requires any produced model-resolution raster layer (`path_sn_raster`) to snap the created polygons. Later, the calculated parameters will be joined to facilitate the display.

```
create_element_polyg(path_tgb_p, path_sn_raster, path_gdb_out, name_tgb_s='tgb_s',
        print_out=False)
```

2.6 Prepare calculation of runoff concentration and standard routing parameters (*prepr_routing_rasters* and *create_fl_ind_point*)

After the model structure is established, we can begin with the calculation of the runoff concentration model parameters. This process requires many calculations and is therefore split in a preparation phase (this chapter) and a summary phase (next chapter).

We start preparing the rasters for the calculation using ***prepr_routing_rasters***. The function will clip the high-resolution DEM to the model domain (`name_dem_hr_ws`), fill it, and calculate its flow direction (FD-HR), accumulation (FA-HR, `name_fa_hr`) and length (FL-HR). Additionally, the function calculates each model element's minimum (`name_dem_min_mr`) and maximum (`name_dem_max_mr`) DEM-HR value as well as (if existing through FNW polylines within the respective model element) the aggregated high-resolution flow length (FL-FNW-MR, `name_fl_fnw_mr`).

```
prepr_routing_rasters(path_dem_hr, path_fnw, path_ws_s, path_fa_mr, path_gdb_out,
            name_fa_hr='fa_hr', name_dem_hr_ws='dem_hr_ws', name_fl_fnw_mr='fl_fnw_mr',
            name_dem_max_mr='dem_max_mr', name_dem_min_mr='dem_min_mr', initial=True,
            print_out=False)
```

ATTENTION: The calculation of the FL-HR might take some minutes up to few hours for large catchments for the tested ArcGIS Pro version 2.5 as it uses only one single CPU. Be patiant and do not interrupt the process too early!

As the calculation might take a while (especially because of the FlowLength function of ArcPy), the calculation time can be shortened using the switch `initial`, which will skip the superfluous repetitive calculation of the rasters in case of a necessary repetition.

The results for the example catchment are summarized in Figure 5.

INFO: The aggregated high-resolution flow length does not necessarily have zero as the lowest value. It is calculated from the clipped FNW polylines and depickted at the center of the elements. The discrepancy to zero will be corrected during later steps.

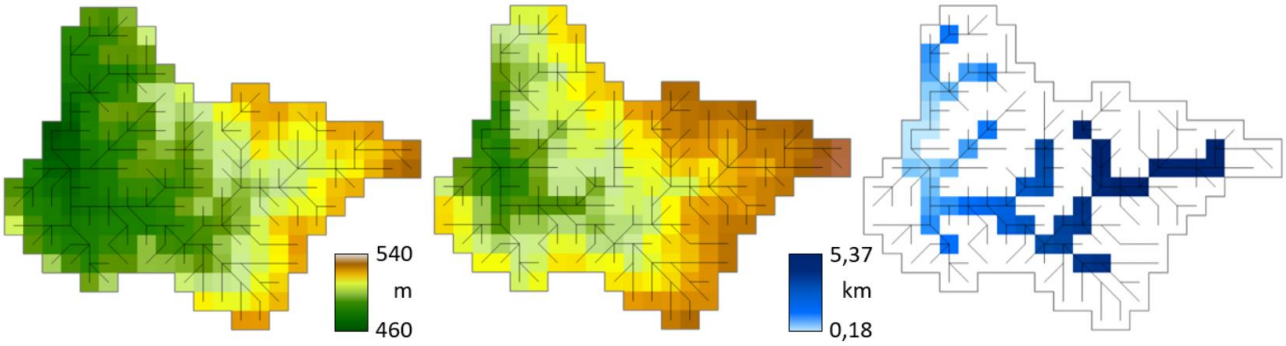

Figure 5: Resulting minimum (left) and maximum (middle) elevation and aggregated high-resolution flow length (right).

The calculation of the aggregated high-resolution flow length (`name_fl_fnw_mr`) is based on the intersection of FNW polylines and model raster. In most cases, this may result first in missing FL-FNW-MR values for model elements along the river, and second in mistakenly FL-FNW-MR > 0 values offside these. The first type is not problematic, as it will be balanced using upstream and downstream FL-FNW-MR values. To tackle the latter type, the FL-FNW-MR is set to null for all model elements with a FA-MR < 0.1 km² assuming that there will be no significant creek. Afterwards, the user can correct the remaining few wrong values by marking model elements as to be ignored during the eventual FL parameter calculation creating identification points. The related point feature class is created applying the function ***create_fl_ind_point***, which only requires a spatial reference object (`sr_obj`) for definition.

```
create_fl_ind_point(sr_obj, path_gdb_out, name_no_fnw_fl='no_fnw_fl', print_out=False)
```

The workflow do identify wrong elements is as follows:

1. Load the initial model network polyline feature class `name_mnw`, the point feature class identifying elements witn no FL-FNW-MR calculation (`name_no_fnw_fl`), and the raster representing the flow network flow length (`name_fl_fnw_mr`).
2. Create point features defining cells where no FL-FNW-MR shall be calculated. Tip: You can activate a “snapping” functionality of GIS for start/end points of polylines and then comfortable locate the points at the connection points of the model network polylines.

It is not necessary in the presented example case to do such a correction.

2.7 Calculate runoff concentration and standard routing parameters (*summar_gisdata_for_roandrout* and *calc_roandrout_params*)

Now, the model structure data (model ID numbers and types), the element coordinates and area, the raster data for elevations (minimum and maximum within model element), and flow lenghts (raster and FNW polyline calculation) are summarized in a point feature class (`name_tgb_par_p`) for the determination of runoff and routing parameters using ***summar_gisdata_for_roandrout***. The field name arguments can be left as they are.

```
summar_gisdata_for_roandrout(path_tgb_p, path_dem_max_mr, path_dem_min_mr, path_fl_mr,
            path_fl_fnw_mr, path_no_fnw_fl, path_gdb_out, name_tgb_par_p='tgb_par_p',
            field_dem_max_mr='dem_max_mr', field_dem_min_mr='dem_min_mr',
            field_fl_fnw_mean_mr='fl_fnw_mean_mr', field_fl_mr='fl_mr', print_out=False)
```

An example of the resulting attribute table is shown in Figure 6.

OBJECTID	Shape	tgb	tgb_down	tgb_type	tgb_dtgb	tgb_a	POINT_X	POINT_Y	fl_mr	dem_max_mr	dem_min_mr	fl_fnw_mean_mr
1	Point	1	3	headwater	-1	0	355200	5354400	5379,899	534,3643	529,179	<Null>
2	Point	2	3	headwater	-1	0	355200	5354600	5297,057	533,857	527,1752	<Null>
3	Point	3	4	routing	-1	0,08	355000	5354600	5097,057	531,6182	521,7267	<Null>
4	Point	4	6	routing	-1	0,12	354800	5354600	4897,057	527,5925	517,7704	<Null>
5	Point	5	6	headwater	-1	0	354600	5354800	4897,057	528,5189	520,8052	<Null>
6	Point	6	8	dummy	8	0	354600	5354600	4697,057	525,8408	515,729	5366,679
7	Point	7	8	headwater	-1	0	354800	5354800	4979,899	531,1134	522,7836	<Null>

*Figure 6: Snippet of the resulting attribute table of **summar_gisdata_for_roandrout.***

We have all GIS input data at hand now to calculate the parameters for the runoff concentration and the standard routing mode, what is done within the function ***calc_roandrout_params***.

```
df_data_tgbdat, ser_tgb_down_nd, ser_ft, ser_area_outfl, ser_ch_form_q =
            calc_roandrout_params(cellsz, q_spec_ch, name_tgb_par_p, field_dem_max_mr='dem_max_mr',
            field_dem_min_mr='dem_min_mr', field_fl_mr='fl_mr',
            field_fl_fnw_mean_mr='fl_fnw_mean_mr', def_fl_upper_lim=np.inf, def_fl_strct_mism=2,
            def_sl_min=0.0001, def_sl_excl_quant=None, def_zmin_rout_fac=0.5, def_zmax_fac=1,
            ser_q_in_corr=None, ch_est_method='combined', def_bx=0, def_bbx_fac=1, def_bnm=1.5,
            def_bnx=100, def_bnvrx=4, def_skm=30, def_skx=20, print_out=False)
```

This function fullfills many tasks and is therefore quite complex. It summarizes all necessary GIS data and calculates the runoff concentration and standard routing parameters. There are several nested functions within overall functional sections:

Data import and index calculations

At the beginning, all required input data is imported and missing up- and downstream model element index relations with and ignoring dummy elements are calculated (e.g., `ser_tgb_down_nd`).

flow lenght parameters

The next section deals with the flow length parameters. First, both flow length values (FL-MR and FL-FNW-MR) have to be “redistributed”. The LARSIM convention assumes the confluence point of cells upstream of the routing element, while the GIS flow length calculation is referenced to the center of each grid cell. Therefore, the resulting discrepancies at confluence points have to be balanced in upstream routing elements for both flow length datasets. Afterwards, the two data sets are merged using the FL-FNW-MR where possible and FL-MR otherwise. A maximum flow length limit per cell (*`def_fl_upper_lim`*) allows the control of unrealistic high values (e.g., $\texttt{def\_fl\_upper\_lim} = 3 \cdot \texttt{cell\_sz}$). The algorithm additionally ensures a minimum flow length value (*`def_fl_strct_mism`*) if there are structural mismatches evolving

through merging. This is essential for LARSIM, as a channel flow length of 1 m will deactivate the routing routine, wherefore the value should be set to 2 m. Finally, the function calculates the cumulative flow length values.

Elevation and slope parameters

The next section calculates the elevation difference for runoff concentration and the channel slope. A minimum channel slope (*def_sl_min*) is retained during this to prevent unrealistic simulation results of the Williams linear storage approach, as the assumption that the energy line is parallel to the channel slope does not hold for very small gradients (default: 0.0001). Additionally, the user may want to exclude unrealistically high slope quantiles, which might arise for high-resolution models defining *def_sl_excl_quant*. A value of 0.999 might be sufficient.

Runoff concentration parameters

The calculation of the runoff concentration with Kirpich requires a definition of "elevation difference" and "flow length" within a raster element, which is not trivial. The Kirpich formula is designed in a way to use the maximum and minimum elevation value of the subcatchment and the longest flow path, whereas LARSIM assumes the inflow of the subcatchment at the center of a rectangular model raster element. As rasters do not respect watersheds, it is necessary to find a definition. The standard setting of TATOO is to use the average of the routing channel's elevations at the upper and lower cell boundary as lower elevation and the maximum within the raster element as the upper elevation for runoff concentration.The user might nevertheless want to change this definition. The factors *def_zmin_rout_fac* (default: 0.5) and *def_zmax_fac* (default: 1) are available for this. The flow length is derived as the half diagonal value of the element.

Cross section parameters

For the standard setting, the cross section geometry is estimated using formula related to bankful discharge, which is assumed to be around a return period of 2 a for natural conditions (it might be much higher in populated regions, as people want to keep discharge inside the river bed).

The channel forming discharge is calculated linearly extrapolating gauge statistics using the inflow catchment area of each model element and corrected for some inflow values if defined (*ser_q_in_corr*). Afterwards, the channel width and depth are estimated using the desired formula (*ch_est_method*), while the bank slope is *def_bnm* for both sides (default: 1.5). The even channel foreland width is *def_bx* (default: 0), while the inclined one is assumed to have a slope of *def_bnx* (default: 100) and a width equal to the channel multiplied with *def_bbx_fac* (default: 1). The outer valley slope is defined with *def_bnvrx* (default: 4). Last but not least the hydraulic roughnesses for channel and forelands have to be defined using *def_skm* (default: 30) and *def_skx* (default: 20).

Finalization and export

Finally, the function calculates few additional informative variables as, e.g., model element name, model element area (*ser_ft*), outflow catchment size (*ser_area_outfl*), or channel-forming discharge (*ser_ch_form_q*), summarizes the informations and provides them as a pandas.DataFrame (*df_data_tgbdat*).

2.8 Summarize and export model element information (*write_tgbdat* and *df_to_table*)

As now all parameters are ready, we can apply the functions to export the parameters as model file (tgb.dat) and GIS information (ArcGIS Table and joined to the model element polygons). Therefore, we apply the functions ***write_tgbdat*** and ***df_to_table***.

```
write_tgbdat(df_data_tgbdat, path_tgbdat, def_tgb_nodata_val=-1, hcs_epsg=31467, vcs_unit='m ue.
        NN', src_geodata='', catch_name='', comment='', print_out=False)

df_to_table(df_dat, path_gdb_out, name_par_tab)
```

The first function takes few variables, which are the used nodata value (*def_tgb_nodata_val*), the horizontal coordinate EPSG code (*hcs_epsg*), the vertical unit system (*vcs_unit*), and informative strings (*src_geodata*, *catch_name*, *comment*).

In this way, we can easily display and inspect the calculated parameter values (see Figure 7).

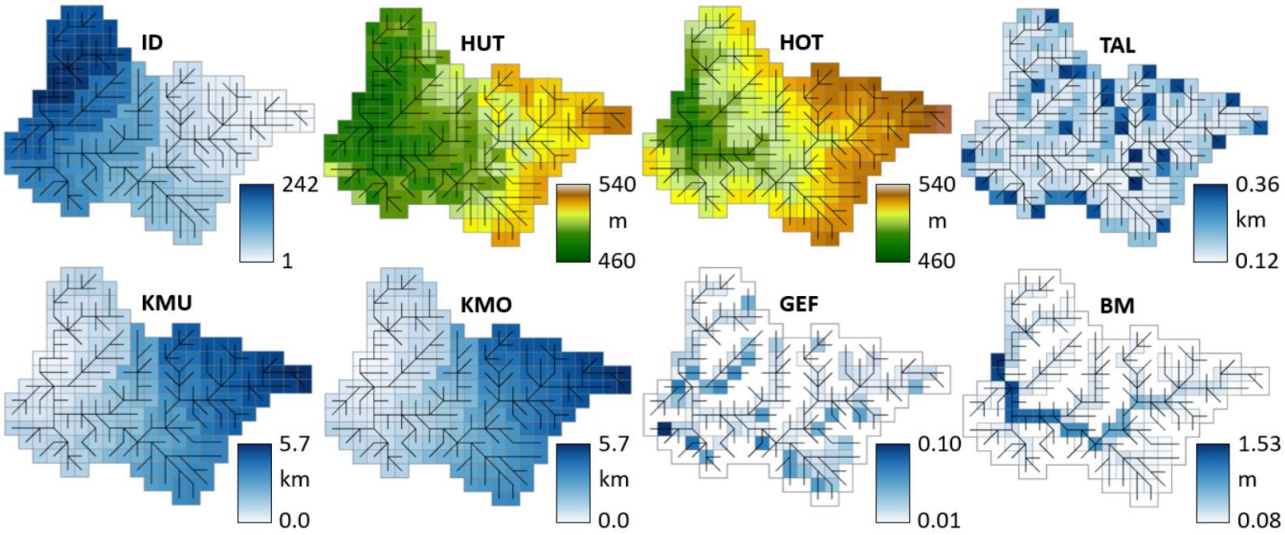

Figure 7: Summary of some representative parameters calculated with TATOO for runoff concentration and routing.

2.9 Implementation of detailled river cross sections (*create_csl*, *df_profdat_from_cs*, and *write_profdat*)

The estimation of cross section parameters from channel-forming discharge is suitable for natural catchments, where we do not have a DEM-HR at hand. Otherwise, we can achieve much better simulation results especially during floods if we implement more realistic cross sections directly derived from the DEM-HR. The TATOO library offers the functions to do so leaving the user the ability to correct the automatically derived cross sections manually. Three functions have to be run to get a profile.dat model input file ready-to-use for LARSIM: First, the cross-sections are automatically derived based on the given flow network (***create_csl***). Second, the user can modify them. Third, the corrections are implemented (***df_profdat_from_cs***) and output files created (***write_profdat***). We will have a look at all three steps in the next parts.

Automatically create cross-section lines

At the beginning, the function ***create_csl*** can create automatically cross sections along the river network (`name_csl`).

```
create_csl(path_fnw, path_ws_s, path_fa_hr, def_cs_dist, def_cs_wmax_eval, path_gdb_out,
        name_csl='csl', name_fnw_o='fnw_o', def_river_a_in=0, def_cs_search_rad=5,
        print_out=False)
```

The user has to define the longitudinal distance (`def_cs_dist`) between and the width (`def_cs_wmax_eval`) of the cross sections. Optimally, one CSL should be defined per model element, but users have to delete some in many cases. Therefore, I recommend $\texttt{def\_cs\_dist} = \texttt{cell\_sz} \cdot 2$. The width

should cover minimum the expected inundated width of the valley at the catchment outlet, what depends on catchment size, runoff ratio and valley shape.

The minimum inflowing catchment (`def_river_a_in`) defines a threshold from which CSL are created. The clipped flow network is given as output (`name_fnw_o`). The function will use the channel-forming discharge estimation formula for model elements with smaller inflowing catchment. It makes sense to choose the value accordingly to the quality and spatial resolution of the DEM-HR: If the quality is high enough to resolve small channels, than the threshold can be low (or even zero, which is the default). The estimation formula tendentially will create steep channel-kind cross sections. The variable `def_cs_search_rad` is a technical variable to ensure correct behaviour at confluence points. Recommendation:
`def_cs_search = cell_sz · 5`.

Correct cross-section lines

After creating the cross sections, users can modify them if needed. This is often the case, as (1) there are mistakes in the DEM, which the cross sections should not reflect too, and (2) the automatically chosen position of the cross sections might not be suitable for the representation of the hydraulic conditions.

The user has different possibilities to modify the cross section lines (CSL). The workflow is as follows:

1. Load the MNW polylines (`name_mnw`), the polygons representing the model raster (`name_tgb_s`), the CSL (`name_csl`) and the DEM-HR (`name_dem_hr`)
2. Manipulate the cross sections fullfilling the following requirements
    a. Only one cross section is allowed per model element
    b. The cross section mid-point (intersection with flow network) has to be within a routing cell (no headwater element!)
    c. Recommended: the cross section should capture a section of the valley that is representative for the model element it is located in
    d. Recommended: intersections of cross sections should be avoided.
3. You may additionally do the following edits:
    a. Turn a cross section.
    b. Move a cross section. Be aware: Mid-point has to stay snapped to the FNW and the CSL's field *fnw_o_rp_fid* value has to stay the same as the intersecting FNW's element *fnw_o_fid* value or it has to be adapted (see Figure 8).
    c. Delete the cross section.
    d. Add a cross section. Be aware: The CSL's field *fnw_o_rp_fid* value has to be defined the same value as the intersecting FNW's element *fnw_o_fid* value.

ATTENTION: Save pending edits and deselect all pour points before you proceed!

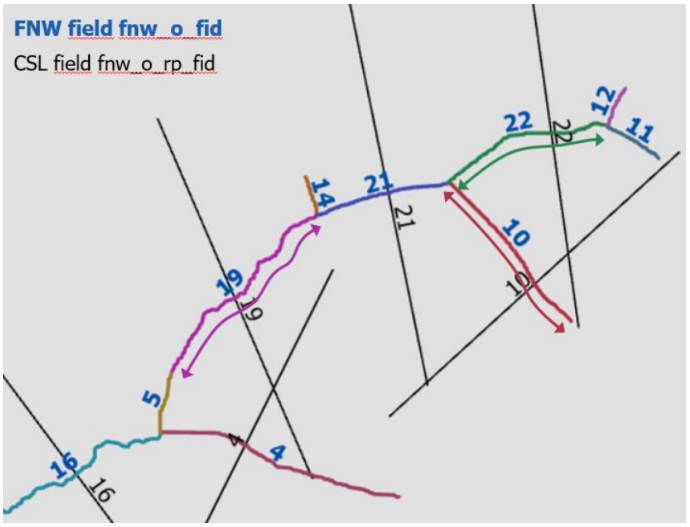

Figure 8: Exemplary FNW sections, where the CSL can be moved to without adapting any attribute numbers.

Figure 9 compares the CSL before and after the manual corrections.

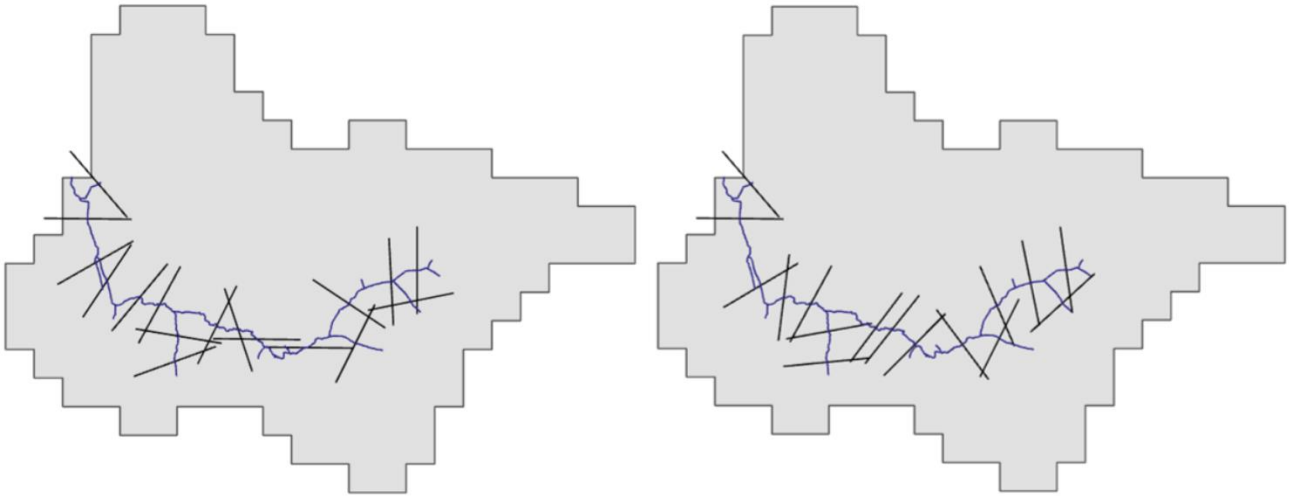

Figure 9: Automatically derived (left) and manually corrected (right) cross-section lines.

Calculate parameters and create model files

Now, we can calculate the parameters (`df_profdat_from_cs`) and create the output file (`write_profdat`) used with the option PROFILE EXTERN.

Some major steps can be distinguished in the first function:

1. Get coordinates and elevation values along the CSL in a resolution defined by `def_cs_intp_buf_dist`. The resolution will be refined in the channel region according to the chosen horizontal (`def_ch_wres`) and vertical resolution (`def_lam_hres`). The latter has to fit to the definitions in the L15 parameter of LARSIM.
2. Calculate center and intersection points of CSL with FNW.
3. Clip CSL
    a. handling intersection points with multiple FNW polylines
    b. applying maximum evaluation elevation (`def_cs_hmax_eval`), and
    c. ensure monotonically increasing CSL parts at both river sides.
4. Add artificial channel with depth `def_ch_h` and width `def_ch_w`, if necessary.

5. Find bankful water level using a maximum channel search width (`def_ch_wmax_eval`), a minimum channel bank slope (`def_chbank_slmin`), a minimum channel depth (`def_ch_hmin`), as well as a minimum (`def_ch_hmin_eval`) and maximum (`def_cs_hmax_eval`) channel search depth (latter is only required for plot outputs). The minimum (`def_ch_vmin`) and maximum (`def_ch_vmax`) velocity thresholds define a reasonable velocity bandwidth for informative outputs which do not influence the model parameters.
6. Calculate and correct parameters for each CSL including rounding applying the decimal digits of the output file (`def_profdat_decmax`).
7. Allocate CSL to model elements.

The function will return pandas.DataFrames, which summarize all parameters (`df_profdat_par`) and the assignment of cross sections and model elements (`ser_tgb_csl`).

The users can additionally show (`ctrl_show_plots`) and save (`ctrl_save_plots`) plots from the channel optimization. An example graph shows Figure 10.

```
df_profdat_par, ser_tgb_csl = df_profdat_from_cs(path_fnw, path_fnw_o, path_csl, path_dem_hr,
            path_fa_hr, path_tgb_s, ser_tgb_down, ser_tgb_down_nd, ser_tgb_type_routing,
            ser_ch_form_q, ser_pef_bm, ser_pef_hm, path_gdb_out, name_profile_par='profile_par',
            name_ch_fit_coords='ch_fit_coords', name_bcwsl='bcwsl', def_cs_wmax_eval=600,
            def_cs_intp_buf_dist=1, def_ch_w=0.5, def_ch_h=0.5, def_ch_wres=0.05,
            def_cs_hmax_eval=10, def_lam_hres=0.1, def_ch_vmin=0.5, def_ch_vmax=3.0,
            def_ch_wmax_eval=40, def_chbank_slmin=0.1, def_ch_hmin=0.2, def_ch_hmin_eval=0.1,
            def_profdat_decmax=2, ctrl_show_plots=False, ctrl_save_plots=False, ser_ft=None,
            ser_area_outfl=None, def_ch_hmax_eval=None, path_plots_out=None, print_out=False)
```

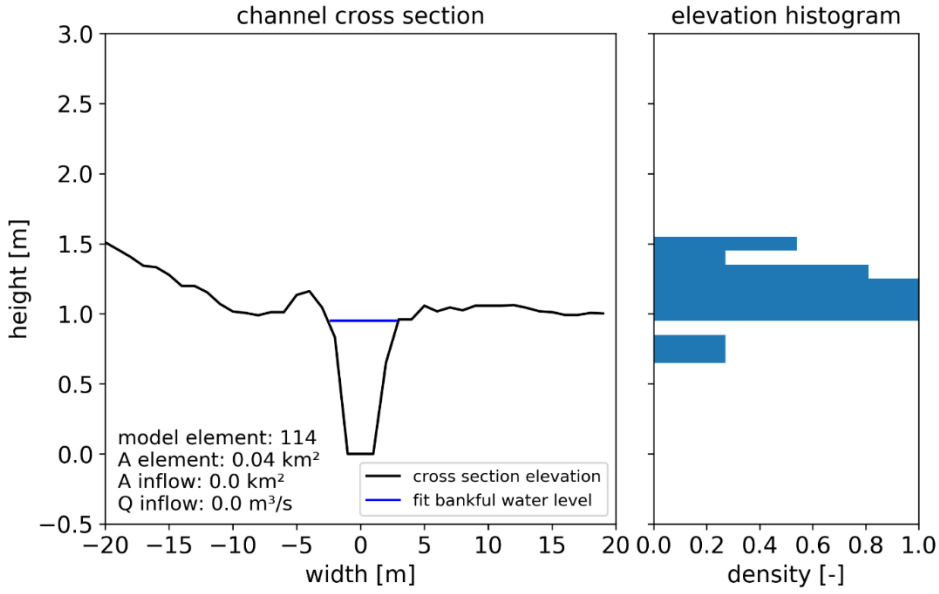

Figure 10: Exemplary result of the cross section channel optimization.

Finally, we can write the model file profile.dat using the function ***write_profdat***. The function requires inputs for the nodata value (`def_profdat_nodata_val`), the separator indication between profile definitions in the file (`def_profdat_exit_val`), and some comment strings (`src_geodata`, `catch_name`, `comment`).

```
write_profdat(df_profdat_par, ser_tgb_csl, path_profdat, def_cs_hmax_eval, def_lam_hres,
            def_profdat_nodata_val=-1, def_profdat_exit_val=999999, src_geodata='', catch_name='',
            comment='', print_out=False)
```

3 How to create a subcatchment model using TATOO

3.1 Overview

The following section describes the workflow *tatoo_subcatch_example.py* to create a LARSIM subcatchment model with TATOO. The workflow to create subcatchment models consists of 7 functions including two potential or necessary user interventions (see workflow in Figure 1).

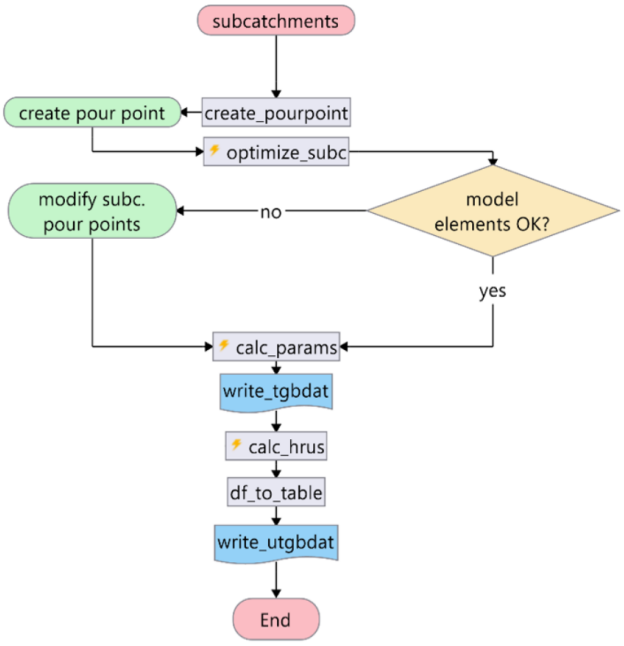

Figure 11: Workflow to create a LARSIM raster model with TATOO

The following descriptions explain the functionalities step-by-step using the test data set of an 8-km² catchment in Eastern Bavaria in Germany.

3.2 Set paths and names for input and output data

The following variables have to be set at the beginning defining the inputs and outputs:

- geodatabase (GDB) where inputs are stored (`path_gdb_in`)
- the respective input layer paths of DEM, FNW, land use, and soil (`path_dem, path_fnw, path_lu, path_soil`)
- geodatabase where outputs are stored (`path_gdb_out`)
- output path for model files (`path_files_out`)
- output path for figure files, if storage is activated (`path_plots_out`)

ATTENTION: Keep the '\\' in the end of the path variables!

All other layer and file names can be left as default but can be changed if required.

3.3 Create pour point, calculate optimized subcatchments for the model domain and correct their pour points (*create_pourpoint* and *optimize_subc*)

For subcatchment models, no DEM preprocessing step is necessary. Therefore, the first step is the creation of a pour point feature class identifying the desired model domain using the function ***create_pourpoint***. Creating feature points within that feature class (e.g., using the software ArcGIS Pro) will enable the user to add ID numbers to the subcatchment pour points (positive integers in field with field name defined in

field_pp_ws) and to exclude subcatchments (negative integers). The latter is especially useful as inflow boundary condition, if another model or gauge input time series exists at the defined inflow point.

```
create_pourpoint(path_fnw, path_gdb_out, name_pp='pp', field_pp='ModelWatershed', print_out=False)
```

We do not need to exclude a subcatchment in this case, wherefore we only set one model domain pour point, which we give the ID number 1.

ATTENTION: Save pending edits and deselect all pour points before you proceed!

HINT: There is no online interaction between your Python environment and ArcGIS Pro. Therefore, you have to refresh the output geodatabase to view recent changes after a TATOO function has been completed.

Consequently, the user can calculate the optimized model subcatchments using ***optimize_subc***. The function requires three numerical parameters:

- def_sc_area: This number defines the desired subcatchment size. Users should keep in mind, that the optimizer will in the mean produce subcatchments with this size as long as there are no conflicts arising due to close confluence points. The overall mean will be approximately 20 % lower than the defined value.
- def_fl_tol: This parameter defines the minimum flow length distance between two confluence points. Subcatchments with smaller flow length differences will be merged with others. A value of 0 will result in no correction of very small subcatchments.
- h_burn: This value defines the burning depth of the given flow network in the DEM.

```
pp_points_df = optimize_subc(path_dem, path_fnw, path_pp_ws, field_pp_ws, def_sc_area,
           def_fl_min_tol, path_gdb_out, path_files_out, name_dem_c='dem_c', name_fd_c='fd_c',
           name_fa_c='fa_c', name_fl_c='fl_c', name_ws_s='ws_s', name_pp_sc='pp_sc',
           name_sc_ws_s='sc_ws_s', name_fnw_fa='fnw_fa', h_burn=10, print_out=False)
```

Now, the user can inspect and manually correct the calculated subcatchments' pour points. This requires the following working steps:

1. Load the overall model watershed polygon feature class name_ws_s, the optimized subcatchments' pour points pp_sc, the watershed polygons of optimized subcatchments sc_ws_s, and the flow path network calculated from flow accumulation fnw_fa.
2. Check size and position of subcatchment polygons and their pour points.
3. If neccessary, adapt position of created pour points using "edit"-tools. You may (i) move, (ii) create or (iii) delete points.

ATTENTION: Keep care of the ID numbers. They have to be unique in the end.

ATTENTION: Save pending edits.

In our example, we do not have conflicts to solve (see).

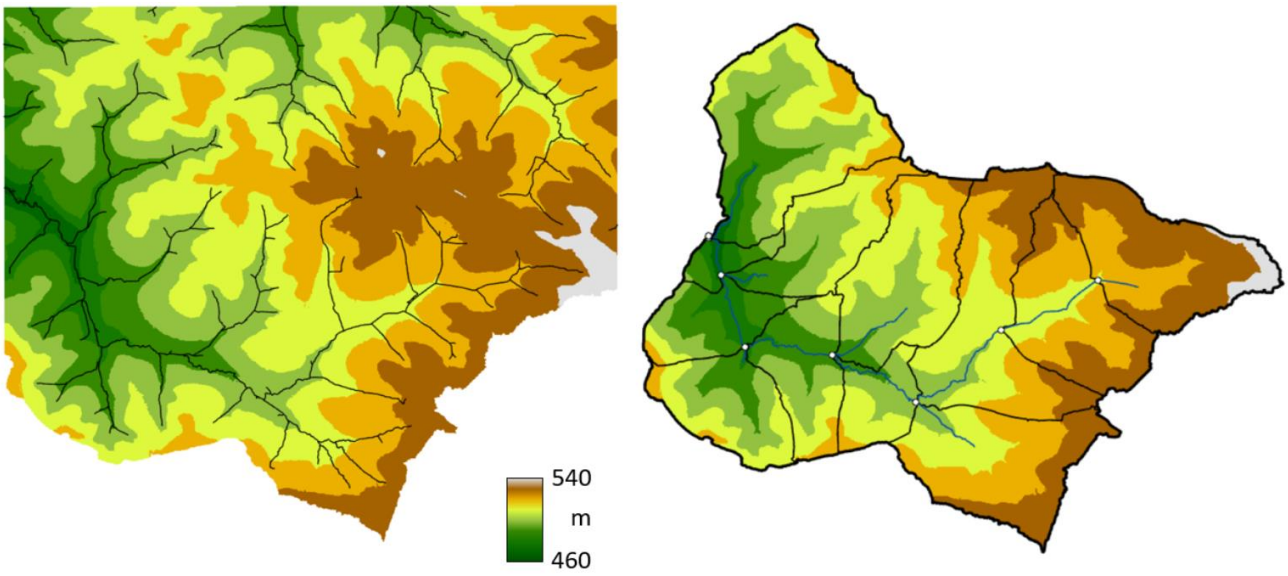

Figure 12: Original DEM and FNW (left) and resulting subcatchments for a desired size of 1.0 km² (right), source: LDBV.

3.4 Calculate model network, runoff concentration, and routing parameters (calc_params)

While the calculation of model network, runoff concentration, and routing parameters is rather complicated and therefore split into several functions for raster models it only requires one overall function for subcatchment models: ***calc_params***. Compared to the raster workflow, this function includes detailled cross section geometry calculations as well – if activated. The function consists of three working steps calling overall 26 subfunctions:

1. calculate model network including up- and downstream relations
2. calculate runoff concentration parameters
3. calculate standard or detailled routing parameters

The parameters for the calculation of the model network, the runoff concentration, and standard routing parameters are mostly the same as for the raster models; please have a look in chapters 2.5 to 2.7. Nevertheless, there is a small difference: There is no parameter `def_zmax_fac`. TATOO delineates subcatchments using high-resolution DEM and they therefore represent realistic surface subcatchments compared to raster elements. It does not make sense to use a lower value than the maximum found DEM value for runoff concentration.

```
df_data_tgbdat = calc_params(hq_ch, hq_ch_a, path_dem_c, path_fd_c, path_fa_c, path_fl_c,
           path_pp_sc, path_files_out, path_gdb_out, name_tgb_p='tgb_p', name_tgb_sj='tgb_sj',
           def_sc_area=1000000, def_a_min_tol=50, def_zmin_rout_fac=0.5, def_sl_excl_quant=0.999,
           ch_est_method='combined', ser_q_in_corr=None, def_bx=0, def_bbx_fac=1, def_bnm=1.5,
           def_bnx=100, def_bnvrx=4, def_skm=30, def_skx=20, ctrl_rout_cs_fit=False,
           def_cs_dist_eval=50, def_cs_wmax_eval=600, def_cs_hmax_eval=10, def_flpl_wmax_eval=50,
           def_ch_wmax_eval=40, def_ch_hmin_eval=0.1, def_ch_hmax_eval=3.0, def_ch_hmin=0.2,
           def_ch_vmin=0.5, def_ch_vmax=3.0, def_chbank_slmin=0.1, def_val_wres=10.0,
           def_flpl_wres=0.20, def_ch_w=0.5, def_ch_h=0.5, def_ch_wres=0.05, def_lam_hres=0.1,
           ctrl_show_plots=False, ctrl_save_plots=False, path_plots_out=None, print_out=False)
```

The usage of high-resolution data for cross sections is different for subcatchment models. Instead of using a separate file with perimeter-area-discharge relationships triple-trapezoids are fitted to an average profile of cross section samples within the subcatchments. Subcatchment models usually have a larger element size than raster models, what makes it difficult to find one representative cross section for each element. Therefore, TATOO derives cross sections in a defined distance () with the desired width () and merges them

using the average cross section. Consequently, a trapezoid is fit to the channel and two others on top for the flood plain and the valley. There are few new parameters appearing, which are important for the fitting of the triple trapezoid cross sections: `def_flpl_wmax_eval` defines the maximum allowed width of the horizontal flood plain. `def_val_wres` and `def_flpl_wres` are parameters, which together with `def_ch_wres` define the horizontal point resolution in different parts of each cross section: valley, flood plain, and channel. The resolution should be fine close to the channel and get coarser to the outer parts. This will ensure a good-quality fit along the river, where most of the time water flows. The outer parts will be flooded seldomly and the influence of the cross section decreases due to high water levels, wherefore the cross section does not have to be so accurate.

ATTENTION: The algorithm will use the estimated (standard) cross section parameters instead of the high-resolution if (1) there is cross section in the subcatchment due to a large distance between cross section samples compared to the defined minimum flow length or (2) if the fit failed. It throws a list of (missing) parameters if `print_out` is active.

For the example explained here, we get cross sections for all subcatchments with minimum one upstream neighbour as LARSIM automatically deactivates the routing for all headwater elements. The parameters of the triple trapezoid fits are summarizes in Table 2. It can be seen, that the cross section parameters for subcatchment 13 are not fitted but estimated. The subcatchment is very small making it non-reasonable to define a cross section.

	type	bm	hm	bnm	bl	br	bbl	bbr	bnl	bnr	bnvrl	bnvrr	ch	fll	flr
2	fit	0.2	0.6	5.1	0.0	0.0	18.4	0.3	50.5	31.2	14.9	30.1	-1.6	-0.7	-0.6
3	fit	1.6	0.5	5.2	0.0	0.0	5.7	92.2	28.8	20.3	19.0	48.5	-0.7	-0.6	-0.8
5	fit	0.3	0.9	7.1	0.0	0.0	6.8	16.9	21.7	58.0	14.1	30.5	-3.6	-2.2	-0.6
7	fit	0.6	1.2	6.6	0.0	0.0	10.3	7.4	5.3	52.8	12.3	23.3	-2.4	-3.0	-1.6
9	fit	0.8	0.9	3.8	11.7	0.0	32.1	21.4	21.5	9.5	10.6	27.7	-1.2	-1.9	-3.9
11	fit	0.7	1.1	4.4	10.4	0.0	19.9	13.5	11.7	33.4	5.8	8.4	-1.3	-1.7	-1.4
13	est	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Table 2: Parameters of all optimized mean cross sections. The three negative columns at the right summarize the least squares of the fit for the left (fll) and right (flr) foreland as well as the channel (ch). The string "fit" in column type shows a successful fit, the string "est" the use of the standard estimated parameters.

As an example, Figure 13 and Figure 14 show the cross section fitting for subcatchment 9.

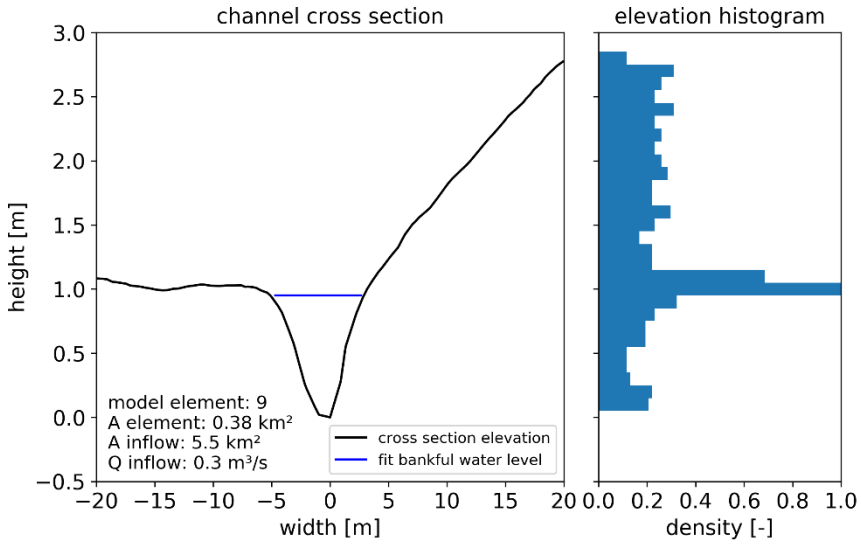

Figure 13: Fit of the bankful discharge for the mean cross section of subcatchment 9.

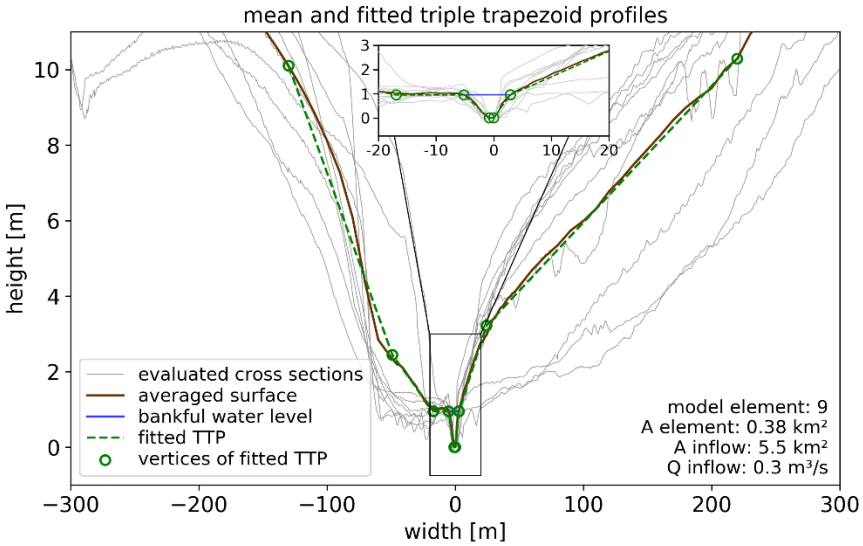

Figure 14: Successful fit of the triple-trapezoid parameters for the cross section of subcatchment 9.

The parameters are exported using the same function ***write_tgbdat*** as for raster models (see section 2.8). All high-resolution parameters are directly exported with this file.

4 Hydrological response units (*calc_hrus*, *df_to_table*, and *write_utgbdat*)

This procedure is the same for raster and subcatchment models. Most integrated hydrological models need some kind of soil and land use data. Using ***df_hru***, the user has to provide at least land use class polygons, usable field and air capacity (see Figure 15).

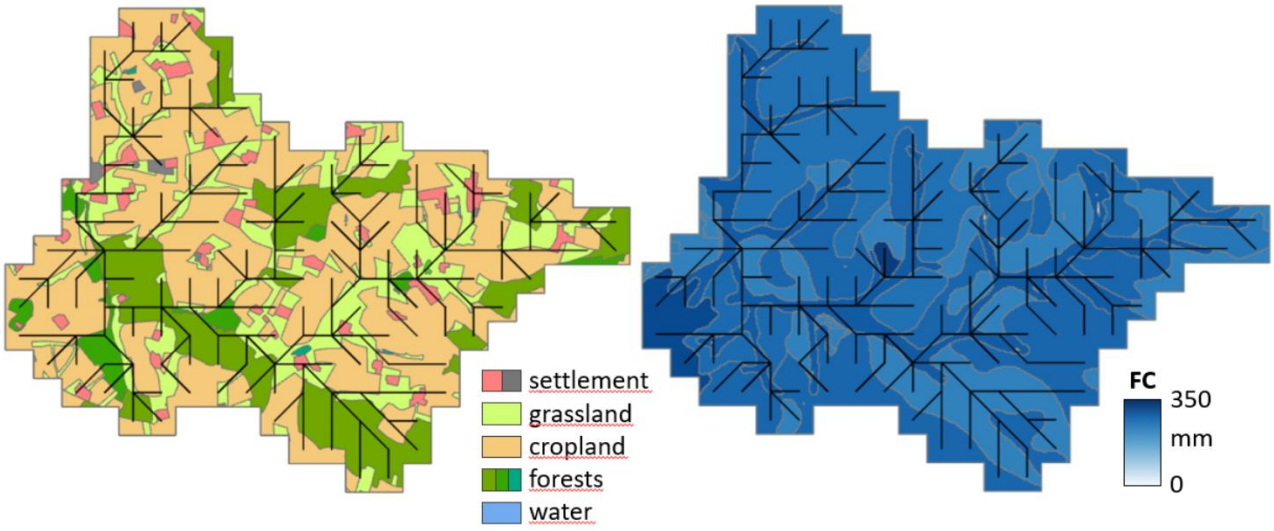

Figure 15: Exemplary land use (left) and soil (right) data for the raster model domain, source: LfU.

Additionally, it is possible to activate more detailled processes setting options with the Boolean True. These are the impervious share of each land use polygon (`ctrl_opt_impperc`), capillary rise capacity (`ctrl_opt_capr`), seven parameters to discribe infiltration capacity based on Green & Ampt (e.g., saturated conductivity or suction at water front, `ctrl_opt_infdyn`), and the silting-up module (`ctrl_opt_siltup`). Finally, the user can decide if TATOO shall aggregate the HRUs to reduce computational time or if it shall leave them, as they are (`ctrl_hru_aggr`). Latter might be useful to couple the model with a hydrodynamic model using effective rainfall. The function will import all defined fields of the input layers, intersect, and summarize the values accordingly.

```
df_hru = calc_hrus(path_tgb_sj, path_soil, path_lu, f_tgb, f_lu_id, lu_id_imp, lu_id_water,
          def_amin_utgb_del, path_gdb_out, name_hru_c='hru_c', ctrl_opt_impperc=False,
          f_impperc='', ctrl_opt_infdyn=False, df_lu_mp=None, ctrl_opt_siltup=False,
          ctrl_opt_capr=False, ctrl_hru_aggr=False, print_out=False)
```

Afterwards, the values can be exported as ArcGIS Table (***df_to_table***) and/or as model file (***write_utgbdat***). The latter function requires inputs for the nodata value (`def_utgb_nodata_val`) and some comment strings (`src_geodata`, `catch_name`, `comment`).

```
write_utgbdat(df_data_utgbdat, path_utgbdat, ctrl_opt_infdyn, ctrl_opt_impperc, ctrl_opt_capr,
          ctrl_opt_siltup, def_utgb_nodata_val=-1, src_geodata='', catch_name='', comment='',
          print_out=False)
```