Conversion of geophysical inputs of HYDROTEL to Raven hydrologic modelling platform

Mohammad Bizhanimanzar, Ph.D., Scenarios and Climate Services group, Ouranos,
mohbiz1@ouranos.ca

1. Objective

The objective of this report is to detail the similarities and differences in the input structure of the Raven hydrologic modelling platform [2] against HYDROTEL hydrologic model [1]. The latter is used by the Québec Direction de l'Expertise Hydrique et Atmosphérique (DEHA) to produce hydrological simulations under various climate change scenarios as well as for streamflow operational forecasting. Since Raven can support a variety of modelling approaches, it has been chosen by Ouranos to explore the uncertainties associated with hydrological model structure and their effects on projected streamflows of watersheds in Québec. This report, presents the methodology for converting the geospatial inputs of Hydrotel to a format required by Raven. The process of delineation of subbasin and corresponding Hydrological Response Units (HRUs) map along with required attributes by Raven are presented. In addition, the process of integration of HydroLAKES [5] for parametrization of lakes in the watershed are discussed.

2. Spatial discretization of watershed in HYDROTEL

HYDROTELis a semi-distributed hydrologic model in which a watershed is divided into Relatively Homogeneous Hydrological Units (RHHUs), and further into different land use classes for spatial representation of the hydrologic processes. The model has its own GIS interface, called PHYSITEL. As with other semi-distributed models, the user is required to provide the altitude raster map and vector map of the river network to determine the cell by cell runoff direction. The river network map allows to distinguish lakes and ponds and better represent the meandering river which is difficult to obtain using only the Digital Elevation Model (DEM) map [1]. The river network cells in the model are identified as cells with a drainage area equal or greater than a given threshold specified by the user. The threshold, can therefore, determine the level of details of the hydrographic network: smaller its value, more cells are identified as river. Once the river network is delineated, the outlet of the watershed is identified and the surface area draining to the outlet (total surface area of the watershed) is determined.

The delineated river network will then need to be linked to the RHHUs composed of a set of cells draining to a common outlet, as shown in Figure 1. In HYDROTEL, not all the RHHUs have a unique corresponding river reach, which means the total number of RHHUs can be different from that of the river reaches[1]. Also, the identification (ID) numbers associated with the river reaches are different from the RHHU ID numbers, as shown in Figure 2 for the Abitibi region. The total number of RHHUs for this region is 4289, while its river network contains only 1644 river reaches. For instance, the surface and subsurface runoff calculated for the RHHUs with the ID 14, 15, and 16 will be added to the river reach with the ID number 5. The RHHU

[1] Note that in other semi-distributed hydrologic models such as SWAT [3], a subbasin in the model is linked to a unique river reach. At each time step, the surface and subsurface runoff calculated at HRU level is routed to corresponding river reach and the outflow of the reach (after subtraction of evaporation from reach, infiltration through channel bed) are routed to the immediate downstream reach; this process is repeated for the entire river network to determine the watershed outflow at each time step.

average size depends on the resolution of the hydrographic network: a less detailed river network is associated with larger RHHU average size.

Once the RHHU and river network are created, the next step is importing the soil and land use map so that the proportion of different land use and soil classes in calculation of the hydrological processes is taken into account. In HYDROTEL, nine classes of land use are recognized: no data, water, bare soil, deciduous forest, range, coniferous forest, impermeable surface, peatland, wetland. As for the soil (texture) map, 20 soil types are defined to represent the soil heterogeneity of the watershed.

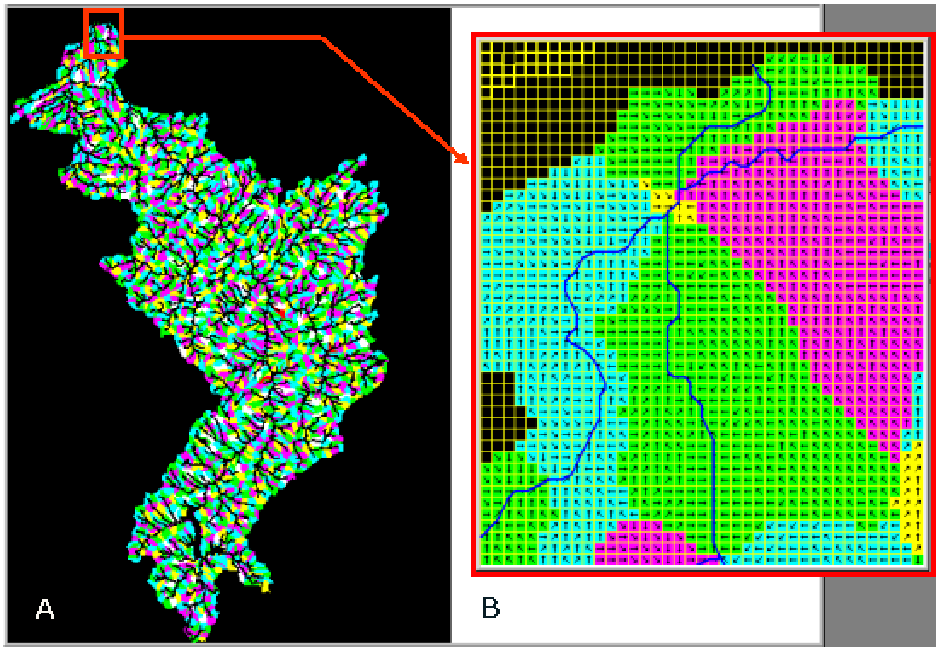

Figure 1. Delineation of RHHUs in Chaudière watershed (A) and the RHHUs for a rectangular box near watershed’s outlet (B) [1].

These soil types are: sand, loamy sand, sandy loam, loam, silt loam, sandy clay loam, clay loam, silty clay loam, sandy clay, silty clay, clay. The hydraulic properties of the soil types are defined based on Rawls’ [4] database. It is important to note that RHHUs are the computational elements in HYDROTEL, which means that the vertical water balance, evapotranspiration, snow water estimation and interpolation of the meteorological data are performed at RHHU level. **As such, only the soil type occupying the greatest proportion of the RHHU is considered for the calculation of the hydrological process of the RHHU**. Therefore, if the spatial variation of the soil type is important, a greater number of RHHUs (smaller in size) must be used [1]. The calculated surface and subsurface runoff at RHHU level and at each time step is given to the overland flow routing module, which distributes the runoff to corresponding reach and channel routing proceeds transformation of the water between reaches using kinematic or diffusive wave approach [1].

In summary, the geospatial inputs to PHYSITEL such as DEM are used to delineate the RHHUs which are calculation elements in HYDROTEL. The hydrological processes will then be performed over each landuse class and dominant soil type at RHHU level.

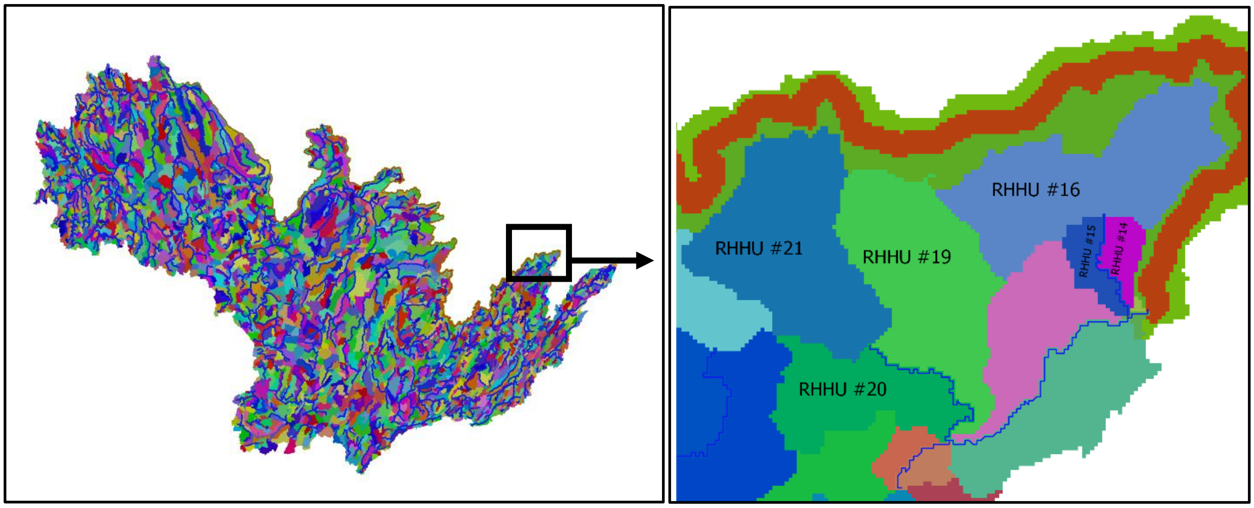

Figure 2. (left) RHHU map of Abitibi region delineated by Physitel and (right) connection between RHHUs within a rectangular box.

3. Definition and parametrization of subbasins and HRUs in Raven

Unlike the similarities in the definition of subbasin in HYDROTEL and Raven, the correspondence between subbasins and river reaches are different in these models. In Raven, a subbasin interacts with a unique river reach where the flow order among the reaches are determined using subbasin's downstream_ID. The subbasin is further divided into HRUs, each having a unique combination of soil type, land use, vegetation cover, aquifer type. The Raven's .rvh configuration file provides the information related to the subbasins and HRUs; the following inputs are required for each subbasin block:

1) ID: a positive unique integer for each subbasin;
2) Name: the nickname of the subbasin;
3) Downstream_ID: the ID of subbasin receiving water from current subbasin;
4) Profile: the channel profile code specified in the .rvp file;
5) reach_length (length of reach in the subbasin in km);
6) Gauges (Flag which determines whether the subbasin is gauged (1) or not (0)).

Note that the channel profile code can be set to NONE for subbasins where in-channel routing is not needed. Otherwise, for modelling of in-channel routing process for a network of reaches, the channel's roughness, reference discharge, as well as bed slope associated with each river reach are required.

Similarly, the HRU block is assigned the following information:

1) HRU_ID (unique positive integer);
2) Area (in km^2);
3) Elevation (in m.a.s.l.);
4) Latitude (HRU centroid), and longitude (HRU centroid);

5) Subbasin_ID (subbasin in which the HRU is located);
6) Land use class;
7) Veg class;
8) Soil profile;
9) Aquifer profile (optional);
10) Terrain class (optional);
11) Slope;
12) Aspect.

Note that Raven offers a flexible modelling platform allowing different levels of complexity in representation of the hydrologic system of the watershed. At the simplest form, a watershed can be defined as a giant subbasin/HRU [2]. In other extreme, the constructed model can be composed of thousands of HRUs with a multilayer soil system [2]. The level of complexity is mainly controlled by availability of the data as well as the objective of the project. Calculated surface runoff at HRU level is routed to the subbasin reach and from there to the downstream river segment.

4. Modelling of lakes and reservoirs in HYDROTEL and Raven

A reservoir/lake in Raven is represented at the outlet of the subbasin. The same is true for HYDROTEL in which a lake/reservoir is a distinct RHHU. Whether the waterbody is regulated (reservoir) or natural (lake) different inputs are furnished to both models. In Raven, a reservoir is identified by inputs shown in figure 3.

```
# Man-made reservoir
:Reservoir [name]
  :SubBasinID [SBID]
  :HRUID [HRUID] # optional
  :StageRelations
    [number of points (N)]
    h_1, Q_1, V_1, A_1, {U_1}
    h_2, Q_2, V_2, A_2, {U_2}
    ...
    h_N, Q_N, V_N, A_N, {U_N}
  :EndStageRelations
  :MaxCapacity {capacity, in m^3} # optional
  :SeepageParameters [K_seep] [h_ref] # optional
:EndReservoir
```

Figure 3. Inputs of a regulated reservoir in Raven [2].

SubbasinID (see Fig. 3) is the identification of the subbasin with a reservoir at its outlet; HRU ID is identification of HRU in which the reservoir is located and is only required for calculating the evaporation from the reservoir surface. If HRUID is not provided, the evaporation from the reservoir surface is assumed to be negligible. The calculation of reservoir volume, outflow, reservoir surface area as well as reservoir stage are based on stage-discharge (Q(h)), stage-volume (V(h)), and stage-area (A(h)). The reservoir's operational constraints are defined in the time series file (.rvt). The MaxCapacity (m^3) is the maximum storage capacity of the reservoir and it is an optional input. The last row is for the calculation of groundwater seepage from the reservoir which is controlled by K_seep (default value is zero) and h_ref which is the reference groundwater head. If the reservoir stage is above the groundwater stage it loses water, otherwise, it gains water.

For lakes or non-regulated reservoirs, the inputs are different as shown in figure 4:

```
# Lake-like reservoir
:Reservoir [name]
  :SubBasinID [SBID]
  :HRUID [HRUID]
  :WeirCoefficient [C]
  :CrestWidth [width [m]]
  :MaxDepth [depth [m]]
  :LakeArea [area [m2]]
  :AbsoluteCrestHeight [elevation [masl]] {optional}
:EndReservoir
```

Figure 4. Inputs of a lake-like reservoir in Raven [2].

Where SubbasinID and HRUID are the same as for regulated reservoirs. For lakes, the stage-discharge, stage-volume, and stage-area relationships are defined based on the weir formula for the prismatic lake as [2]:

$$Q(h) = \frac{2}{3}\sqrt{2gC}.L.s^{1.5}$$

$$A(h) = A$$

$$V(h) = A.(s + D)$$

D [m] is maximum lake depth, g [m2/s] is the gravitational constant, C [-] is weir coefficient (typically assumed to be 0.6), A [m^2] is constant lake area, L [m] is the crest width, s is the stage measured with reference to the crest height and can have a negative value). Figure 4 shows a schematic view of a lake conceptualized in Raven. AbosolouteCrestHeight (shown as H in Figure 4) is the crest height with reference to the stage which is an optional parameter. Note that the calculated area and volume relationship can be overridden by using: AreaStageRelation and :VolumeStageRelation commands.

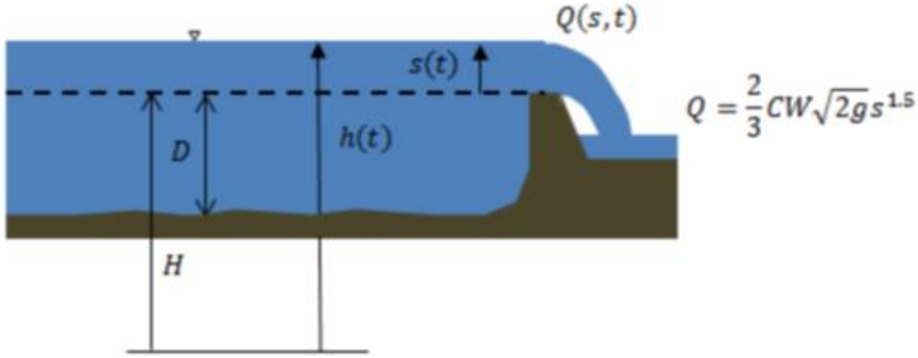

Figure 4. Conceptualization of a lake in Raven.

For dams and operational reservoirs, HYDROTEL requires coefficients v0, v1, v2, v3, v4, v5 (see Figure 5) as well as reference level (from which the water level in the reservoir is estimated) to establish volume-stage relation which is in form of a polynomial relation, the maximum and minimum acceptable discharge associated to the spillway as shown in Figure 5.

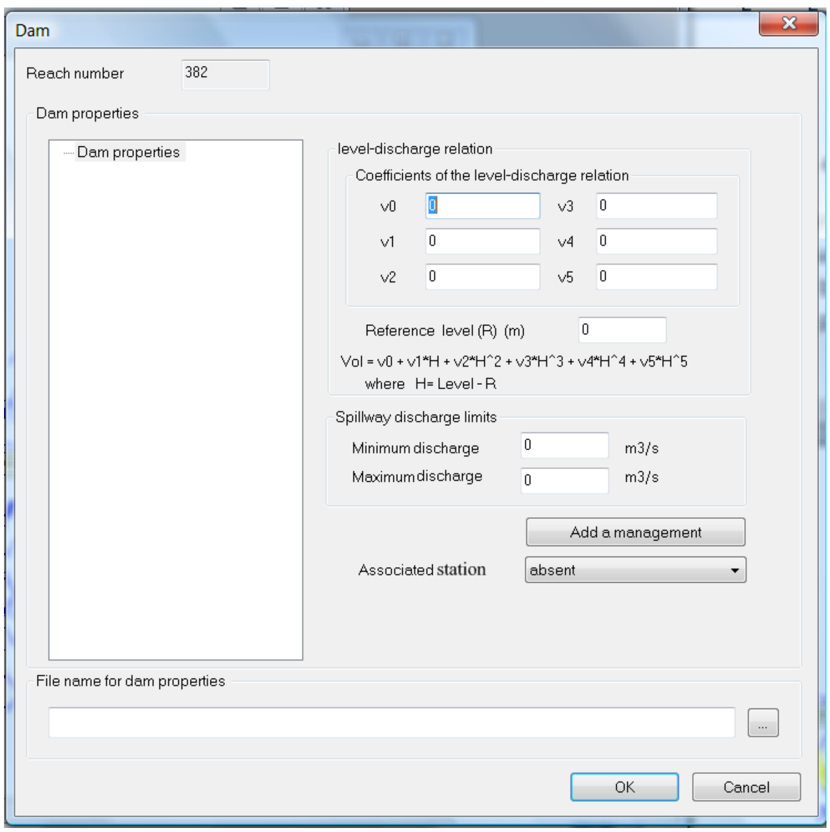

Figure 5. Dialog box for parameterization of an operational dam in Hydrotel.

In addition to the reservoir's general parameters, depending on the type of management of the dam, a new set of inputs must be furnished. There are two types of management available in the model: 1) Target level 2) Level-discharge relation. In case the Target level management option is selected, the first input is the date and hour at which there will be a management change shown in Figure 6. The user may also furnish the number of hours between operations as well as information required for the definition of target level as a function of time. In Target-level operation, it is assumed that the operation of the spillway is in a way that the reservoir reaches to the target level as soon as possible.

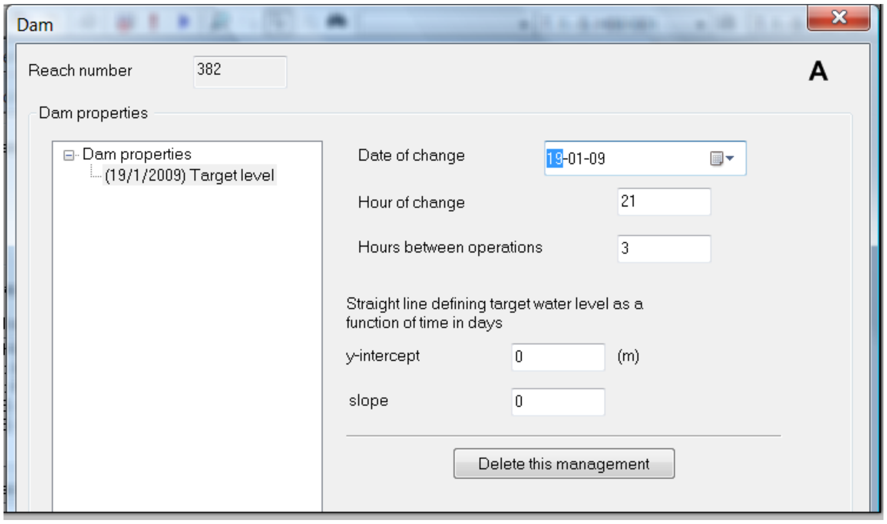

Figure 6. Dialog box for data input for the Target-level management option [1].

For the case of level-discharge management option, as in the Target-level case, the user must furnish the date and time (Figure 7) at which the management changes as well as information on linear relations defining the maximum and minimum critical levels as a function of time. This approach considers constraints of minimum and maximum water level as well as maximum and minimum spillway discharges.

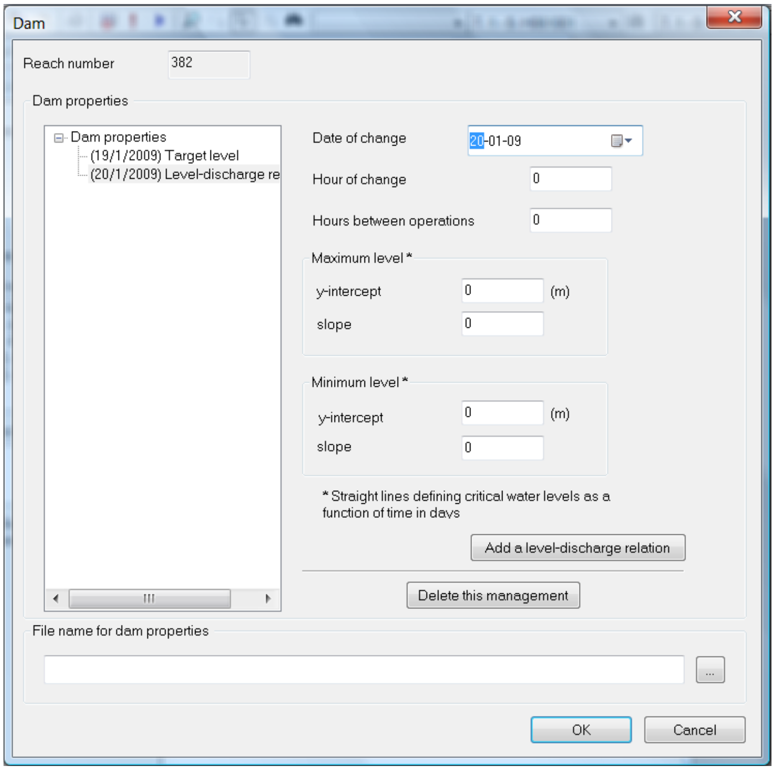

Figure 7. Dialog box for data input for the level-discharge management option [1].

In addition to the parameters related to each management option, the user has to furnish the parameters of the discharge-stage of the spillway which is based on polynomial relation shown in Figure 8.

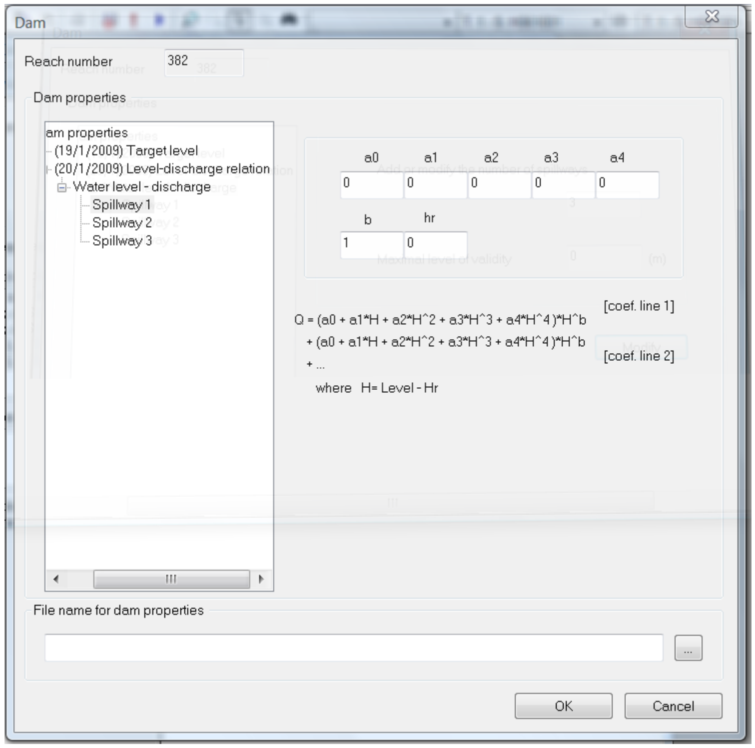

Figure 8. Dialog box for input required for establishing discharge-stage relations of the spillway.

In Hydrotel, the volume-discharge relationship of a lake is defined as:

$$Q(h) = Ch^k$$

where C = $(g * L)^{1/2}$ (for a rectangular section), g = 9,8 m/s, L [m] is lake outlet width and K = 1,5.

5. Conversion of geophysical (subbasin/HRU) inputs of HYDROTEL to Raven

As discussed in sections 2 and 3, the geospatial outputs of HYDROTEL (created by PHYSITEL) in their original form are not compatible with Raven's input format. For example, a river reach in HYDROTEL is associated with either two RHHUs (for river reaches with inflow from the upstream reaches) or three RHHUs (for headwater reaches). The total number of river reaches, therefore, are not the same as the total RHHUs in the watershed. This section aims to provide a step-by-step of the developed methodology for preparation of geospatial inputs of Raven using PHYSITEL inputs/outputs. The Jupyter Notebook report of the Python script developed for this purpose is presented at the end of in this section and is available in following GitHub repository: https://github.com/Bijan55699/Hydrotel_to_Raven/tree/for_RavenPy

The first step in the conversion process is to create the subbasin map by merging the RHHUs associated with a river reach and assigning the same ID number as for the corresponding reach. In this way, for example, the river reach with the ID number 6 corresponds to subbasin with the ID number 6. In case an RHHU is a lake, the process is the same with the exception of reach length which will be assigned zero (no in-channel routing will be performed for the subbasin with zero river length). Once the new subbasin map is created, the downstream ID of each subbasin is determined using the upstream/downstream node numbering file created by PHYSITEL. If node number *n* is downstream for the subbasin *i* and upstream for the subbasin *j*, the subbasin *j* is a downstream of the subbasin *i*.

The second step in the conversion process would be the delineation of HRU feature classes that is required in Raven. To this aim, the soil type, land use, maps are intersected so that the feature classes with unique combinations of these two inputs are created. Depending on the size of the watershed and heterogeneity of soil map and land use classes, this process may result in a very large number of HRUs complicating the execution process. To alleviate this issue, an HRU aggregation process was developed so that for each subbasin, the landuse classes covering the area less than the percentage specified by the user (for example 10%) will be merged with the dominant landuse type of that subbasin. This process, can reduce the number of HRUs so that the computational process becomes reasonable. Larger the threshold value, more landuse classes will be aggregated at the subbasin scale.

Other required information for each HRU such as elevation, latitude, longitude, would also be added to the aggregated HRU map. The HRU map is then intersected by the new subbasin map in order to extract the subbasin ID to which an HRU belongs. An important consideration in the conversion process is parameterization of the lake HRUs. This includes lake average depth, surface area, and total lake volume. As this information does not exist in PHYSITEL/HYDROTEL structure, we use the HYDROLAKES database to parameterize the lakes of a watershed. Note that the HYDROLAKES covers only lakes with surface area equal to or grater than 10ha (hectares).

6. References

1. Fortin, J.-P. and Royer, A. (2004) Le modèle hydrologique HYDROTEL – Bases théoriques. INRS-ETE, Québec, Canada.
2. Craig, J.R., and the Raven Development Team, Raven user's and developer's manual (Version 3.0.1), URL: http://raven.uwaterloo.ca/ (Accessed on 4 Jan, 2021).
3. Neitsch, S.L.; Williams, J.R.; Arnold, J.G.; Kiniry, J.R. Soil and Water Assessment Tool Theoretical Documentation Version 2009; TR-406; Texas A&M University System: College Station, TX, USA, 2011.
4. Rawls, W.J. and D.L. Brakensiek(1989). Estimation of soil water retention and hydraulic properties, in: Unsaturated flow in hydrologic modeling, Theory and practice, H.J. Morel-Seytoux (ed). Kluwer Academic Pub. p. 275-300.
5. M. L. Messager, B. Lehner, G. Grill, I. Nedeva, and O. Schmitt, "Estimating the volume and age of water stored in global lakes using a geo-statistical approach," *Nat. Commun.*, vol. 7, no. 1, p. 13603, Dec. 2016, doi: 10.1038/ncomms13603.

Delineation of subbasin map

September 28, 2021

0.1 1. This script intends to prepare the subbasin attributes of Raven hydrological modelling platform using the Physitel inputs/outputs. The script runs on Python version 3.8 and relies heaviliy on geopandas library for geospatial proecsses.

0.1.1 Section 0: import libraries

[ ]:
```
import pandas as pd
import scipy.io as sio
import shutil,os
import geopandas as gpd
from geopandas.tools import sjoin
from rasterstats import zonal_stats
```

0.1.2 Section 1: Read inputs

[ ]:
```
Troncon_path = r'C:
↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\DEH\INFO_TRONCON.mat' # the␣
↪projet database
data = sio.loadmat(Troncon_path, struct_as_record=False, squeeze_me=True)
region_name = data['SLSO_TRONCON']
size = region_name.shape[0]
```

[ ]:
```
# make a copy of the project directory
df = []
for i in range(size):
    rec = region_name[i]
    df.append([rec.NOEUD_AVAL.NUMERO,rec.NOEUD_AMONT.NUMERO,rec.NO_TRONCON,rec.
↪TYPE_NO,rec.LONGUEUR,rec.LARGEUR,rec.UHRH_ASSOCIES,rec.C_MANNING,rec.
↪PENTE_MOYENNE,rec.SUPERFICIE_DRAINEE])
Troncon_info= pd.DataFrame(df,columns =␣
↪['NODE_AVAL','NODE_AMONT','SubId','TYPE_NO','RivLength','BnkfWidth','ASSOCI_UHRH','Ch_n','R

pathtoDirectory = r"C:
↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\DEH\MG24HA\SLSO_MG24HA_2020\physitel"
workspace = os.path.join(pathtoDirectory+ "\HRU")
shutil.copytree(pathtoDirectory,workspace)
Troncon_info.head()
```

0.1.3 Section2: Add subbasin id (SubId) to UHRH shapefile

[ ]:
```
uhrh_fpth = os.path.join(workspace,"uhrh"+ "." + "shp") # The uhrh shape file␣
↪created by Physitel
uhrh = gpd.read_file(uhrh_fpth)
uhrh['SubId'] = 0

Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'Ch_n'] = 0.
Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'BnkfWidth'] = 0.

i=0
for i in range(size):
    a = Troncon_info['ASSOCI_UHRH'][i]
    id = Troncon_info['SubId'][i]
#     print ('writing subbasin :', i )
    if type(a) is int:
        aa = [a]
        st = len(aa)
        stt = st-1
        dict = {i: aa[i] for i in range(0, len(aa))}
    else:
        al = a.tolist()
        st = len(al)  # number of UHRH associated with current reach
        stt = st - 1
        #create a temporary dictionary
        dict = {i: al[i] for i in range(0, len(al))}
    for j in range(st):
        for index, row in uhrh.iterrows():
            if uhrh.loc[index,'ident'] in dict.values():
                uhrh.loc[index,'SubId'] = id

os.chdir(workspace)
uhrh.to_file('uhrh_diss.shp')
```

0.1.4 Section 3: Merge the UHRHs based on SubId field. The number of feature classes in the output file should be same sa number of river reaches (Troncons)

[ ]:
```
uhrh_diss = gpd.read_file(os.path.join(workspace,"uhrh_diss"+ "." + "shp"))
uhrh_dissolve = uhrh_diss.dissolve(by='SubId')
uhrh_dissolve.reset_index(inplace=True)
uhrh_dissolve['BasArea'] = uhrh_dissolve.area  # calculating the Area (m2) of␣
↪each subbasin

os.chdir(workspace)
uhrh_dissolve.to_file('uhrh_dissolve.shp')

# step3: finding the downstream subwatershed ID associated with each uhrh
```

```python
Troncon_info['DowSubId']=-1
for i in range(size):
    naval = Troncon_info['NODE_AVAL'][i]
    for j in range(size):
        namont= Troncon_info['NODE_AMONT'][j]
        id = Troncon_info['SubId'][j]
        if type(namont) is int:
            nal = [namont]
        else:
            nal = namont.tolist()
        if naval in nal: # if naval (downstream node) for reach i is upstream␣
 ↪node for reach j, then reach j is downstream reach i
            Troncon_info.loc[i, 'DowSubId'] = id

Troncon_info['Has_Gauge'] = (Troncon_info['DowSubId'] == -1).astype(int) ␣
 ↪#create a boolean indicator to set 1 for gauged
#subwatershed and 0 for others
Troncon_info['BkfDepth'] = 0.13 * (Troncon_info['SA_Up'] ** 0.4) # taken from␣
 ↪equation 10 in paper Fossey et. al., 2015
Troncon_info['Lake_Cat']= 0
Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'Lake_Cat'] = 1

# TO BE DISCUSSED:
# In Troncon_info dataframe, the outlet has the DowSubId of -1, which can be␣
 ↪the number of gauge.
```

0.1.5 Section4: Parametrization lake features using the HyLAKES database

[ ]:
```python
pth = r"C:
 ↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\HydroLAKES_polys_v10_shp"
pth2 = os.path.join(pth,"HydroLAKES_polys_v10_Canada2"+ "." + "shp") # The␣
 ↪clipped version of HyLAKES for Canada
HyLAKES_Canada = gpd.read_file(pth2)

pth3 = os.path.join(workspace,"lacs"+ "." + "shp") # The lake shape file␣
 ↪created by Physitel
Hydrotel_lakes = gpd.read_file(pth3)
join_lakes_attr = sjoin(HyLAKES_Canada, Hydrotel_lakes, how="right")

# Dealing with cases where there are two lakes in subwatershed whereas only one␣
 ↪lake is identified in the lacs.shp shapefile
# by Hydrotel
#finding rows with similar ident (uhrh) value
repeatd_ident = join_lakes_attr[join_lakes_attr.duplicated(subset = ['ident'],␣
 ↪keep= False)]
```

```
repeatd_ident.reset_index(level=0,inplace = True)

diss_repeat = repeatd_ident.dissolve(by = 'index',aggfunc='sum')

diss_repeat['Depth_avg'] = diss_repeat['Vol_total']/diss_repeat['Lake_area']␣
↪#recalculating lake average depth
diss_repeat['Lake_type'] = 1


#replacing this to the repeatd_ident dataframe

join_lakes_attr = join_lakes_attr.drop(diss_repeat.index)
lake_final = (pd.concat([join_lakes_attr,diss_repeat])).sort_index()
lake_final = lake_final.drop(['index_left'], axis=1)

os.chdir(workspace)
lake_final.to_file('lake_final.shp')

# Intersecting with uhrh_dissolve to find the SubId of each lake
lake_sub = sjoin(lake_final,uhrh_dissolve,how = 'right',op='within')

lake_sub['Lake_Area'] = lake_sub['Lake_area'] * 1000000.  # To convert the area␣
↪in Km2 in HydroLAKES database to m2
lake_sub['LakeVol'] = lake_sub['Vol_total'] / 1000.  # To convert the volume in␣
↪MCM in HydroLAKES database to km3
lake_sub['LakeDepth'] = lake_sub['Depth_avg']

os.chdir(workspace)
lake_sub.to_file('uhrh_with_lake.shp')
```

0.1.6 Section5: Add the downstream ID to the shapefile of the created subbasin shapefile (uhrh_diss.shp)

[ ]:
```
pth4 = os.path.join(workspace,"uhrh_with_lake"+ "." + "shp")
subbasin = gpd.read_file(pth4)

subbasin['DowSubId'] = 0
subbasin['RivLength'] = 0.0
subbasin['BkfWidth'] = 0.0
subbasin['BkfDepth'] = 0.0
subbasin['Has_Gauge'] = 0.0
subbasin['RivSlope'] = 0.0
subbasin['Ch_n'] = 0.0
subbasin['FloodP_n'] = 0.0
subbasin['Lake_Cat'] = 0
#Lake data from HydroLAKES database
```

```
subbasin['HyLakeId'] = subbasin['Hylak_id']

j=0
for index, row in subbasin.iterrows():
    if index > subbasin.index[-1]:
        break
    subbasin.loc[index,'DowSubId'] = Troncon_info['DowSubId'][j]
    subbasin.loc[index,'RivLength'] = Troncon_info['RivLength'][j]
    subbasin.loc[index,'BkfDepth'] = Troncon_info['BkfDepth'][j]
    subbasin.loc[index,'BkfWidth'] = Troncon_info['BnkfWidth'][j]
    subbasin.loc[index,'Has_Gauge'] = Troncon_info['Has_Gauge'][j]
    subbasin.loc[index,'RivSlope'] = Troncon_info['RivSlope'][j]
    subbasin.loc[index,'Ch_n'] = Troncon_info['Ch_n'][j]
    subbasin.loc[index,'FloodP_n'] = Troncon_info['Ch_n'][j]     # to be␣
 ↪discussed
    subbasin.loc[index,'Lake_Cat'] = Troncon_info['Lake_Cat'][j]
    j = j+1

os.chdir(workspace)
subbasin.to_file('subbasin.shp')
```

0.1.7 Section6: Calculating BasSlope,BasAspect,, and Mean_Elev of subbasin features

[ ]:
```
# Slope

os.chdir(workspace)
cmd_slope = 'gdaldem  slope altitude.tif slope.tif -compute_edges'
os.system(cmd_slope)
# slope must be between 0 to 60 degree (http://hydrology.uwaterloo.ca/
 ↪basinmaker/data/resources/attribute_tables_20210429.pdf)

# Aspect
os.chdir(workspace)
cmd_aspect = 'gdaldem  aspect altitude.tif aspect.tif -trigonometric␣
 ↪-compute_edges'
os.system(cmd_aspect)


# loop over the subbasin features and adding the mean elevation, mean aspect␣
 ↪and
ss = os.path.join(workspace,"slope"+ "." + "tif") # The lake shape file created␣
 ↪by Physitel
pth5 = os.path.join(workspace,"subbasin"+ "." + "shp") # The lake shape file␣
 ↪created by Physitel
subbasin = gpd.read_file(pth5)
```

```
subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= ss,
            stats=['mean']
        )
    ),
    how='left'
)

subbasin.loc[subbasin['mean'] < 0 , "mean"] = 0

subbasin['BasSlope'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

#aspect
aa = os.path.join(workspace,"aspect"+ "." + "tif") # The lake shape file␣
 ↪created by Physitel

subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= aa,
            stats=['mean']
        )
    ),
    how='left'
)

subbasin['BasAspect'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

#elevation

ee = os.path.join(workspace,"altitude"+ "." + "tif") # The lake shape file␣
 ↪created by Physitel

subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= ee,
            stats=['mean']
        )
    ),
```

```
    how='left'
)

subbasin['MeanElev'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

# cleaning: removing irrelevant attributes
subbasin['Lake_Area'] = subbasin['Lake_Are_1']

subbasin = subbasin.
 ↪drop(['Lake_name','Country','Continent','Poly_src','Grand_id','Lake_area','Shore_len','Shor
                          ␣
 ↪'Vol_res','Vol_src','Depth_avg','Dis_avg','Res_time','Elevation','Slope_100',
                          ␣
 ↪'Wshd_area','Pour_long','Pour_lat','Shape_Leng','Shape_Area','ident_x','ident_y','Hylak_id'
 ↪axis=1)
```

0.1.8 Section7: Writing final subbasin map

[ ]:
```
os.chdir(workspace)
subbasin.to_file('subbasin_final.shp')


for fname in os.listdir(workspace):
    if fname.startswith("lake_final"):
        os.remove(os.path.join(workspace, fname))
```

Delineation of HRU map

September 28, 2021

0.1 This script delineates the HRU map of a watershed using the Physitel inputs/outputs. Note that the subbasin map created using previous script will be used for identificaction of subbasin ID of the HRUs.

0.1.1 Section 0: Import libraries

[ ]:
```
import pandas as pd
import os
import geopandas as gpd
from geopandas.tools import sjoin
from rasterstats import zonal_stats
import rasterio
from rasterio.features import shapes
```

0.1.2 Section 1: Read inputs

[ ]:
```
pathtoDirectory = r"C:
↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\DEH\MG24HA\SLSO_MG24HA_2020\physitel"
workspace = os.path.join(pathtoDirectory+ "\HRU")

#Read the subbasin, land use, and soil maps

subwshd_pth = os.path.join(workspace,"subbasin_final"+"."+"shp") #subwatershed␣
↪map created by Hydrotel_Raven_V2 script
lu_raster = os.path.join(workspace,"occupation_sol"+"."+"tif") #Lu map in raster
soil_raster = os.path.join(workspace,"type_sol"+"."+"tif") #soil type map in␣
↪raster
lake = os.path.join(workspace,"lacs"+"."+"shp") #lake map
altitude = os.path.join(workspace,"altitude"+ "." + "tif") # The altitude␣
↪raster map
aspect = os.path.join(workspace,"aspect"+ "." + "tif") # The aspect raster map
slope = os.path.join(workspace,"slope"+ "." + "tif") # The slope raster map
```

0.1.3 Section 2: Polygonizing the raster maps (land use, soil) for further processing

[ ]:
```
# land use map

with rasterio.Env():
    with rasterio.open(lu_raster) as src:
        lu = src.read(1) # first band
        mask = src.dataset_mask()
        ras_crs = src.crs
        results = (
        {'properties': {'LU_ID': v}, 'geometry': s}
        for i, (s, v)
        in enumerate(
            shapes(lu, mask=mask, transform=src.transform)))

geoms = list(results)
lu_poly  = gpd.GeoDataFrame.from_features(geoms,crs=ras_crs)

os.chdir(workspace)
lu_poly.to_file('lu_test.shp')

# soil

with rasterio.Env():
    with rasterio.open(soil_raster) as src:
        soil = src.read(1) # first band
        mask = src.dataset_mask()
        ras_crs = src.crs
        results = (
        {'properties': {'soil_ID': v}, 'geometry': s}
        for i, (s, v)
        in enumerate(
            shapes(soil, mask=mask, transform=src.transform)))

geoms = list(results)
soil_poly  = gpd.GeoDataFrame.from_features(geoms,crs=ras_crs)

os.chdir(workspace)
soil_poly.to_file('soil_type.shp')
```

0.1.4 Section 3: overlaying the soil and land use map to create the HRU map

[ ]:
```
hru1 = gpd.overlay(lu_poly, soil_poly, how='intersection')
```

0.1.5 Section 4: Finding the major land use and soil class in lake polygons

[ ]:
```python
lake_poly = gpd.read_file(lake)
# Union lake polygon with HRU map (a lake is a unique HRU)

hru2 = gpd.overlay(lake_poly, hru1, how='intersection')
hru3 = hru2.dissolve(by='ident',aggfunc = 'first') #aggregate all the polygons␣
↪that are lake in the hru
hru4 = gpd.overlay(hru1, hru3, how='symmetric_difference')
hru5 = gpd.overlay(hru4, hru3, how='union')

hru6 = sjoin(lake_poly,hru5,how = 'right',op='within')
# os.chdir(workspace)
# hru6.to_file('hru6.shp')

hru6['LU'] = 0
hru6['SOIL'] = 0
for index, row in hru6.iterrows():
    if hru6.loc[index,'ident'] < 0 and hru6.loc[index,'ident'] != 'nan':
        hru6.loc[index,'LU'] =2
        hru6.loc[index,'SOIL'] = hru6.loc[index,'soil_ID']
    else:
        hru6.loc[index,'LU'] = hru6.loc[index,'LU_ID_1']
        hru6.loc[index,'SOIL'] = hru6.loc[index,'soil_ID_1']


hru7 = hru6.
↪drop(['index_left','LU_ID_1','LU_ID_2','soil_ID_1','soil_ID_2','soil_ID','LU_ID'],␣
↪axis=1)
```

0.1.6 Section 5: Identifying major land use and soil classes in each subbasin

[ ]:
```python
subbasin = gpd.read_file(subwshd_pth)
# land use
subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= lu_raster,
            stats=['majority']
        )
    ),
    how='left'
)

subbasin['LU_major'] = subbasin['majority'].astype(int)
subbasin = subbasin.drop(['majority'], axis=1)
```

```
#soil
subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= soil_raster,
            stats=['majority']
        )
    ),
    how='left'
)

subbasin['soil_major'] = subbasin['majority'].astype(int)
subbasin = subbasin.drop(['majority'], axis=1)
```

0.1.7 Section 6: Identity: intersecct the hru7 with subbasin to cut the HRU's on subbasin limits

[ ]:
```
hru8 = gpd.overlay(hru7, subbasin, how='intersection')
```

0.1.8 Section 7: # calculate area of each land use class within each subbasin: This will be needed to dissolve small (based on a threshold given by user) land use classes by the major one

[ ]:
```
subbasin ['LUID_1'] = 0.  # No data
subbasin ['LUID_2'] = 0.  # Water
subbasin ['LUID_3'] = 0.  # Bare soil
subbasin ['LUID_4'] = 0.  # deciduous forest
subbasin ['LUID_5'] = 0.  # agricultur
subbasin ['LUID_6'] = 0.  # coniferous forest
subbasin ['LUID_7'] = 0.  # impermeable surface
subbasin ['LUID_8'] = 0.  # peatland
subbasin ['LUID_9'] = 0.  # wetland

for index, row in subbasin.iterrows():
    sub = subbasin.loc[subbasin['SubId'] == subbasin['SubId'][index]] # selects␣
↪the subbasin a in the subbasin map
    intersection = gpd.overlay(sub, lu_poly, how='intersection') # intersection␣
↪operation
    intersection['area'] = intersection.area  # the area of each row in the␣
↪intersection
    subbasin.loc[index,'LUID_2'] = intersection.loc[intersection['LU_ID'] ==␣
↪2,'area'].sum()
    subbasin.loc[index,'LUID_3'] = intersection.loc[intersection['LU_ID'] ==␣
↪3,'area'].sum()
```

```
    subbasin.loc[index,'LUID_4'] = intersection.loc[intersection['LU_ID'] ==␣
↪4,'area'].sum()
    subbasin.loc[index,'LUID_5'] = intersection.loc[intersection['LU_ID'] ==␣
↪5,'area'].sum()
    subbasin.loc[index,'LUID_6'] = intersection.loc[intersection['LU_ID'] ==␣
↪6,'area'].sum()
    subbasin.loc[index,'LUID_7'] = intersection.loc[intersection['LU_ID'] ==␣
↪7,'area'].sum()
    subbasin.loc[index,'LUID_8'] = intersection.loc[intersection['LU_ID'] ==␣
↪8,'area'].sum()
    subbasin.loc[index,'LUID_9'] = intersection.loc[intersection['LU_ID'] ==␣
↪9,'area'].sum()

os.chdir(workspace)
subbasin.to_file('subbasin2.shp')
```

0.1.9 Section 8: Defining the threshold and creating the aggregated HRU map

[ ]:
```
c = hru8.columns
merge = gpd.GeoDataFrame(columns = c,crs = hru8.crs)
hru8['LU_agg'] = hru8['LU']
Threshold = 5 # This is the threshold (%) based on which the land use classes␣
↪covering smaller than that will be aggreagted to the major land use class
for i in range(subbasin.shape[0]):
    subwsh_number = i + 1
    sub = subbasin.loc[subbasin['SubId'] == subwsh_number] # selects the␣
↪subbasin a in the subbasin map
    lu_major = sub['LU_major'][i]
    area_total = sub['BasArea'][i]
    perc_lu2 = (sub['LUID_2'][i]/area_total)*100.
    perc_lu3 = (sub['LUID_3'][i]/area_total)*100.
    perc_lu4 = (sub['LUID_4'][i]/area_total)*100.
    perc_lu5 = (sub['LUID_5'][i]/area_total)*100.
    perc_lu6 = (sub['LUID_6'][i]/area_total)*100.
    perc_lu7 = (sub['LUID_7'][i]/area_total)*100.
    perc_lu8 = (sub['LUID_8'][i]/area_total)*100.
    perc_lu9 = (sub['LUID_9'][i]/area_total)*100.
    # hru check and modifications
    hru_temp = hru8.loc[hru8['SubId'] == subwsh_number] # selects the subbasin␣
↪a in the subbasin map
    for index, row in hru_temp.iterrows():
        #check the percentage with threshold
        # if (row.LU==2):
        #     if (perc_lu2<=Threshold and perc_lu2>0 and row.Lake_Cat==0):
        #         hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==3):
```

```
            if (perc_lu3<=Threshold and perc_lu3>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==4):
            if (perc_lu4<=Threshold and perc_lu4>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==5):
            if (perc_lu5<=Threshold and perc_lu5>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==6):
            if (perc_lu6<=Threshold and perc_lu6>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==7):
            if (perc_lu7<=Threshold and perc_lu7>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==8):
            if (perc_lu8<=Threshold and perc_lu8>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
        if (row.LU==9):
            if (perc_lu9<=Threshold and perc_lu9>0 and lu_major!=2):
                hru_temp.loc[index,'LU_agg'] = lu_major
    # import pdb; pdb.set_trace()
    temp = hru_temp.dissolve(by = ["LU_agg","SOIL"], as_index = False)
    temp2 = merge.append(temp)
    merge = temp2
```

0.1.10 Section 9: Adding SOIL_PROF and LAND_USE_CODE fields (string) to the hru map

[ ]:
```
merge = merge.reset_index()
merge['SOIL'] = merge['SOIL'].astype(int)
merge['LU'] = merge['LU'].astype(int)
merge['Soil_ID'] = merge['SOIL']
merge['Landuse_ID'] = merge['LU_agg']

st = {1:'sand', 2:'loamy_sand', 3:'sandy_loam', 4:'loam', 5:'silt_loam', 6:
 ↪'silt', 7:'sandy_clay_loam', 8:'clay_loam', 9:'silty_clay_loam',
      10:'sandy_clay', 11:'silty_clay', 12:'clay'}  # to be confirned with DEH

merge['SOIL_PROF'] = merge['Soil_ID'].map(st)


lu_codes = st = {1:'No_data', 2:'Water', 3:'bare_soil', 4:'deciduous_forest', 5:
 ↪'agriculture', 6:'coniferous_forest', 7:'impermeable_surface', 8:'peatland',␣
 ↪9:'wetland'}  # to be confirned with DEH

merge['LAND_USE_CODE'] = merge['LU_agg'].map(lu_codes)
```

```
os.chdir(workspace)
merge.to_file('hru10.shp')
```

0.1.11 Section10: Add latitude,longitude, HRU ID, Slope, aspect, and Elevation to the HRU feature class

[ ]:
```
# adding HRU_ID
merge['HRU_ID'] = 0
j=1
for index, row in merge.iterrows():
    merge.loc[index,'HRU_ID'] = j
    j = j+1


# calculating the ara of each HRU polygon in m2
merge['HRU_Area'] = merge.area

# adding mean elevation of each HRU

#elevation

merge = merge.join(
    pd.DataFrame(
        zonal_stats(
            vectors=merge['geometry'],
            raster= altitude,
            stats=['mean']
        )
    ),
    how='left'
)

merge['HRU_E_mean'] = merge['mean']
merge = merge.drop(['mean'], axis=1)


#aspect

merge = merge.join(
    pd.DataFrame(
        zonal_stats(
            vectors=merge['geometry'],
            raster= aspect,
            stats=['mean']
        )
    ),
```

```
    how='left'
)

merge['HRU_A_mean'] = merge['mean']
merge = merge.drop(['mean'], axis=1)


# adding mean slope


# pth5 = os.path.join(workspace,"subbasin"+ "." + "shp") # The lake shape file␣
 ↪created by Physitel
# subbasin = gpd.read_file(pth5)

merge = merge.join(
    pd.DataFrame(
        zonal_stats(
            vectors=merge['geometry'],
            raster= slope,
            stats=['mean']
        )
    ),
    how='left'
)

merge.loc[merge['mean'] < 0 , "mean"] = 0

merge['HRU_S_mean'] = merge['mean']
merge = merge.drop(['mean'], axis=1)

# adding latitude, longitude

merge['HRU_CenX'] = merge.centroid.x
merge['HRU_CenY'] = merge.centroid.y



merge = merge.
 ↪drop(['index_left','ident','index','OBJECTID','LU_major','soil_major','LU_agg','LU','SOIL']
 ↪axis=1)


os.chdir(workspace)
merge.to_file('hru_final.shp')
```