Prasanna Dahal

Dr. David Tarboton & Dr. David Maidment

CEE 6400

04 December 2015

Preparing input for the TOPKAPI model

TOPKAPI (TOPographic Kinematic Approximation and Integration) is TOP model based physically-based fully-distributed rainfall-runoff model deriving from the integration in space of the kinematic wave model. The geometry of the catchment is described by a lattice of cells over which the equations are integrated to lead to a cascade of nonlinear reservoirs (Liu and Todini, 2002). This project aims at building a tool to assist preparing input data for the model. There are five python script in the tool which have to be run from Step 1 through Step 5, with each script requiring some inputs. The output of the tool will be set of rasters associated with physical properties of the watershed.

Progea.net [www.progea.net] is distributor or TOPKAPI model, and its explanation to the utility of the model is "Beside subsurface, overland and channel flow, it includes components representing infiltration, percolation, evapo-transpiration and snowmelt. It can be applied at increasing spatial scales without losing model and parameter physical interpretation Topkapi allows study of evolution of all the hydrological state variables of the catchment, such as rainfall, temperature, evapotranspiration, soil moisture conditions, snow accumulation and runoff generation". The model is foreseen to be suitable for land-use and climate change impact assessment; for extreme flood analysis, given the possibility of its extension to ungauged

catchments (Liu and Todini, 2002). The parameter for the TOPKAPI model can be obtained from digital elevation maps, soil maps and vegetation or land use maps in terms of slope, soil permeability, roughness and topology after a sequence of GIS processing. For this reason, preparing input data for TOPKAPI model was chosen as a semester project. An open source version of TOPKAPI model known as pyTOPKAPI was selected as the variant for study, because it was free and usable across Windows, Mac OS or LINUX platform. PyTOPAPI is developed by Theo Vischel & Scott Sinclair. This can be downloaded from https://github.com/sahg/PyTOPKAPI.git

Although a model with various functionality, it has only been used in handful of cases in the US. On analysing potential causes of its limited use, one major reason identified was the lack of input data preparing tools available for data that are specific to the US. PyTOPKAPI has its own module to prepare its input parameters, but that approach requires input rasters that are not common in the US, or are much coarser than similar data prepared by the US. The diagrammatic representation of its recommended input rasters to prepare input parameter is shown in figure 1. For example, pyTOPKAPI can prepare its parameters if given GLCC (Global Land Cover Characterization) land cover dataset, soil type raster as defined by SIRI (1987) and soil texture raster as defined by Midgley et. al. (1984). Although GLCC is available for the US, it is not regularly updated and is much coarser that NLCD (National Land Cover Dataset), the land cover rasters adopted in the US. Thus, this project uses NLCD land cover data to prepare the input parameters. Also, for Soil type and soil texture, US has its own corresponding data management practice- called SSURGO (Soil Survey Geographic database). This project uses SSURGO data to prepare much of the soil related parameter used in the model.

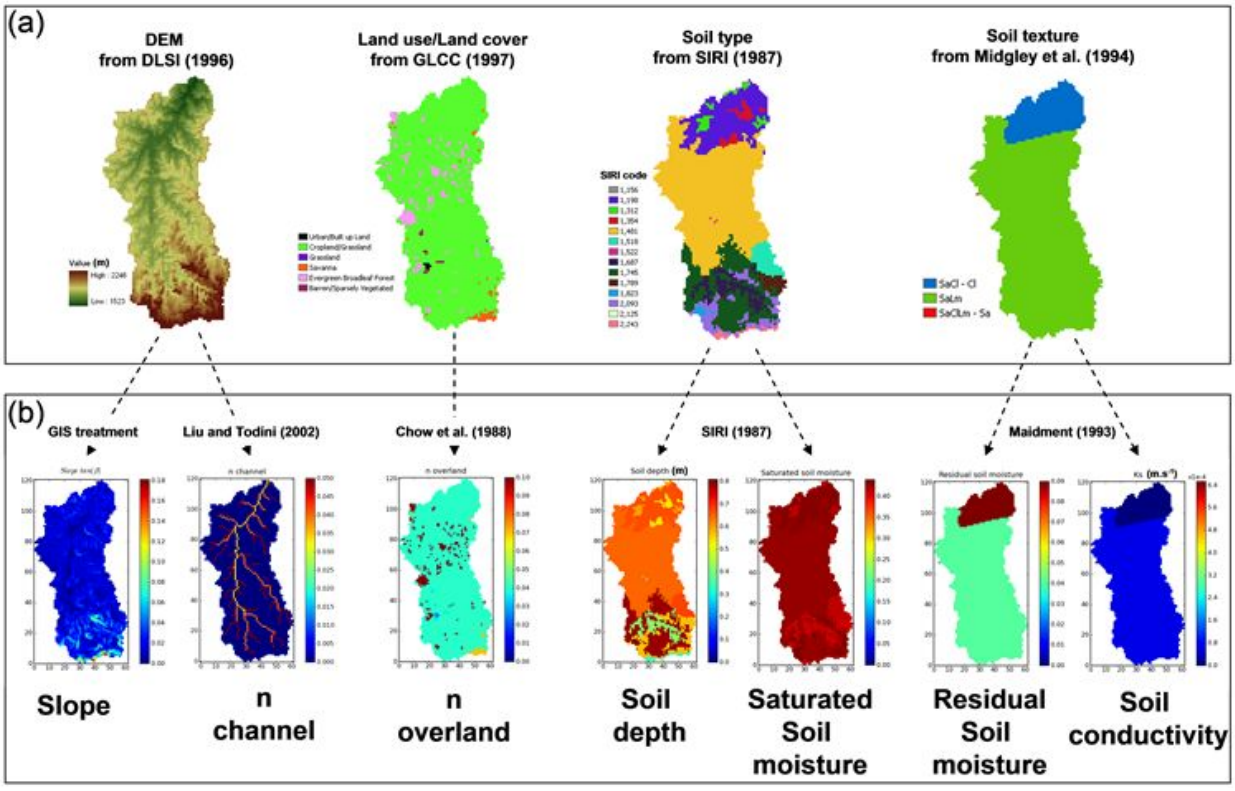


Figure 1 a)Catchment data map b)From Catchment data to TOPKAPI model parameter

Source: Viscel & Sinclair et. al.

As outputs from the project, there will be a set of rasters corresponding the parameters:

- Slope (in degree) of overland area and the channel
- Soil Depth
- Saturated hydraulic conductivity
- Residual soil moisture content
- Initial Saturation of soil reservoir
- Water content
- overland Slope of overland
- Saturated soil moisture content

- Manning's coefficient for overland
- Manning's coefficient for channel
- Bubbling pressure
- Pore size distribution

METHODOLOGY

Study Area: The 1688.4 square km large study area is Provo river located at East-South of the Salt Lake City, as shown in figure 2 below. It is a small watershed taken for demonstrative purpose only. The tool is supposed to create the output rasters irrespective of any location the user wants to use it for.

Figure 2 Provo river, the project study area, to the south-east of Salt Lake City

Data Download and preparation: Most of the input parameters are derived from DEM, land cover data and soil data from SSURGO. They are obtained from:

- DEM data (Downloaded using ArcGIS web services, http://landscape2.arcgis.com/arcgis),
- NLCD land cover data (Downloaded using ArcGIS web services, http://landscape2.arcgis.com/arcgis)
- SSURGO soil data (Downloaded from http://websoilsurvey.nrcs.usda.gov or https://gdg.sc.egov.usda.gov)

*SSURGO stands for Soil Survey Geographic database

For a particular watershed, the input parameters listed above are prepared using Python scripts bundled together in an ArcGIS toolbox, named "AutoPTPK2", as shown in figure 2. The step by step guides to use the tool is provided in Appendix C. There are four python scripts are in the toolbox.

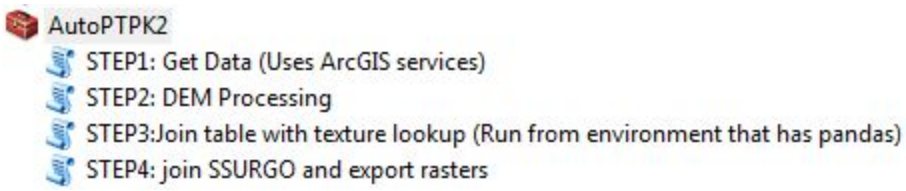


Figure 2 A snippet of the AutoPTK tool, the result of the term project

The first 'STEP1: Get Data (Uses ArcGIS service)' takes as its inputs the ESRI polygon shapefile of the watershed of our interest, as well as ESRI online account username and password. This scripts also needs user to input a folder (or preferably a geodatabase) where the downloaded file will be stored. As the outputs, the scripts downloads DEM and NLCD (2011)

data for the region specified and stores them on the output location specified earlier. The python code of the script is attached in Appendix A-1.

Processing the DEM: The second script of the tool 'STEP2: DEM Processing' takes the DEM (Digital Elevation Model) downloaded in the first step as and a geodatabase or a folder to store the files it creates. This step processes the input DEM it to come up with a few of the required input parameter files such as a)flow direction for each cell in the watershed, b)Strahler's stream order raster for stream, and c) the stream itself. A part of the workflow of the DEM processing is demonstrated as Model Builder workflow diagram in figure 3 below. The diamond figure in blue represents inputs to the DEM processing step. The inputs are DEM, and the watershed boundary as you can see in the figure below. The outputs are hexagon in grey with red borders. They are Flow direction raster (based on D-8 method), stream order as described by Strahler (1957). The yellow rectangles represent ArcGIS tools, while green oval represent data used by and produced by those tools. Note that this figure however, is just for demonstrative purpose as python script was used in the step and not model builder. The python code of the script is attached in Appendix A-2.

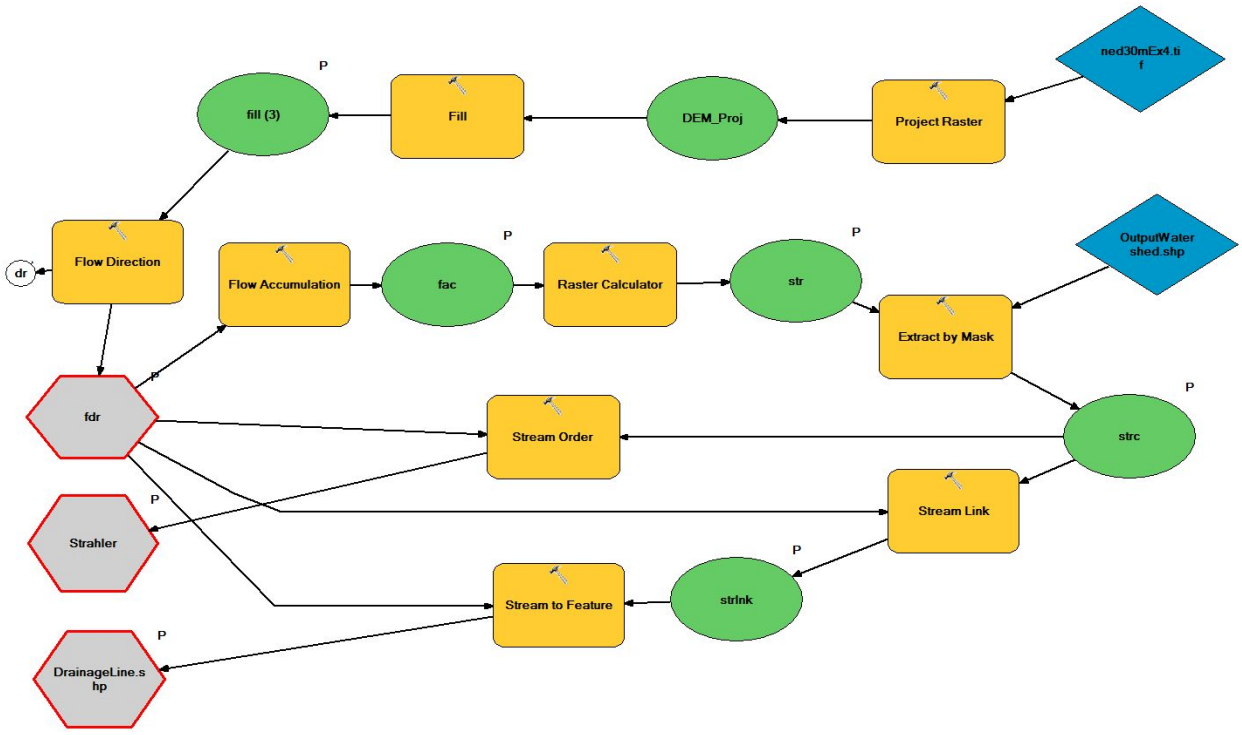


Figure 3 Workflow of DEM processing used in the second script of the tool

The Strahler order raster was reclassified using a table by Liu and Todini (2002) of Strahler order to Manning's coefficient n for a channel in the watershed , shown in Table 1:

Table1: Strahler Order to Manning's n relation used to during reclassification.

Strahler's Stream Order	**Manning's coefficient n**
1	0.05
2	0.04
3	0.035
4	0.03
5	0.03
6	0.025

Similar to reclassifying Strahler order raster to get Manning's n for channel , NLCD data downloaded earlier is also reclassified to Manning's n for overland portion of the watershed based on the look up table by Kalyanapu et. al.(2009) shown below.

Table2: NLCD land cover values Vs Manning's coefficient n

NLCD Land Cover Code	**Manning's N**
21	0.0404
22	0.0678
23	0.0678
24	0.0404
31	0.0113
41	0.36
42	0.32
43	0.4
52	0.4
71	0.368
81	0.325
90	0.086
95	0.1825

The ArcGIS tools used to achieve this were:

- Fill Flow direction
- Flow Accumulation
- Raster Calculation

- Extract by mask
- StreamLink
- Stream To Feature
- Stream Order (Strahler)
- Join table
- Feature to raster

ArcGIS did not accept values below 1 to be classified. Hence, in this project, reclassification was done by multiplying the Manning's value by 10000. Then, the raster-calculator tool was used, and the value for the raster was divided by 10000 to get the final raster with Manning's n.

The third script 'STEP3:Join table with texture lookup (Run from environment that has pandas)' uses SSURGO database, extracts/cleans data, located in tables in the 'tabular' folder of each SSURGO given folder. The created table is used to create parameter raster for python script, which will be discussed later.

A brief introduction to SSURGO database and sample calculations: SSURGO data needs to be downloaded for the study region from web (https://gdg.sc.egov.usda.gov/ or http://websoilsurvey.nrcs.usda.gov). One watershed region can have many folders where the data are present. The path to this folder collection of such folders is passed on as an input to this script. Each of those folder downloaded, when unzipped, will have two folders 'Spatial' and 'Tabular' and some files in it.

The SSURGO divides our region into multiple polygon called map-units. The shapefile containing the region is located in the spatial folder under the name soilmu_a_xxxx (for example soilmu_a_ut611). Each of those map units have different component of soil in it, whose property

is available along with what percentage of a particular component of soil is present in each map units, but their spatial information is not. So the properties for each component will have to be averaged out based on percentage given to get one value for one map unit. However, not all properties of soil are based on component of soil present in a map unit. That is because each component has one or more layers of soil. Most properties of soil (for example soil type, porosity etc.) is associated with each of the layers. Thus, first, one representative value for one component needs to be calculated by taking weighted average based on height of the layer. The calculation example for finding Ksat (saturated hydraulic conductivity) for one map-unit (map unit 1) is explained in the Figure 4 below. Notice in the upper portion of the figure where ksat value was calculated by taking weighted average based on height of the soil layer. The value obtained, 2.95 is value of one of four components present in the map unit we are concerned. To calculated Ksat representing all of map-unit 1, another weighted average based on component percentage will need to be done, as shown in the lower part of the figure 4.

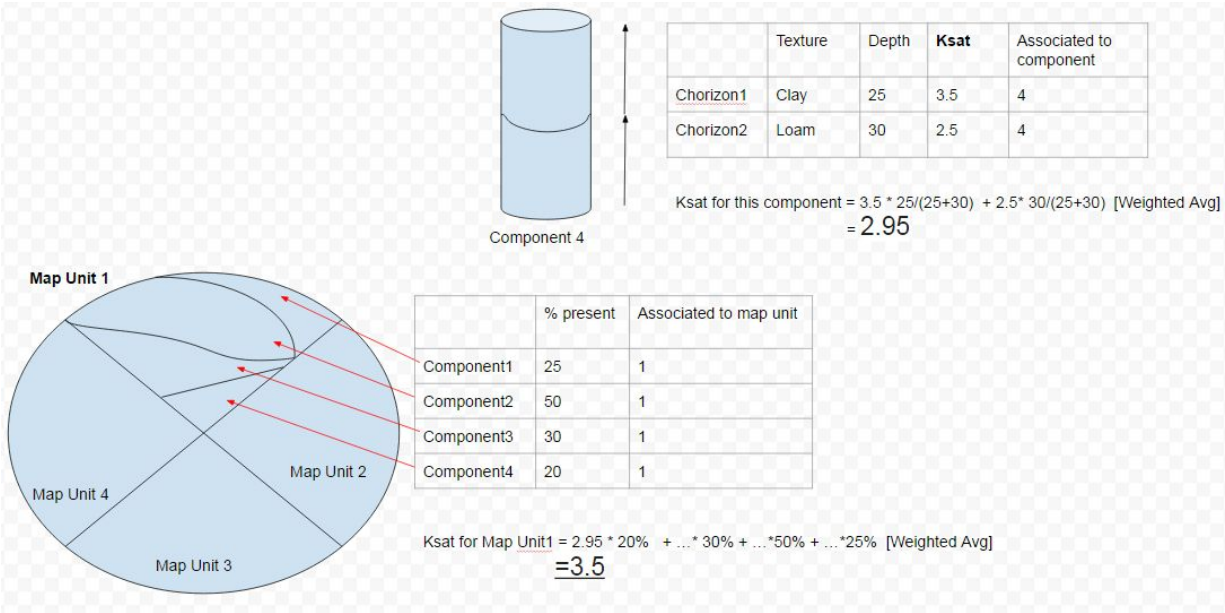


Figure 4 Schematic diagram of SSURGO representation of soil properties, and their calculations

The exact same calculation process is done by the script 'STEP3:Join table with texture lookup (to be run from environment that has pandas)'. This script calculates one representing value for one component of the soil first, taking weighted average based on height of the soil layer. Then, it uses the values obtained in similar way for all the component, and takes weighted average based on component percentage to get one value for one map-unit. Same step is repeated for all the soil properties.

SSURGO table extraction and joining: Ahead of all these calculations, data contained in the SSURGO, which are organised as relational tables, will have to be managed first. There are different tables in the folder 'tabular' of the data downloaded. The one containing data of each map unit is named muaggatt.txt (same as table muaggatt in access database), of all the components present is named comp.txt (same as table component in access database), of each soil layer is named chorizon.txt (same as table chorizon in access database), of each texture group is named chtexgrp.txt (same as table Chorizon Texture group in access database) and the one containing texture information is named chtextur.txt (same as table Chorizon Texture in access database).

Because this is a SSURGO is a relational database, its tables can be joint based on primary key for one table to foreign key of another table to create a large table that maps all the required soil property to map units. The tables, as you can see in the entity relationship diagram in figure 5 below, was joint in the following order: first joining "Chorizon Texture" table to "Chorixon Texture Group" based on key Chtxtgrpkey. Then the combined table was joint to 'Chorizon", which was then joint to "Component" and then finally all these was joint to

"Muaggatt". After all these joins, the final table looks like as shown in Figure 6, with each item in "Chorizon" or in "Chroizon Texture" table mapped to one Map unit.

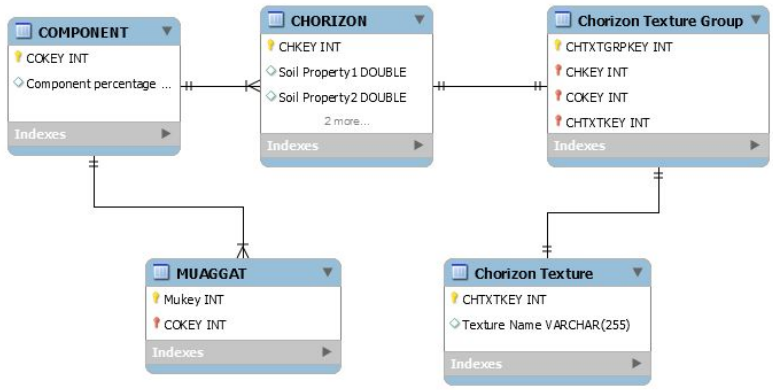


Figure 5 Entity relationship diagram of SSURGO tables used in the project

textureName	CHtxtgrpKEY	CHTXTKEY	CHKEY	AvaWaterCon	MUKEY	ComponentPercent	MajorComponent	COKEY	TopDepth	BottomDepth	HorizonDepth	ksat_r	dbthirdbar_r
Silty clay loam	69519129	70749678	35401183	0	482190	50	Yes	12395040	15	28		0.42	1.3
Silty clay loam	69519130	70750735	35401184	0	482190	50	Yes	12395040	28	43		0.42	1.28
	69519131	70750736	35401185	0	482190	50	Yes	12395040	43	102		0.42	
Silty clay loam	69519132	70750737	35401186	0	482191	85	Yes	12395042	0	33		0.42	1.18
Clay	69519146	70750838	35401200	0	482192	5	No	12395048	102	152		0.42	1.13
Silty clay loam	69519147	70750190	35401201	0	482192	85	Yes	12395050	0	33		0.42	1.2
Clay	69519148	70749680	35401202	0	482192	85	Yes	12395050	33	99		0.42	1.2
Silty clay	69519149	70750191	35401203	0	482192	85	Yes	12395050	99	152		0.01	1.2
Silty clay	69519162	70750553	35401216	0	482193	20	Yes	12395054	99	152		0.42	1.18
Loam	69519163	70749682	35401217	0	482194	85	Yes	12395055	0	20		14	1.38
Loam	69519164	70750194	35401218	0	482194	85	Yes	12395055	20	30		14	1.38
Loam	69519165	70750739	35401219	0	482194	85	Yes	12395055	30	43		14	1.35
	69519166	70750740	35401220	0	482194	85	Yes	12395055	43	69		0	
Clay	69519179	70750478	35401233	0	482196	5	No	12395064	33	99		0.42	1.18
Silty clay	69519180	70749780	35401234	0	482196	5	No	12395064	99	152		0.42	1.18
Loam	69519181	70750382	35401235	0	482197	85	Yes	12395065	0	15		4	1.28
Silt loam	69519182	70750383	35401236	0	482197	85	Yes	12395065	15	30		4	1.25
Silt loam	69519196	70750384	35401248	0	482199	90	Yes	12395072	15	30		4	1.25
Silty clay loam	69519197	70750385	35401249	0	482199	90	Yes	12395072	30	46		0.42	1.23
Silty clay	69519198	70750386	35401250	0	482199	90	Yes	12395072	46	79		0.42	1.4
Silty clay	69519199	70750387	35401251	0	482199	90	Yes	12395072	79	102		0.42	1.4
Fine sandy loam	69519213	70750389	35401263	0	482201	90	Yes	12395079	0	25		14	1.4
Fine sandy loam	69519214	70750390	35401264	0	482201	90	Yes	12395079	25	33		14	1.4
Clay loam	69519215	70750391	35401265	0	482201	90	Yes	12395079	33	76		0.42	1.28
Loam	69519216	70750206	35401266	0	482201	90	Yes	12395079	76	97		4	1.5
Very fine sandy loam	69519229	70750392	35401279	0	482203	90	Yes	12395086	97	122		14	1.38
Sandy loam	69519230	70750393	35401280	0	482203	90	Yes	12395086	122	152		14	1.5

Figure 6 SSURGO extracted table by joint the mentioned three tables

In the table "Chorizon Texture", the soil type in named a texture name, like Loam, Sand, Clay, Loamy Sand etc. Each of these is a unique name of soil type present in each layer. All type of soil can be grouped into one of these 11 soil type, as shown from a soil triangle in the figure 7.

And for each of these soil type, tables such as one presented by Rawls et. al. (1982) as shown in figure 8 provides useful approximation of many soil properties.

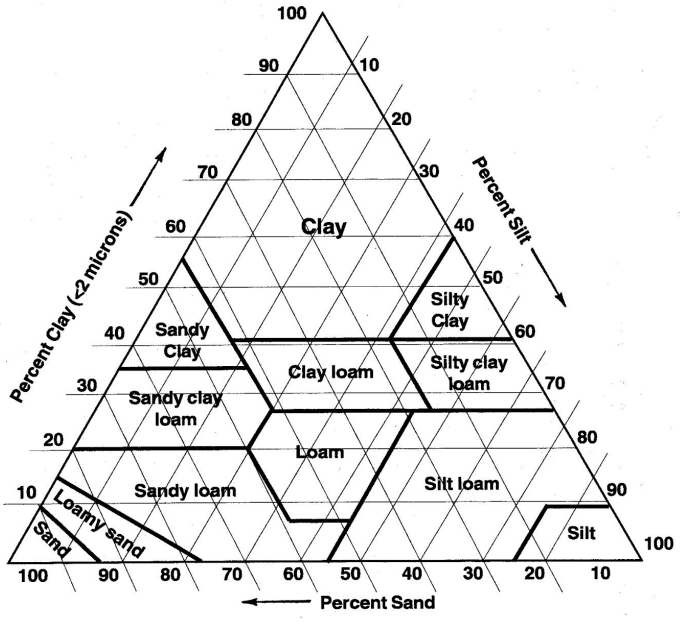


Figure 7 A typical soil triangle

Texture Class	Sample Size	Total Porosity (ϕ)	Residual Water Content	Effective Porosity (ϕ_e)	Bubbling Pressure (h_b)		Pore Size Distribution (λ)		Water Retained at -33 kPa	Water Retained at -1500 kPa	Saturated Hydraulic Conductivity (K_s)
		cm^3/cm^3	cm^3/cm^3	cm^3/cm^3	Arithmetic cm	Geometric** cm	Arithmetic	Geometric**	cm^3/cm^3	cm^3/cm^3	cm/hr
Sand	762	0.437* (.374-.500)	0.020 (.001-.039)	0.417 (.354-.480)	15.98 (.24-31.72)	7.26 (1.36-38.74)	0.694 (.298-1.090)	0.592 (.334-1.051)	0.091 (.018-.164)	0.033 (.007-.059)	21.00
Loamy Sand	338	0.437 (.368-.506)	0.035 (.003-.067)	0.401 (.329-.473)	20.58 (-4.04-45.20)	8.69 (1.80-41.85)	0.553 (.234-.872)	0.474 (.271-.827)	0.125 (.060-.190)	0.055 (.019-.091)	6.11
Sandy Loam	666	0.453 (.351-.555)	0.041 (.024-.106)	0.412 (.283-.541)	30.20 (-3.61-64.01)	14.66 (3.45-62.24)	0.378 (.140-.616)	0.322 (.186-.558)	0.207 (.126-.288)	0.095 (.031-.159)	2.59
Loam	383	0.463 (.375-.551)	0.027 (.020-.074)	0.434 (.334-.534)	40.12 (-20.07-100.3)	11.15 (1.63-76.40)	0.252 (.086-.418)	0.220 (.137-.355)	0.270 (.195-.345)	0.117 (.069-.165)	1.32
Silt Loam	1206	0.501 (.420-.582)	0.015 (.028-.058)	0.486 (.394-.578)	50.87 (-7.68-109.4)	20.76 (3.58-120.4)	0.234 (.105-.363)	0.211 (.136-.326)	0.330 (.258-.402)	0.133 (.078-.188)	0.68
Sandy Clay Loam	498	0.398 (.332-.464)	0.068 (.001-.137)	0.330 (.235-.425)	59.41 (-4.62-123.4)	28.08 (5.57-141.5)	0.319 (.079-.889)	0.250 (.125-.502)	0.255 (.186-.324)	0.148 (.085-.211)	4.3
Clay Loam	366	0.464 (.409-.519)	0.075 (.024-.174)	0.390 (.279-.501)	56.43 (-11.44-124.3)	25.89 (5.80-115.7)	0.242 (.070-.414)	0.194 (.100-.377)	0.318 (.250-.386)	0.197 (.115-.279)	.23
Silty Clay Loam	689	0.471 (.418-.524)	0.040 (.038-.118)	0.432 (.347-.517)	70.33 (-3.26-143.9)	32.56 (6.68-158.7)	0.177 (.039-.315)	0.151 (.090-.253)	0.366 (.304-.428)	0.208 (.138-.278)	.15
Sandy Clay	45	0.430 (.370-.490)	0.109 (.013-.205)	0.321 (.201-.435)	79.48 (-20.15-179.1)	29.17 (4.96-171.6)	0.223 (.048-.398)	0.168 (.078-.364)	0.339 (.245-.433)	0.239 (.162-.316)	.12
Silty Clay	127	0.479 (.425-.533)	0.56 (.024-.136)	0.423 (.334-.512)	76.54 (-6.47-159.6)	34.19 (7.04-166.2)	0.150 (.040-.260)	0.127 (.074-.219)	0.387 (.332-.442)	0.250 (.193-.307)	.09
Clay	291	0.475 (.427-.523)	0.090 (.015-.195)	0.385 (.269-.501)	85.60 (-4.92-176.1)	37.30 (7.43-187.2)	0.165 (0.31-.293)	0.131 (.068-.253)	0.396 (.326-.466)	0.272 (.208-.336)	.06

*First line is the mean value, and second is one standard deviation about the mean.
**Antilog of the log mean.

Figure 8 Texture class versus soil property map given by Rawls et al. (1982)

Hence, after altering units to the ones we need, and using must the mean value from the table, a table as shown in figure 9 was prepared. This lookup table is located in the zipped folder containing the tool and is named "GREENAMPT_LOOKUPTABLE.csv". This is the file that has to be passed on as an input to run the script by the user. Other lookup table with additional parameters or different units for different soil properties too can be passed. But those tables need to have headers in them, with 'textureName' as the name for texture groups field. The scripts then maps soil properties for each soil map unit.

textureName	Porosit	EffectivePorosi	ResidualWaterConte	BubblingPressure_arithm	BubblingPressur	PoreSizeDistributio	PoreSizeDistributio	WaterRetained	WaterRetaine	Ks
Sand	0.437	0.417	0.02	159.8	72.6	0.694	0.592	0.091	0.033	5.83
Loamy sand	0.437	0.401	0.035	205.8	86.9	0.553	0.474	0.125	0.055	1.7
Sandy loam	0.453	0.415	0.041	302	146.6	0.378	0.322	0.207	0.095	0.72
Silt loam	0.463	0.434	0.027	401.2	111.5	0.252	0.22	0.27	0.117	0.37
Loam	0.501	0.486	0.015	508.7	207.6	0.234	0.211	0.33	0.133	0.19
Sandy clay loam	0.398	0.33	0.068	594.1	280.8	0.319	0.25	0.255	0.148	0.12
Silty clay loam	0.464	0.39	0.075	564.3	258.9	0.242	0.194	0.318	0.197	0.06
Clay loam	0.474	0.432	0.04	703.3	325.6	0.177	0.151	0.366	0.208	0.04
Sandy clay	0.43	0.321	0.109	794.8	291.7	0.223	0.168	0.339	0.239	0.03
Silty clay	0.479	0.423	0.56	765.4	341.9	0.15	0.127	0.387	0.25	0.03
Clay	0.475	0.385	0.09	856	373	0.165	0.131	0.396	0.272	0.02

Figure 9 Rawls's lookup table with units changed as required by TOPKAPI model

The last script of the tool, 'STEP4: join SSURGO and export rasters' The table obtained after combining tables of SSURGO (as shown in figure 6) is joint to table mapping soil type to its properties, i.e. table shown in figure 9. Then, weighted average procedure is carried out to finally obtain a table that has all the weighted averaged soil properties for each map unit. The table is then saved on user's hard drive under the name "MUKEY-Vs-Values.csv in the each of the SSURGO folders. The table looks like the one shown in figure 10 below.

PoreSizeDistributio	MUKEY	ksat_r_WtAvg	Ks_WtAvg	dbthirdbar_r_WtAv	dbfifteenbar_	Porosity_WtAvg	EffectivePorosity	BubblingPressure_	PoreSizeDistribution
0.127525	658420	7.26887	0.252975	0.6373695652		0.21385	0.202025	76.71	0.127525
0.210846022	658421	16.354134419	0.534143207	0.9453488372		0.2650020808	0.2401406365	89.3914810282	0.210846022
0.213625	658429	14.749465	0.456075	1.035		0.36175	0.345725	130.035	0.213625
0.2010267031	658437	4.9909127315	0.237169451	1.1546668002		0.4067955657	0.369264431	186.390234501	0.2010267031
0.0329173872	658440	1.914317588	0.06403704	0.1701778963		0.0546577949	0.0494900167	26.489996993	0.0329173872
0.3108	658470	33.8744	0.9175	1.2625		0.3568	0.3271	96.385	0.3108
0.1516	658471	9.880025	0.4343	0.58		0.1788	0.1639	49.685	0.1516
0.191807772	658472	6.3242954404	0.329212021	0.9519689119		0.3164891192	0.2826300518	138.465336788	0.191807772
0.1346	658476	10.712035842	0.4602	0.4577684211		0.1319	0.121	29.055	0.1346
0.411883114	659480	26.2711056743	2.4110619847	1.4583634868		0.4504851974	0.4158309759	133.911222588	0.411883114
0.1706323444	659481	7.83345	0.339622589	0.8205894771		0.2809760455	0.2630034434	109.532975208	0.1706323444
0.1869127096	659482	2.2988518752	0.21748324	1.175142885		0.4018880117	0.3686392788	200.844152047	0.1869127096
0.2029691479	659506	8.133242216	0.274508932	1.2335799872		0.4315601884	0.402215844	202.531067904	0.2029691479
0.2535685031	659507	9.1342880888	0.374498707	1.40052347		0.4639554234	0.4276000566	209.081626185	0.2535685031
0.161802636	659520	2.8299112545	0.1151870545	0.9122369069		0.3083859812	0.2726044867	178.440295845	0.161802636
0.2378549452	659531	6.6196657785	0.323051458	1.3544741087		0.4369694688	0.4005125401	204.19933407	0.2378549452
0.1016727273	659553	0.7153190909	0.027858182	0.8565909091		0.3070318182	0.2728227273	207.292272727	0.1016727273
0.2619182796	659738	13.69088	1.058613602	1.3881182796		0.4739032258	0.4444064516	223.200716846	0.2619182796
0.2150366167	659741	6.9554632946	0.577324386	1.0573525905		0.3511606858	0.3286421929	133.582773733	0.2150366167
0.1539607543	659743	2.3985993032	0.126509593	1.1847273378		0.4241599554	0.3709988799	282.603302957	0.1539607543
0.1837296492	659745	6.7037753589	0.2740391107	0.8890980048		0.3910259871	0.3503812045	235.698161004	0.1837296492

Figure 10 MUKEY-Vs-Values.csv created with soil values for each texture class

In the next step of the work, the table as shown in figure 10 above, is joint to the attribute table of spatial file of ssurgo named soilmu_a_xxxxx. The joining of fields of shapefile in arcGIS is shown in the figure 11 below. So now, all the map units have been mapped to one representative soil properties for the region. The new field added in the attribute table is used to convert the feature to the raster files. Hence, there will be as many rasters as there are properties described in the lookup table 10 above.

soilmu_a_ut603

FID	Shape	AREASYMBOL	SPATIALVER	MUSYM	MUKEY	MUKEY	EffectivePorosity_WtAvg_x	Ksat_r_WtAvg	Ks_WtAvg	h2oBarOneThird_WtAvg	h20Bar15_WtAvg	Porosity_WtAvg
594	Polygon	UT603	3	MIA	482784	482784	0.486	4	0.65	23.418182	9.009091	0.501
2760	Polygon	UT603	3	MIA	482784	482784	0.486	4	0.65	23.418182	9.009091	0.501
2786	Polygon	UT603	3	MIA	482784	482784	0.486	4	0.65	23.418182	9.009091	0.501
2956	Polygon	UT603	3	MIA	482784	482784	0.486	4	0.65	23.418182	9.009091	0.501
3217	Polygon	UT603	3	MIA	482784	482784	0.486	4	0.65	23.418182	9.009091	0.501
68	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
192	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
288	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
289	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
404	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
654	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
785	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
997	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1148	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1212	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1213	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1221	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1224	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1508	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1522	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1524	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1547	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1582	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1609	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1827	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1830	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1843	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1890	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1892	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1895	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
1980	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2018	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2072	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2104	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2179	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2256	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2280	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2290	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2300	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2464	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2582	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2637	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726
2671	Polygon	UT603	3	Rs	482818	482818	0.483045	3.609118	0.621086	17.333191	8.240493	0.499726

Figure 11 Table obtained in step3 merged to shapefile based on MUKEY

RESULTS

The result of the term project is that an ArcGIS tool is prepared. The use of the tool is simple, same drag and drop procedure as other inbuilt GIS tools. This tool produces rasters that represents properties of the watershed using DEM, NLCD rasters, lookup tables and SSURGO database. The rasters can now be used as input for pyTOPKAPI model. This might also be used for some other hydrological models.

The running of the tool on the study region produced the rasters as expected.The produced rasters are shown below from figure 12 through figure 19. The produced rasters will be saved in the output folder that the user gives during running the results. The rasters are in TIFF format. If there are more than one SSURGO folders, these results might need to be joined using.

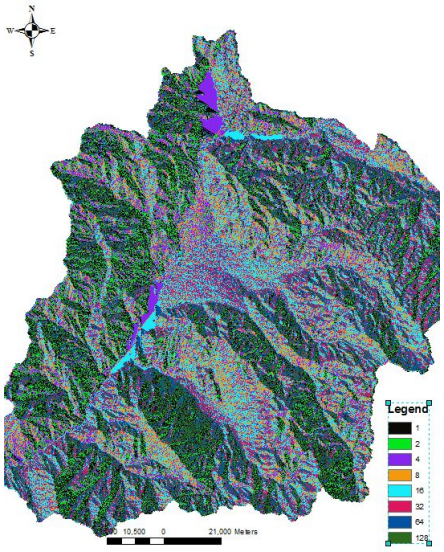


Figure 12

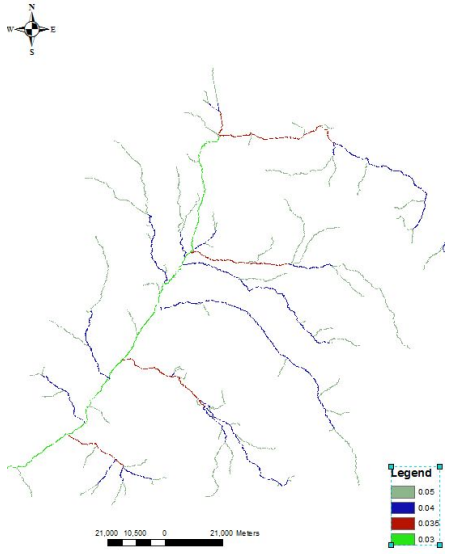


Figure 13

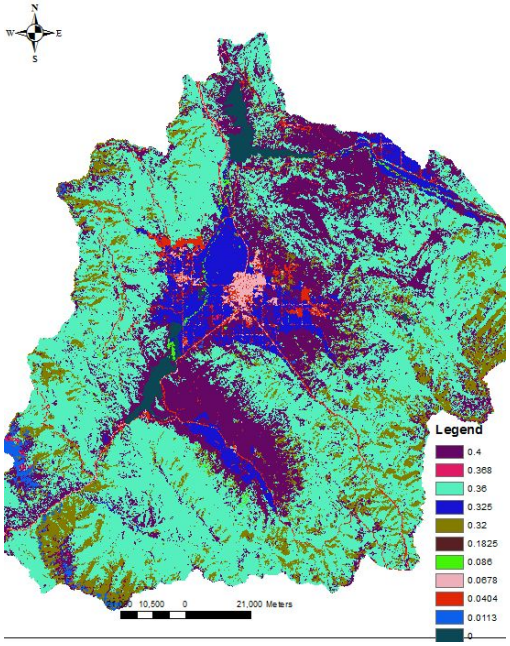


Figure 14

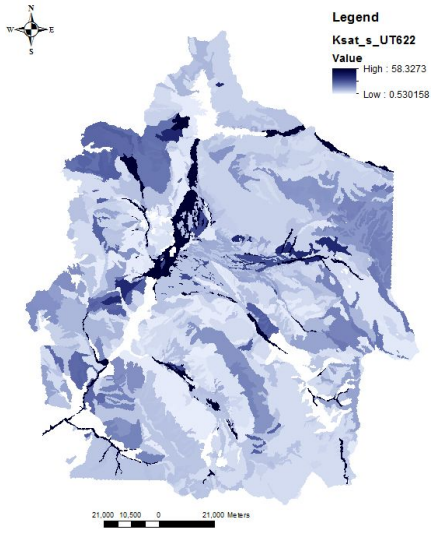


Figure 15

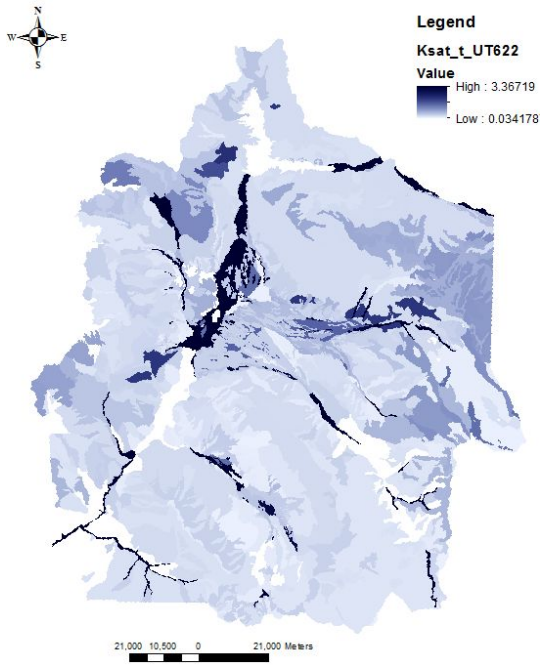


Figure 16

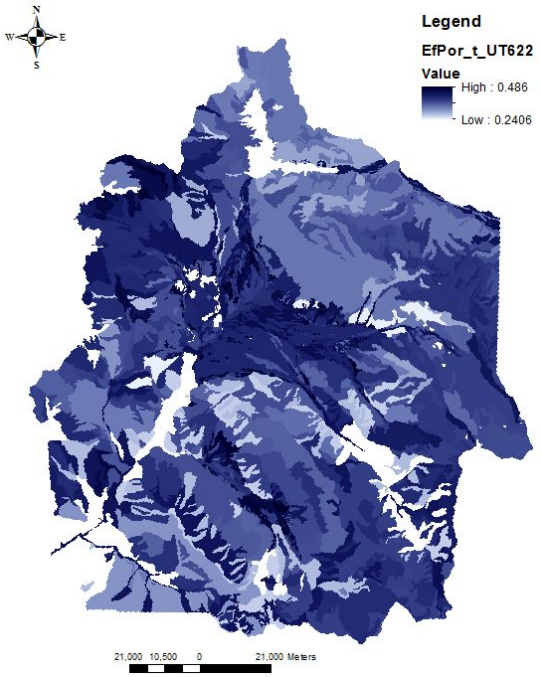


Figure 17

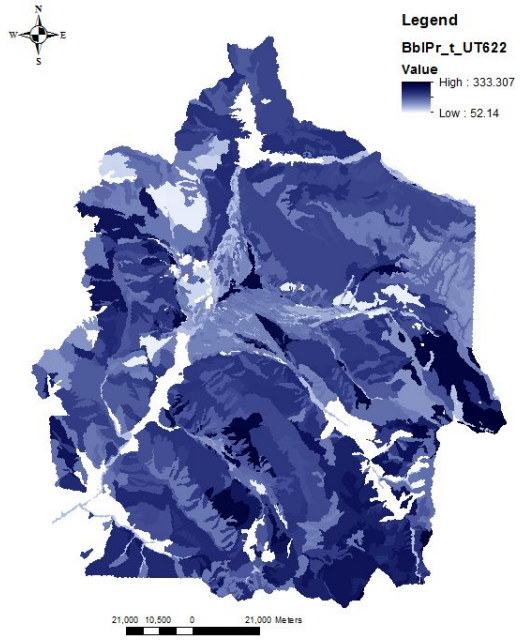


Figure 18

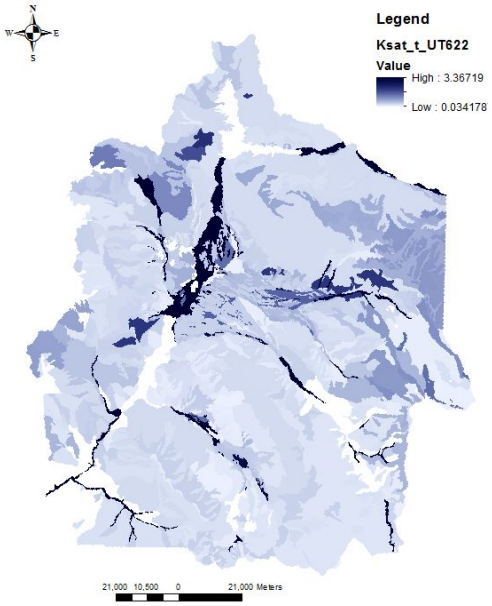


Figure 19

CONCLUSION

Thus, the term project was completed. Much of ArcGIS tools was learnt. An ArcGIS tool was prepared that makes input parameters preparation work easier for pyTOPKAPI model. It might also be useful to other hydrological models. However, the third script may not work from ArcGIS environment as it may not have python modules pandas installed on it. If so, the script will have to be run from an environment that has pandas installed in it.

FUTURE WORKS

The tool makes use of 'pandas' module which is not present in python that comes with ArcGIS installation. This creates nuisance to user, having to run a portion of the tool from another environment, or worse get stuck because of not having pandas in your computer. Hence rewriting the similar code avoiding pandas module might make life easier for those who want to use this tool.

The tool can be further simplified for use. There are still a few drag and drops too many to get the tasks done. With relatively many steps required to get the task done, there is more chances for error. That could be minimized by slightly better design of the tool.

Also, the tool can be expanded to extract more data from SSURGO database, as core program is already written to accomplish the task.

Works Cited

GitHub,. (2015). sahg/PyTOPKAPI. Retrieved 5 December 2015, from https://github.com/sahg/PyTOPKAPI.git

Kalyanapu, A. J., Burian, S. J., & McPherson, T. N. (2010). Effect of land use-based surface roughness on hydrologic model output. *Journal of Spatial Hydrology*, *9*(2).

Liu, Z., & Todini, E. (2002). *Towards a comprehensive physically-based rainfall-runoff model*. *Hydrology and Earth System Sciences Discussions*, *6*(5), 859-881.

Soil Survey Staff, Natural Resources Conservation Service, *United States Department of Agriculture. Web Soil Survey.* Available online at http://websoilsurvey.nrcs.usda.gov/. Accessed 12/04/2015

Strahler, A. N. (1957). *Quantitative analysis of watershed geomorphology*. *Civ. Eng*, *101*, 1258-1262.

Vischel, T., Pegram, G., Scott Sinclair, S., & Parak, M. (2008). *Implementation of the TOPKAPI model in South Africa: Initial results from the Liebenbergsvlei catchment.* Water Sa, 34(3), 331-342.

Appendix A

1. STEP1: Get Data (Uses ArcGIS services) code

```
import arcpy
import os
import sys
'''
Written initially by Cyndia Castro
Modified by Prasanna Dahal
'''

arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

inUsername = arcpy.GetParameterAsText(0)
inPassword = arcpy.GetParameterAsText(1)
outDir = arcpy.GetParameterAsText(2)
wshedDir = arcpy.GetParameterAsText(3)
bufferDi= arcpy.GetParameterAsText(4)

#defaulted, to make things easier
if inUsername == "": inUsername = ""
if inPassword  == "": inPassword = ""

# Set workspace environment
arcpy.env.workspace = arcpy.env.scratchWorkspace = outDir
arcpy.env.outputCoordinateSystem = arcpy.SpatialReference(102008)

# Add Data to Geodatabase
arcpy.FeatureClassToFeatureClass_conversion(wshedDir, outDir, "Boundary")
```

```
arcpy.MakeFeatureLayer_management("Boundary", "Boundary")

# Buffer
arcpy.Buffer_analysis("Boundary", "Buffer", str(bufferDi)+" Meters", "FULL", "ROUND",
"NONE", "", "PLANAR")

# Connect to ArcGIS Servers
out_folder_path = 'GIS Servers'

out_landscape = 'Landscape.ags'
server_landscape = 'https://landscape5.arcgis.com:443/arcgis/services/'

out_elevation = 'Elevation.ags'
server_elevation = 'https://elevation.arcgis.com:443/arcgis/services/'

arcpy.mapping.CreateGISServerConnectionFile("USE_GIS_SERVICES",
                                            out_folder_path,
                                            out_landscape,
                                            server_landscape,
                                            "ARCGIS_SERVER",
                                            username=inUsername,
                                            password=inPassword,
                                            save_username_password=True)

arcpy.mapping.CreateGISServerConnectionFile("USE_GIS_SERVICES",
                                            out_folder_path,
                                            out_elevation,
                                            server_elevation,
```

```
                                        "ARCGIS_SERVER",
                                        username=inUsername,
                                        password=inPassword,
                                        save_username_password=True)


# Extract Image Server Data
""" Land Use """
NLCD_ImageServer = "GIS Servers\\Landscape\\USA_NLCD_2011.ImageServer"
arcpy.MakeImageServerLayer_management(NLCD_ImageServer,"NLCD_Layer")
arcpy.gp.ExtractByMask_sa("NLCD_Layer", "Buffer", "Land_Use")


""" DEM, 30m NED """
NED30m_ImageServer = "GIS Servers\\Elevation\\NED30m.ImageServer"
arcpy.MakeImageServerLayer_management(NED30m_ImageServer, "NED30m_Layer")
arcpy.gp.ExtractByMask_sa("NED30m_Layer", "Buffer", "DEM")


#project DEM to UTM
arcpy.ProjectRaster_management(in_raster="DEM", out_raster="DEM_Prj",
out_coor_system="PROJCS['NAD_1983_2011_UTM_Zone_12N',GEOGCS['GCS_NAD_1983_2011',DATUM[
'D_NAD_1983_2011',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Greenwich',0.0
],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETER['Fals
e_Easting',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',-111
.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0],UNIT['Meter',
1.0]]", resampling_type="NEAREST", cell_size="30.9220807759341 30.922080775934",
geographic_transform="'WGS_1984_(ITRF00)_To_NAD_1983 +
WGS_1984_(ITRF08)_To_NAD_1983_2011'", Registration_Point="",
```

```
in_coor_system="PROJCS['North_America_Albers_Equal_Area_Conic',GEOGCS['GCS_North_Ameri
can_1983',DATUM['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],
PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Albers'],PARAME
TER['False_Easting',0.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',
-96.0],PARAMETER['Standard_Parallel_1',20.0],PARAMETER['Standard_Parallel_2',60.0],PAR
AMETER['Latitude_Of_Origin',40.0],UNIT['Meter',1.0]]")


#project Land Use to UTM
arcpy.ProjectRaster_management(in_raster="Land_Use", out_raster="Land_Use_Prj",
out_coor_system="PROJCS['NAD_1983_2011_UTM_Zone_12N',GEOGCS['GCS_NAD_1983_2011',DATUM[
'D_NAD_1983_2011',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Greenwich',0.0
],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETER['Fals
e_Easting',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',-111
.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0],UNIT['Meter',
1.0]]", resampling_type="NEAREST", cell_size="30.9220807759341 30.922080775934",
geographic_transform="'WGS_1984_(ITRF00)_To_NAD_1983 +
WGS_1984_(ITRF08)_To_NAD_1983_2011'", Registration_Point="",
in_coor_system="PROJCS['North_America_Albers_Equal_Area_Conic',GEOGCS['GCS_North_Ameri
can_1983',DATUM['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],
PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Albers'],PARAME
TER['False_Easting',0.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',
-96.0],PARAMETER['Standard_Parallel_1',20.0],PARAMETER['Standard_Parallel_2',60.0],PAR
AMETER['Latitude_Of_Origin',40.0],UNIT['Meter',1.0]]")
```

2. STEP2: DEM Processing

```
import arcpy
from arcpy import env
from arcpy.sa import *
```

```
arcpy.env.overwriteOutput = True

arcpy.CheckOutExtension("Spatial")

DEM = arcpy.GetParameterAsText(0)

outDir= arcpy.GetParameterAsText(1)  #output directory

Basin = arcpy.GetParameterAsText(2)  #boundary

threshold = arcpy.GetParameterAsText(3)  #Threshold for defining stream


if threshold == "": threshold = "5000"

# Set workspace environment

arcpy.env.workspace = arcpy.env.scratchWorkspace = outDir

arcpy.env.outputCoordinateSystem = arcpy.SpatialReference(102008)


fill = "fill"

fdr =  'fdr'

fac = 'fac'

strlnk = 'strlnk'

str = 'str'

strc = 'strc'

drp = 'drp'

Catchment = 'Catchment'

DrainageLine_shp = 'DrainageLine'

CatchPoly_shp =  'CatchPoly'

CatchPolyDissolve_shp = 'CatchPolyDissolve'

STRAHLER = "STRAHLER"



arcpy.gp.Fill_sa(DEM, fill, "")

arcpy.gp.FlowDirection_sa(fill, fdr, "NORMAL", drp)

arcpy.gp.FlowAccumulation_sa(fdr, fac, "", "FLOAT")
```

```
arcpy.gp.RasterCalculator_sa('"fac" > ' + threshold, str)
arcpy.gp.ExtractByMask_sa(str, Basin, strc)
arcpy.gp.StreamLink_sa(strc, fdr, strlnk)

arcpy.gp.StreamToFeature_sa(strlnk, fdr, DrainageLine_shp, "NO_SIMPLIFY")

arcpy.gp.Watershed_sa(fdr, strlnk, Catchment, "VALUE")
arcpy.RasterToPolygon_conversion(Catchment, CatchPoly_shp, "NO_SIMPLIFY", "VALUE")
arcpy.Dissolve_management(CatchPoly_shp, CatchPolyDissolve_shp, "GRIDCODE", "",
"MULTI_PART", "DISSOLVE_LINES")

arcpy.gp.StreamOrder_sa("str", "fdr", STRAHLER, "STRAHLER")  #the last arameter,
Strahler string, is actually a method of ordering stream. NOT A NAME
arcpy.AddField_management(in_table="Land_Use_Prj", field_name="ManningsN",
field_type="LONG", field_precision="", field_scale="", field_length="",
field_alias="", field_is_nullable="NULLABLE", field_is_required="NON_REQUIRED",
field_domain="")


# reclassField_nlcd = "Land Cover"
# #remap_nlcd = RemapValue([["Open Water", 0], ["Developed, Open Space",
0.0404],["Developed, Low Intensity", 0.0678],['Developed, Medium
Intensity',0.0678],['Developed, High Intensity', 0.0404],['Barren
Land',0.0113],['Deciduous Forest' , 0.36],['Evergreen Forest' , 0.368],[ 'Mixed
Forest', 0.325],[ 'Shrub/Scrub' ,0.086],[ 'Herbaceuous', 0.1825],[ 'Hay/Pasture'
,0.086],[ 'Cultivated Crops', 0.086],['Woody Wetlands', 0.086],['Emergent Herbaceuous
Wetlands',0.1825] ])
```

```
# remap_nlcd = RemapValue([["Open Water", 0*10000], ["Developed, Open Space",
0.0404*10000],["Developed, Low Intensity", 0.0678*10000],['Developed, Medium
Intensity',0.0678*10000],['Developed, High Intensity', 0.0404*10000],['Barren
Land',0.0113*10000],['Deciduous Forest' , 0.36*10000],['Evergreen Forest' ,
0.368*10000],[ 'Mixed Forest', 0.325*10000],[ 'Shrub/Scrub' ,0.086*10000],[
'Herbaceuous', 0.1825*10000],[ 'Hay/Pasture' ,0.086*10000],[ 'Cultivated Crops',
0.086*10000],['Woody Wetlands', 0.086*10000],['Emergent Herbaceuous
Wetlands',0.1825*10000] ])
# outReclassify = Reclassify("Land_Use", reclassField_nlcd, remap_nlcd, "NODATA")
# outReclassify.save("n_Overland")

# reclassField_strahler = "Value"
# remap_strahler =
RemapValue([[1,0.05],[2,0.04],[3,0.035],[4,0.03],[5,0.03],[6,0.025,]])
# outReclassify = Reclassify("STRAHLER", reclassField_strahler, remap_strahler,
"NODATA")
# outReclassify.save("n_Channel")

#Straight process reclassification, as above, did not work. so lets multiply mannings
n by 10,000. We will later divide it by 10,000 again
arcpy.gp.Reclassify_sa("Land_Use_Prj", "Value", "11 0;21 404;22 678;23 678;24 404;31
113;41 3600;42 3200;43 4000;52 4000;71 3680;81 3250;82 3250;90 860;95 1825",
outDir+"/nx10000_Overl", "DATA")

#reclassifyin strahler order to get manninsg for channel in the same way
arcpy.gp.Reclassify_sa("STRAHLER", "Value", "1 500;2 400;3 350;4 300;5 300;6 250",
outDir+"/nx10000_Chan", "DATA")

#now calulate the real mannins, divide reclassified raster by 10,000
```

```
#NLCD to n
arcpy.gp.RasterCalculator_sa("""nx10000_Overl" /10000.0""", outDir+"/n_Overland")

#strahler order to n
arcpy.gp.RasterCalculator_sa("""nx10000_Chan" /10000.0""", outDir+"/n_Channel")
```

3. STEP3:Join table with texture lookup (Run from environment that has pandas)

```
import pandas as pd
import numpy as np
import os

#Input a folder that has all the folders of names similar to  UT012, Ut027 etc.
path2collectionOfssurgoFolders = r"G:\StudyArea SSUROG"
path2lookupTable =
r"C:\Users\Prasanna\Dropbox\CLASSES\Hydroinformatics\PyProject_HI\PROJECT_RESEARCH\Aut
oPTPK2\GREENAMPT_LOOKUPTABLE.csv"


lookupTable = pd.read_csv(path2lookupTable , sep=',', skiprows = 0)
#create a list of folders only
folderList = []
[folderList.append(folders) for folders in os.listdir(path2collectionOfssurgoFolders)
   if os.path.isdir(os.path.join(path2collectionOfssurgoFolders, folders))]

for folder in folderList:
   path2ssurgo= path2collectionOfssurgoFolders + "/" + folder
   path2tabular = path2ssurgo+"//tabular"
   path2Spatial= path2ssurgo+"//spatial"
```

```
   #Make changes here!
   valuesToAvg =
['ksat_r','Ks','dbthirdbar_r','dbfifteenbar_r','Porosity','EffectivePorosity',
'BubblingPressure_Geometric', 'PoreSizeDistribution_geometric'  ]           #use the
values that we need to average
   fileNameColNoListHeaders = [
["comp",[1,5,107,108],["ComponentPercent","MajorComponent","MUKEY","COKEY"]],
                                ["muaggatt",[10,39],["AvaWaterCon","MUKEY"]],

["chorizon",[6,9,12,81,72,75,169,170],["TopDepth","BottomDepth",
"HorizonDepth","ksat_r","dbthirdbar_r","dbfifteenbar_r","COKEY","CHKEY"]],

["chtextur",[0,2,3],["textureName","CHtxtgrpKEY","CHTXTKEY"]],
                                ["chtexgrp",[4,5],["CHKEY","CHtxtgrpKEY"]]
                                ]

   def STEP1_rawToRefined(fileName_ColNoList_Headers, path=path2tabular):
       for afileColHdr in fileName_ColNoList_Headers:
           txtFilename= afileColHdr[0]
           colNo = afileColHdr[1]
           header = afileColHdr[2]

           txtFile = path + "\\" + txtFilename + ".txt"   #RETURNS FULL ADDRESS
           csvFileData = pd.read_csv(txtFile, sep = "|",  header=None, comment='#')

           reqdData = csvFileData.iloc[:,colNo]
           reqdData.columns = header
           reqdData.to_csv(path + "\\" + txtFilename + ".csv", index=False)
```

```
        return reqdData


    def STEP2_mergeCSV(path=path2tabular):
        muaggatt  = pd.read_csv(path+"/muaggatt.csv") ; print "/muaggatt.csv",
len(muaggatt.index)
        component = pd.read_csv(path+"/comp.csv")    ; print "/comp.csv",
len(component.index)
        chorizon = pd.read_csv(path+"/chorizon.csv")  ; print "/chorizon.csv",
len(chorizon.index)
        chtextur = pd.read_csv(path+"/chtextur.csv")  ; print "/chtextur.csv",
len(chtextur.index)
        chtexgrp = pd.read_csv(path+"/chtexgrp.csv")  ; print "/chtexgrp.csv",
len(chtexgrp.index)

        component_Muaggatt =  pd.merge(muaggatt , component, on='MUKEY')
        chorizon_Component_Muaggatt =  pd.merge(component_Muaggatt , chorizon,
on='COKEY')

        chTxt_chTxtGrp =  pd.merge(chtextur , chtexgrp, on='CHtxtgrpKEY')
        merged = pd.merge(chTxt_chTxtGrp , chorizon_Component_Muaggatt, on='CHKEY')



        #print chorizonWithComponent
        merged.to_csv(path + "/MERGED.csv", index=False)
        return merged


    #__main__
    try:
        #take necessary columns from the files, and add headers to them
```

```
        STEP1_rawToRefined(fileNameColNoListHeaders) ; print "Headers applied to raw
txts"
    except Exception, e:
        print e


    try:
        #STEP2 Merge (Chorizon to Component) to Muaggatt ---> result: MERGED.csv
        mergdf = STEP2_mergeCSV() ; print "Merging completed"
    except Exception, e:
        print e




    try:
        #STEP3 Merge lookup table to the LARGE table  ----->result:
OverallMergedWithTexture.csv
        mergeWithLookUp = pd.merge(mergdf, lookupTable, on= 'textureName')
        mergeWithLookUp.to_csv(path2tabular + "\\OverallMergedWithTexture.csv",
index=False)
        print "Merging with texture lookup table completed"
    except Exception, e:
        print e


    try:
        #STEP4 Take i)Height Weighted Average ii)Component % weighted average -------->
result MUKEY-Vs-Values.csv
        merged = pd.read_csv(path2tabular + "\\OverallMergedWithTexture.csv")



        #Caclulation of weighted average
```

```
        HorizonDepth2 = merged['BottomDepth'] - merged['TopDepth'] ;
merged.loc[:,'HorizonDepth2']= HorizonDepth2


        #the values whose weighted average we want, needs to be given in the list below
        #-------> MUKEY Vs Value (just one) MUKEY-Value.csv
        for valueName in valuesToAvg:       #add those values to merged
            VxD = merged['HorizonDepth2']* merged[valueName] ;
merged.loc[:,valueName+"xD_sum"]= VxD
            chorizonCalc = merged.groupby('COKEY').agg({valueName+"xD_sum":np.sum ,
'HorizonDepth2':np.sum,'ComponentPercent':np.max,'COKEY':np.max,'MUKEY':np.max })
            chorizonCalc=chorizonCalc.rename(columns =
{'HorizonDepth2':'HorizonDepth2_sum'}) #because grouping by cokey, the column name
doesnt match its data
            VxD_by_sum =
chorizonCalc[valueName+"xD_sum"].astype('float').div(chorizonCalc['HorizonDepth2_sum']
.astype('float'))
            chorizonCalc.loc[:,valueName+"_avgH"]= VxD_by_sum


            #percentage weightage
            compPerc_X_Havg = chorizonCalc['ComponentPercent'].astype('float')/100. *
chorizonCalc[valueName+"_avgH"]
            chorizonCalc.loc[:,valueName+"_WtAvg"] = compPerc_X_Havg


            #now Group it by MUKEY, and done!
            componentPercentageCalc =
chorizonCalc.groupby('MUKEY').agg({'MUKEY':np.max, valueName+"_WtAvg":np.sum })
            componentPercentageCalc.to_csv(path2tabular+"\\MUKEY-"+ valueName  +".csv",
index=False)
```

```
        #now, function to use the 'valuesToAvg' list above, and merge them against
MUKEY
        mukeyValues = componentPercentageCalc.MUKEY

    except Exception, e:
        print e
    try:
        #STEP5: Merge all the MUKEY Vs Values csv --------> result MUKEY-Vs-Values.csv
        lastValueFile = pd.read_csv(path2tabular+"\\MUKEY-"+ valuesToAvg[-1]  +".csv")
        for valueName in valuesToAvg:
            #if valueName == valuesToAvg[-1] : break
            fl = pd.read_csv(path2tabular+"\\MUKEY-"+ valueName  +".csv")
            print path2tabular+"\\MUKEY-"+ valueName  +".csv"
            lastValueFile = pd.merge(lastValueFile, fl, on="MUKEY")

        #print mukeyValuesAllMerged
        lastValueFile.to_csv(path2ssurgo+"\\MUKEY-Vs-Values.csv", index=False)
        print 'All values table written down in the ssurgo folder'

        #create a schema.ini so that arcGIS can understand the MUKEY field
        schema = open(path2ssurgo+"\\schema.ini", "w")
        schema.write("[MUKEY-Vs-Values.csv]"+ "\n" + "Col2=MUKEY Text")  #may not
always be column 1 though
        schema.close()

        #delete all the csv files made so far, except the MUKEY-Vs-Values.csv
        filelist = [ f for f in os.listdir("path2tabular") if f.endswith(".csv") ]
        for f in filelist:
```

```
            os.remove(f)


    except Exception, e:
        print e
```

4. STEP4: join SSURGO and export rasters

```
import arcpy
from arcpy import env
import os

#this program is supposed to take ssurgo datafolder path as input and attach to
soil_mu_xxx the table combined for MUKEY earlier
#one limitation, you have to open the spatial soilmu_a_xxxx file for it to work



arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

path2ssurgoFolders = arcpy.GetParameterAsText(0)
outDir = arcpy.GetParameterAsText(1)      #this is where the output rasters and the
projected polygon shapefile will be saved
cellSize = arcpy.GetParameterAsText(2)

arcpy.env.outputCoordinateSystem = arcpy.SpatialReference(102008)

#create a list of folders only
folderList = []
[folderList.append(folders) for folders in os.listdir(path2ssurgoFolders)
    if os.path.isdir(os.path.join(path2ssurgoFolders, folders))]
```

```
folderList = folderList
for folder in folderList:
   path2ssurgo= path2ssurgoFolders + "/" + folder
   path2tabular = path2ssurgo+"/tabular"
   path2Spatial= path2ssurgo+"/spatial"
   arcpy.env.workspace = arcpy.env.scratchWorkspace = path2ssurgo

   muShapefile = os.listdir(path2Spatial)[1].split('.')[0]

   #project the shapefile in ssurgo table
   arcpy.Project_management(in_dataset= path2ssurgo+"/spatial/" + muShapefile +".shp"
, out_dataset=outDir + "/"+ muShapefile +"_prj",
out_coor_system="PROJCS['NAD_1983_UTM_Zone_12N',GEOGCS['GCS_North_American_1983',DATUM
['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Greenwi
ch',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETE
R['False_Easting',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridia
n',-111.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0],UNIT['
Meter',1.0]]", transform_method="WGS_1984_(ITRF00)_To_NAD_1983",
in_coor_system="GEOGCS['GCS_WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_1984',6378137.0
,298.257223563]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]]",
preserve_shape="NO_PRESERVE_SHAPE", max_deviation="")

   # to add the projected shapefile from ssurgo, as a layer to the map at the bottom
of the TOC in data frame 0
   mxd = arcpy.mapping.MapDocument("CURRENT")                                  # get
the map document
   df = arcpy.mapping.ListDataFrames(mxd,"*")[0]                               #first
dataframe in the document
```

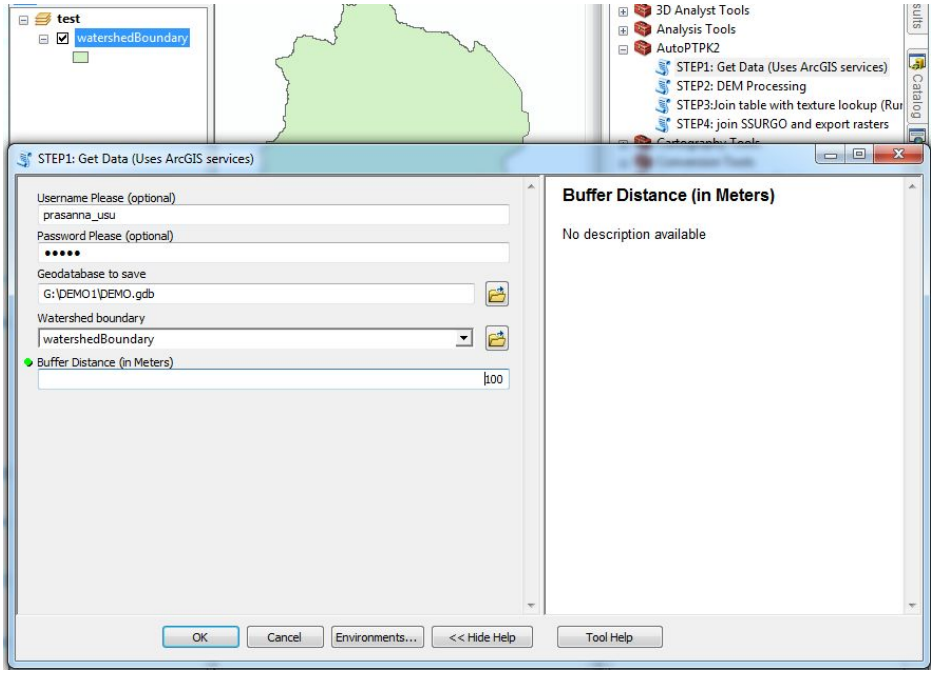
```
   newlayer = arcpy.mapping.Layer(outDir + "/"+ muShapefile +"_prj"+".shp")     #
create a new layer
   arcpy.mapping.AddLayer(df, newlayer,"BOTTOM")
   muShapefileAsLayer = muShapefile +"_prj"


   try:
       #join the table that had mUKEY mapped to all soil properties
       arcpy.AddJoin_management(muShapefileAsLayer, "MUKEY",
path2ssurgo+"/MUKEY-Vs-Values.csv", "MUKEY")


       #ssr for ssurgo, and tbl for lookup table


       soilProperties = [["ksat_r_WtAvg", "Ksat_s_"+folder ],
                         ["Ks_WtAvg", "Ksat_t_"+folder ],
                         ["Porosity_WtAvg","Por_t_"+folder ],
                         ["EffectivePorosity_WtAvg","EfPor_t_" +folder ] ,
                         ["BubblingPressure_Geometric_WtAvg", "BblPr_t_"+folder ] ,
                         ["PoreSizeDistribution_geometric_WtAvg_y","PoreSz_t_"+folder]
                         ]
       #soilProperties = [[ "ksat_r_WtAvg", "Ksat_ssur" ], ["Ks_WtAvg", "Ksat_tbl" ],
["dbthirdbar_r_WtAvg","dbthirdbar_ssur"  ] ]
       for a_soil_property in soilProperties:
           #covert from features to rasters
           #take first element of a_soil_property to find values in joint table, and
second element to name the raster
           firstNameOfSoilProperty = a_soil_property[1].split('_')[0]    #example
Ksat_s
           if not os.path.exists(outDir+"/"+firstNameOfSoilProperty):
               os.makedirs(outDir+"/"+firstNameOfSoilProperty)
```

```
            arcpy.FeatureToRaster_conversion(in_features=muShapefileAsLayer,
field="MUKEY-Vs-Values.csv." + a_soil_property[0] ,
out_raster=outDir+"/"+firstNameOfSoilProperty+"/"+ a_soil_property[1], cell_size=
cellSize )   #   ""2.71097365691884E-03")

            newRasterlayer =
arcpy.mapping.Layer(outDir+"/"+firstNameOfSoilProperty+"/"+ a_soil_property[1])    #
create a new layer
            arcpy.mapping.AddLayer(df, newRasterlayer,"BOTTOM")

        print "Folder done: ", folder
    except Exception, e:
        print "failed in folder ", folder



try:
    #merge rasters present in outDir
    FOLDERSOFRASTERS = [folder.split('_')[0] for folder in (list[1] for list  in
soilProperties)]
    for afolderOfRaster in FOLDERSOFRASTERS:
        arcpy.env.workspace = outDir+"/"+afolderOfRaster
        raster_list=arcpy.ListRasters("", "tif")
        arcpy.CompositeBands_management(raster_list, afolderOfRaster+".tif") #will save
output on the same folder

        newRasterlayer = arcpy.mapping.Layer(outDir+"/"+afolderOfRaster +".tif")    #
create a new layer
        arcpy.mapping.AddLayer(df, newRasterlayer,"BOTTOM")
```

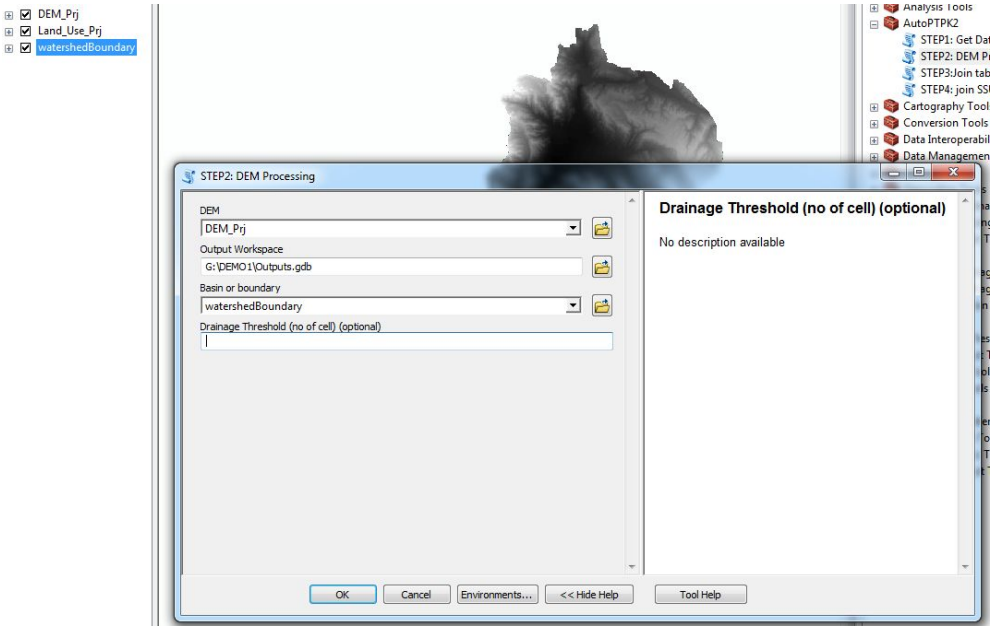
```
except Exception,e :
   print e
```

Appendix B

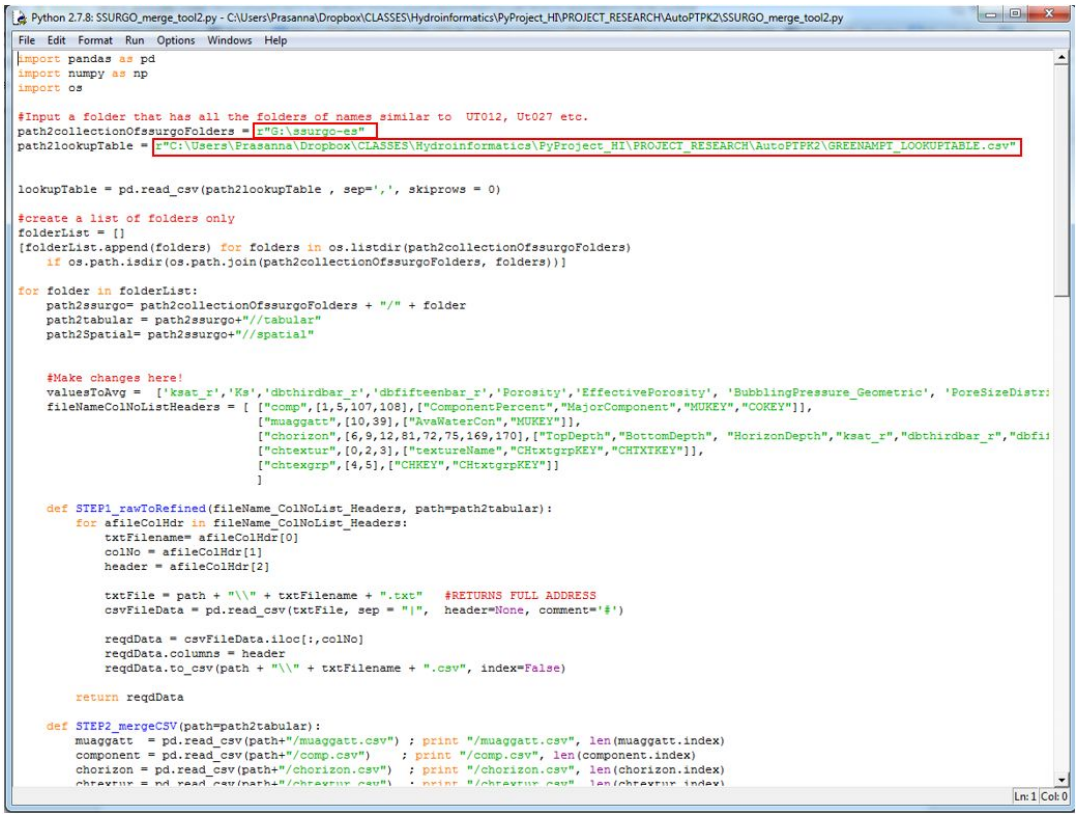
Step1:



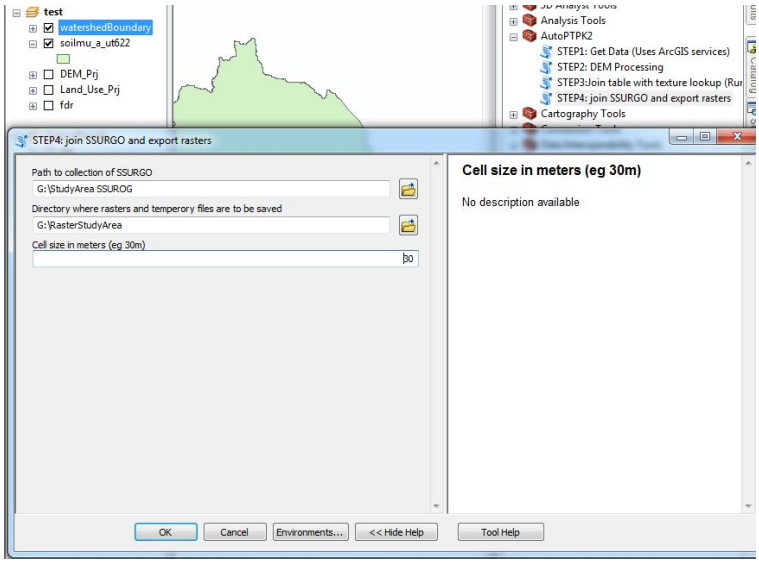
Step2:



Step3:


Python 2.7.8: SSURGO_merge_tool2.py - C:\Users\Prasanna\Dropbox\CLASSES\Hydroinformatics\PyProject_HI\PROJECT_RESEARCH\AutoPTPK2\SSURGO_merge_tool2.py
File Edit Format Run Options Windows Help
import pandas as pd
import numpy as np
import os
#Input a folder that has all the folders of names similar to UT012, Ut027 etc.
path2collectionOfssurgoFolders = r"G:\ssurgo-ea"
path2lookupTable = r"C:\Users\Prasanna\Dropbox\CLASSES\Hydroinformatics\PyProject_HI\PROJECT_RESEARCH\AutoPTPK2\GREENAMPT_LOOKUPTABLE.csv"
lookupTable = pd.read_csv(path2lookupTable , sep=',', skiprows = 0)
#create a list of folders only
folderList = []
[folderList.append(folders) for folders in os.listdir(path2collectionOfssurgoFolders)
if os.path.isdir(os.path.join(path2collectionOfssurgoFolders, folders))]
for folder in folderList:
path2ssurgo= path2collectionOfssurgoFolders + "/" + folder
path2tabular = path2ssurgo+"//tabular"
path2Spatial= path2ssurgo+"//spatial"
#Make changes here!
valuesToAvg = ['ksat_r','Ks','dbthirdbar_r','dbfifteenbar_r','Porosity','EffectivePorosity', 'BubblingPressure_Geometric', 'PoreSizeDistr
fileNameColNoListHeaders = [["comp",[1,5,107,108],["ComponentPercent","MajorComponent","MUKEY","COKEY"]],
["muaggatt",[10,39],["AvaWaterCon","MUKEY"]],
["chorizon",[6,9,12,81,72,75,169,170],["TopDepth","BottomDepth", "HorizonDepth","ksat_r","dbthirdbar_r","dbfi
["chtextur",[0,2,3],["textureName","CHtxtgrpKEY","CHTXTKEY"]],
["chtexgrp",[4,5],["CHKEY","CHtxtgrpKEY"]]
]
def STEP1_rawToRefined(fileName_ColNoList_Headers, path=path2tabular):
for afileColHdr in fileName_ColNoList_Headers:
txtFilename= afileColHdr[0]
colNo = afileColHdr[1]
header = afileColHdr[2]
txtFile = path + "\\" + txtFilename + ".txt" #RETURNS FULL ADDRESS
csvFileData = pd.read_csv(txtFile, sep = "|", header=None, comment='#')
reqdData = csvFileData.iloc[:,colNo]
reqdData.columns = header
reqdData.to_csv(path + "\\" + txtFilename + ".csv", index=False)
return reqdData
def STEP2_mergeCSV(path=path2tabular):
muaggatt = pd.read_csv(path+"/muaggatt.csv") ; print "/muaggatt.csv", len(muaggatt.index)
component = pd.read_csv(path+"/comp.csv") ; print "/comp.csv", len(component.index)
chorizon = pd.read_csv(path+"/chorizon.csv") ; print "/chorizon.csv", len(chorizon.index)
Ln: 1 Col: 0

Step4:


test
watershedBoundary
soilmu_a_ut622
DEM_Prj
Land_Use_Prj
fdr
Analysis Tools
AutoPTPK2
STEP1: Get Data (Uses ArcGIS services)
STEP2: DEM Processing
STEP3:Join table with texture lookup (Ru
STEP4: join SSURGO and export rasters
Cartography Tools
STEP4: join SSURGO and export rasters
Path to collection of SSURGO
G:\StudyArea SSUROG
Directory where rasters and temperory files are to be saved
G:\RasterStudyArea
Cell size in meters (eg 30m)
30
Cell size in meters (eg 30m)
No description available
OK
Cancel
Environments...
<< Hide Help
Tool Help