Automation of Input Data Preparation of TOPNET model Using Python script.

ArcGIS in Water Resource Engineering Term Project

Nazmus Sazib

Table of Contents

List of Figures:

1. Introduction:

GIS enables user to manipulate, analyze, visualize, and share the spatial datasets in easy and efficient way. A numerous number of operation (e.g., clipping, buffering, intersecting, joining) can be done through GIS. This operation requires using right tools for operating user data. But if anybody wants to carry out the same analysis on different location, needs to use same tool repeatedly which is time consuming and cumbersome. Automation of this analysis makes work easier, faster and more accurate. There are different ways for automation of GIS work. Model Builder is one option that allows user to merge tools together using the output of one tool as input in another. It automates complex GIS workflows without the need for programming. A script which executes multiple procedures of steps provides greater flexibility than Model Builder. Iteration or loops can be included in a script to repeat a single action as many times as needed to accomplish a task. There are different language for writing scripts, including Python, Jscript, and Perl. ESRI emphasizes Python in its documentation and includes Python with the ArcGIS install. Objective of this project is to automation of input data preparation of TOPNET model using Python script.

TOPNET was developed by combining TOPMODEL which is most suited to small watersheds, with a kinematic wave channel routing algorithm so as to have a modeling system that can be applied over large watersheds using smaller subbasin within the large watershed as model elements [1].

Digital Elevation Model (DEM), watershed boundaries, stream network and points of interest are used as input to establish the topological configuration of the model. Soils, land use, mean annual precipitation, and artificially drained areas are needed to estimate model parameters. Climate input data comprised of precipitation, temperature, wind, humidity, etc. Streamflow data is assembled for use in calibration and to drive upstream boundary inputs. In addition to streamflow, TOPNET output includes mean water table depth, soil zone storage, and canopy storage for each model element. Infiltration excess runoff, saturation excess runoff, base flow, drainage from the soil to saturated zone, potential evapotranspiration and actual evapotranspiration are also estimated by TOPNET model [2]. Preparation of input data for TOPNET is complex and involved the following steps: (1) Delineate streams, model elements and establish relationship between model element and stream connectivity (2) Establish rainfall weights for precipitation inputs to model elements (3) Estimating model parameter for each drainage using GIS layers of soils, land use and mean annual precipitation (4) Establish wetness index distribution (5) Establish distance distribution. Model elements are topographically delineated. Extensive GIS operations are carried out for preparing TOPNET model input. As I will use this model on multiple locations, it is necessary to automate the input data preparation. In this project I focused on the delineating the model element, wetness index distribution and distance distribution for each of the basin in watershed.

2. Watershed delineation:

Watershed is a basin-like landform defined by highpoints and ridgelines that descend into lower elevations and stream valleys. A watershed carries water "shed" from the land after rain falls and snow melts. Drop by drop, water is channeled into soils, ground waters, creeks, and streams, making its way to larger rivers and eventually the sea.

Delineated watershed is one of the main inputs for the TOPNET model. Sequences of Geographic Information System (GIS) processing steps are required for delineating watershed, where the digital elevation model (DEM) is used as the starting point. The TauDEM tool [3] was used for delineating the watershed. I developed a python code that contains TauDEM command line functions in the sequenced order required for delineating the watershed. The code took the DEM as an input and delineated watershed as output.

2.1 Steps of the code:

The steps for watershed delineation are given below:
1. Removed pit from DEM for creating a hydrological correct DEM.
2. Calculated D8flow direction and slope of each cell based on the corrected DEM.
3. Calculated D8contributing area based on the D8flow direction.
4. Used grid network function to get longest path, the total path and the Strahler order number.
5. Defined the stream by threshold number.
6. Moved outlet to streams.
7. Determined D8 contributing area using outlets shapfile.
8. Defined stream by threshold.
9. Estimated Stream Reach.
10. Delineated the watershed.

2.2 How to use the code:

First user need to specify the location of the working folder that contains the DEM and outlet shapfile. Change the variable name mention in the code (see appendix A) as “x1”. There are some other options (e.g., in case of drop analysis, minimum and maximum value of threshold) user can change according to their needs.

2.3 Sample input and output:

I run my code using DEM and outlet shapfile of Logan River and got delineated watershed as final output. Figure (1) shows the DEM of Logan River and Figure (2) shows the delineated watershed.

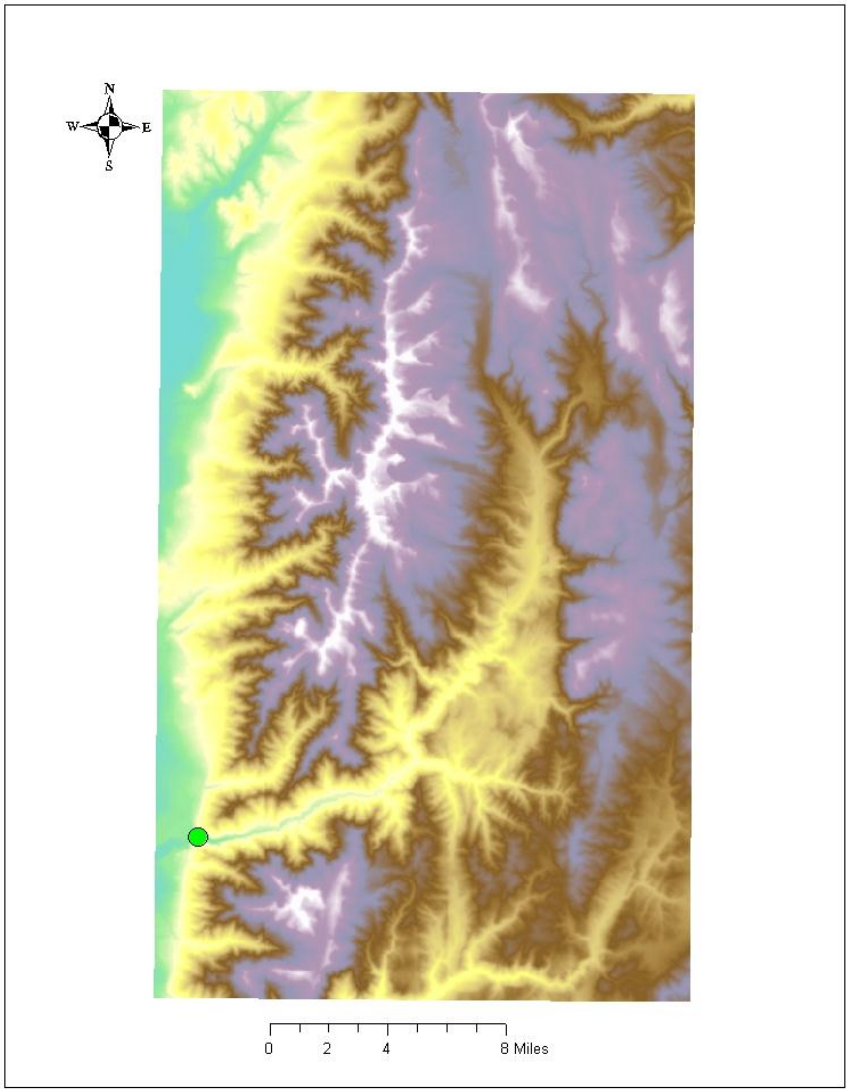


Figure 1: Digital Elevation Model (DEM) of Logan River used as an input for the watershed delineation program.

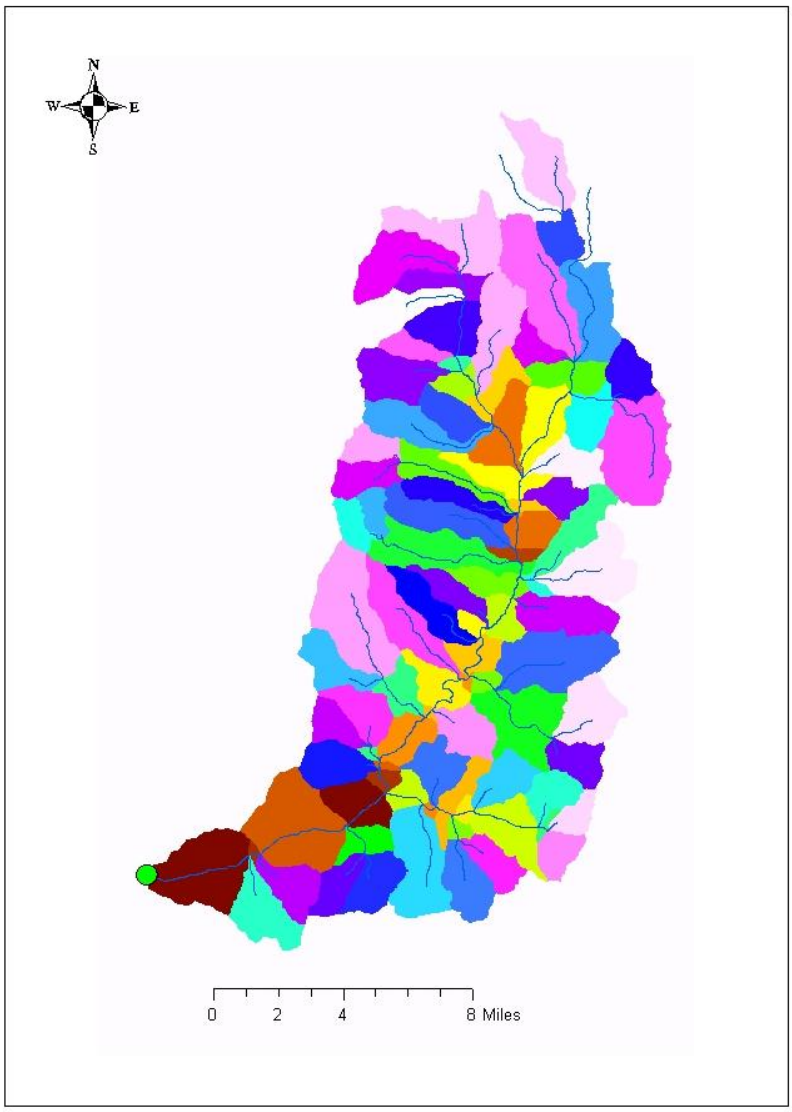


Figure 2: Delineated watershed of Logan River resulted from the watershed delineation program.

3. Wetness index distribution:

Wetness index (WI) is a function of natural logarithm of ratio of local slope contributing area and slope (from ESRI website).
Wetness index,
WI = ln(A/tanB)
A = local upslope contributing area from flow accumulation raster.
B = local slope angle.
Groundwater saturated zone is one of the TOPNET physical model component where wetness index distribution is used for calculating the local depth to water table. A histogram of wetness index values over each subbasin is used to record the proportion of each subbasin falling within each wetness index class.

I developed a python code (see appendix B) for wetness index distribution for each of the basin in watershed. The code took the model element, wetness index as input and provided wetness index distribution of each of the basin as an output.

3.1 Steps of the code:

The steps for wetness index distribution are given below:

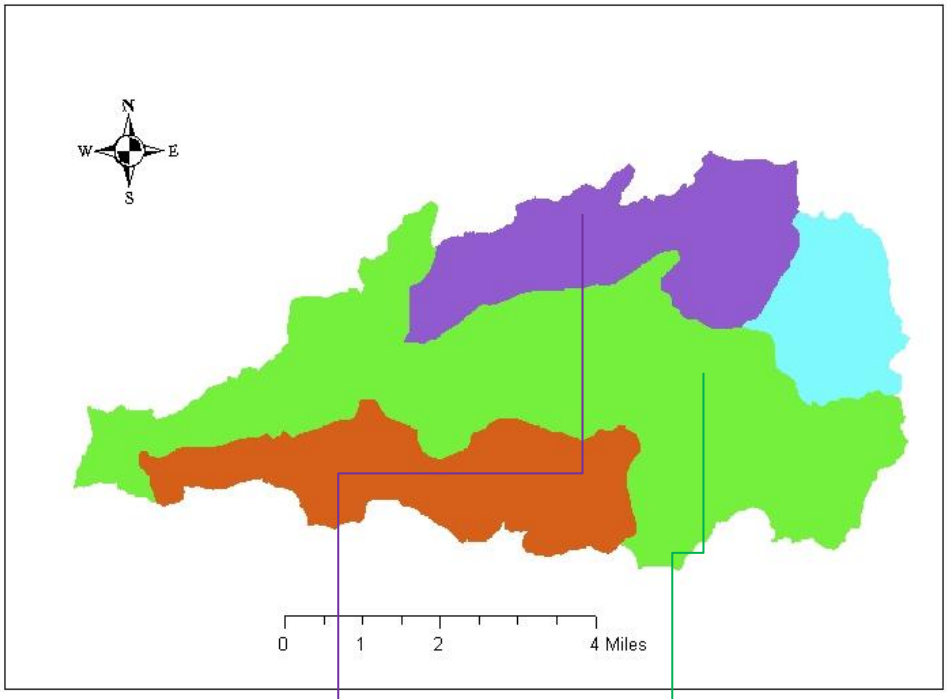
1. At first raster file of model element and a/tanb was converted to shape file using raster to point conversion function.
2. Two shapfile were then intersected to get the all of the information of each point. Intersected shapfile contained wetness index for each of the cell in a particular basin.
3. This code went through each of the cell in a basin and extract wetness index for specified basin which was appended in a list.
4. A function was developed to create bin on a criteria that each bin could not have more than 5% of the total cell in a basin and also the increment between the bins was not more than the particular value.
5. This function was applied on the basin and distribution was written in a text file. The whole process was repeated for each of the basin in a watershed.

3.2 How to run the code:

To run the code, user need to change the working folder where input files are located. Name of the feature class mentioned in “raster to point conversion” and output text file name mentioned in function “myfunc” need to be changed.

3.3 Sample input and output:

I used raster file of model element and atanb value of tenmile watershed and got wetness index distribution as output. Figure (3) depict the model element and wetness index distribution as table (bottom panel).



Wetness index distribution		Wetness index distribution	
4.510372	0	4.103021	0
5.010372	0.000505	4.603021	0.002004
5.510372	0.00314	5.103021	0.005972
6.010372	0.016203	5.603021	0.012063
6.510372	0.034537	6.103021	0.019284
6.974098	0.05	6.603021	0.028906
7.340432	0.05	7.103021	0.043092
7.669931	0.05	7.524071	0.05
7.954696	0.05	7.847929	0.05
8.202091	0.05	8.117377	0.05
8.438382	0.05	8.346607	0.05
8.645652	0.05	8.550491	0.05
8.85078	0.05	8.753105	0.05
9.061145	0.05	8.946823	0.05
9.293287	0.05	9.139137	0.05
9.550299	0.05	9.343872	0.05
9.844653	0.05	9.557489	0.05

Figure 3: Sample model element (top portion) and wetness index distribution (bottom table) for different basin in watershed.

4. Distance distribution:

Flow distance can be defined as the distance from each grid cell moving downstream until a stream grid cell as defined by the Stream Raster grid is encountered[3]. Distance distribution is used in the TOPNET model for channel routing.

I developed a python code for distance distribution. TauDEM command line function was used for distance calculation. My python code took flow distance and model element raster file as inputs and resulted distance distribution for each of the basin wrote in a text file as output.

4.1 Steps of the code:

The steps for watershed delineation are given below:

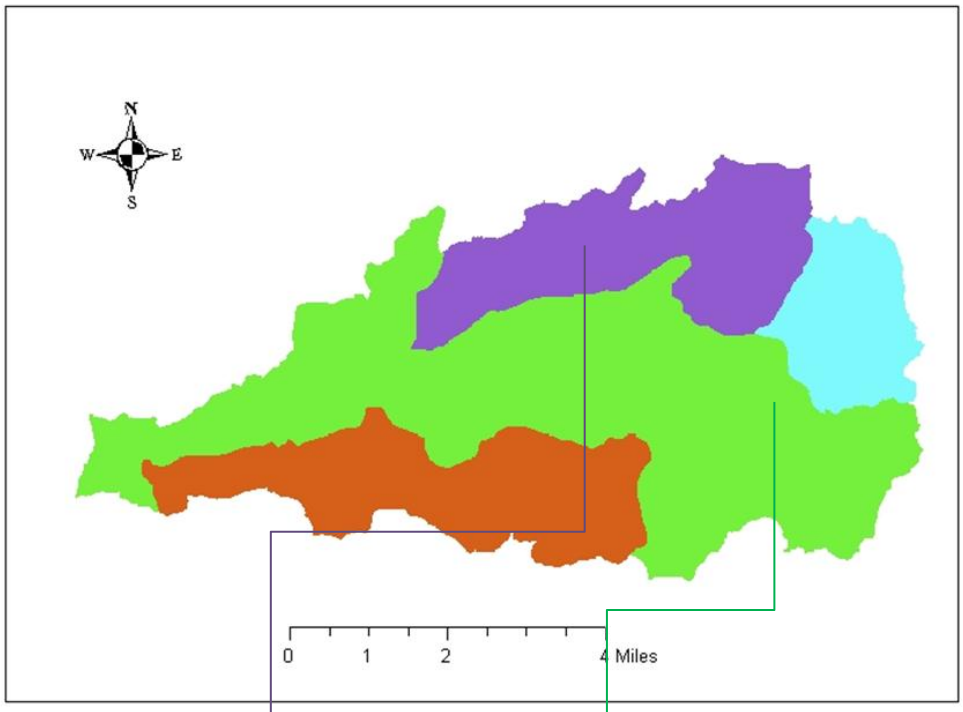
1. First raster file of model element and flow distance was converted to shapfile using raster to point conversion function.
2. Two shapfile were then intersected to get the all of the information of each point. Intersected shapfile contained distance value for each of the cell in a particular basin.
3. This code went through each of the cell in a basin and extract flow distance value for specified basic which was appended in a list.
4. Then a function was called to create bin on a criteria that each bin could not have more than 20% of the total cell in a basin and also the increment between the bins was not more than the particular value.
5. Distance distribution was written in a text file. The whole process was repeated for each of the basin in a watershed.

4.2 How to run the code:

To run the code, user need to change the working folder where input files are located. Name of the feature class mentioned in “raster to point conversion” and output text file name mentioned in function “mydist” need to be changed.

4.3 Sample input and output:

I used raster file of model element and distance value of tenmile watershed and got distance distribution as output. Figure (4) depict the model element (top panel) and distance distribution as table (bottom panel).



Distance distribution		Distance distribution	
0.00	0.00	0.00	0
162.43	0.20	199.71	0.2
354.85	0.40	432.43	0.4
621.84	0.60	729.41	0.6
959.12	0.80	1141.25	0.8
1459.12	0.96	1641.25	0.94
1959.12	1.00	2141.25	1

Figure 4: Model element and distance distribution of tenmile watershed.

5. *Conclusion:*

The TOPNET model is a distributed hydrologic model developed for simulating stream flow. Input data preparation of TOPNET model is complex. Some of the data preparation steps were automated in this project which made preprocessing work faster and easier because:

- User don't have to put as much effort remembering which tools to use or the proper sequence in which they should be run.
- A computer can open and execute tools in sequence much faster than user can accomplish by pointing and clicking.
- There is a chance for error while performing delineation of watershed on a computer. Once an automated procedure is configured, a computer can be trusted to perform the same sequence of steps every time.

I have some limitations for the wetness index distribution and distance distribution program, as it took time (around 1minute) to run. My future plan is to make those codes more efficient so that it can run fast. In the future, I plan to automate the other TOPNET input data preparation steps (e.g., establish relationships between model elements and stream connectivity, establish rainfall weights for each of the basins in watershed).

6. *References:*

[1] Bandaragoda, C., D. G. Tarboton and R. Woods, (2004), "Application of Topnet in the Distributed Model Intercomparison Project," Journal of Hydrology, 298: 178-201, doi:10.1016/j.jhydrol.2004.03.038.
[2] Tarboton, D. G., (2007), "Surface Water Quantity Model Development and Calibration, WRIA 1 Watershed Management Project Phase III, Task 4.1 report," Utah Water Research Laboratory, Utah State University.
[3] Tarboton, D. G., (2007), Guide to using the TauDEM command line functions. Available at: http://hydrology.usu.edu/taudem/taudem5.0/TauDEM5CommandLineGuide.pdf

Appendix A: Code for watershed delineation:

```
import arcpy
import os
import sys
import time
import string
import subprocess
os.chdir("E:\Course_work\FALL_2012\CEE6440_GIS\Term Project\Logan")
os.environ=["PATH"]
x1="logan"
cmd='mpiexec -n 8 pitremove logan.tif'
os.system(cmd)
cmd2='mpiexec -n'+" "+' 8' + " " +'D8Flowdir' +" " + '-p'+ " " +x1+ 'p.tif' + " " +'-sd8'+ " "
+x1+'sd8.tif'+ " " + ' -fel' +" " + x1 +'fel.tif'
os.system(cmd2)
cmd3='mpiexec -n 8 DinfFlowdir -ang ' + x1+'ang.tif'+ " " +'-slp' +" " + x1+'slp.tif'+ " " + '-fel'+
" " + x1+'fel.tif'
os.system(cmd3)
cmd4='mpiexec -n 8 AreaD8 -p '+ " " +x1+ 'p.tif'+ " "+ '-ad8'+ " " +x1+ 'ad8.tif'
os.system(cmd4)
cmd5='mpiexec -n 8 AreaDinf -ang'+ " " +x1+'ang.tif' + " " + '-sca'+ " " +x1+ 'sca.tif'
os.system(cmd5)
cmd6='mpiexec -n 8 Aread8 -p ' + " " +x1+'p.tif' + " " +' -o' + " " +x1+'outlet.shp'+ " " + '-ad8' +
" " +x1+'ad8o.tif'
os.system(cmd6)
cmd7='mpiexec -n 8 Gridnet -p' + " " +x1+'p.tif'+ " " + '-plen' + " "+ x1+ 'plen.tif'+ " "+ '-tlen' +"
"+ x1+'tlen.tif'+ " "+ '-gord' + " "+ x1+'gord.tif'
os.system(cmd7)
cmd8='mpiexec -n 8 PeukerDouglas -fel' +" " +x1+ 'fel.tif'+ " "+ '-ss' +" "+ x1+'ss.tif'
os.system(cmd8)
cmd9='mpiexec -n 8 Aread8 -p '+" " +x1+'p.tif'+" "+' -o' +" "+x1+'outlet.shp' +" "+'-ad8' + "
"+x1+ 'ssa.tif '+ " " + '-wg'+ " " +x1+ 'ss.tif'
os.system(cmd9)
cmd10='mpiexec -n 8 Dropanalysis -p'+" "+x1+ 'p.tif'+ " "+ '-fel'+ " "+x1+ 'fel.tif'+" "+ '-ad8'+"
"+x1+ 'ad8.tif'+" "+ '-ssa'+" "+ x1+'ssa.tif' +" "+ '-drp'+" "+x1+ 'drp.txt' +" "+ '-o'+ " "+
x1+'outlet.shp' +" "+ '-par' +' 5 500 10 0'
os.system(cmd10)
cmd11='mpiexec -n 8 Threshold -ssa'+" "+x1+'ssa.tif'+" "+ '-src'+" "+ x1+'src.tif' +" "+' -thresh
300'
os.system(cmd11)
cmd12='mpiexec -n 8 Streamnet -fel'+" "+x1+ 'fel.tif'+ " "+' -p' + " "+x1+ 'p.tif'+" "+' -ad8'+" "+
x1+ 'ad8.tif'+" "+' -src'+"  "+  x1+'src.tif'+" "+' -ord' +" "+x1+ 'ord3.tif' +" "+ '-tree'+ " "+x1+
```

```
'tree.dat '+" "+ '-coord' + " " +x1+'coord.dat -net' +" " +x1+ 'net.shp -w' +" " + x1+'w.tif' + " "+' -o' +" "+ x1+'outlet.shp'
os.system(cmd12)
```

Appendix B: Code for wetness distribution:

```
import arcpy, os
from arcpy import env
import math
import numpy
import scipy
arcpy.env.overwriteOutput = True
env.workspace =r'E:\Course_work\FALL_2012\CEE6440_GIS\Term Project\tenmilenew'
import arcpy
from arcpy import env
arcpy.RasterToPoint_conversion("tmdemdatanb", "tmdematanb.shp", "VALUE")
arcpy.RasterToPoint_conversion("tmdemme", "tmdemme.shp", "VALUE")
inFeatures = ["tmdemdist.shp", "tmdemme.shp"]
intersectOutput = "atanbintersection.shp"
clusterTolerance = 1.5
arcpy.Intersect_analysis(inFeatures, intersectOutput, "", clusterTolerance, "point")
#featureClass = "modelintersectiondist.shp"
featureClass = "atanbintersection.shp"
rows = arcpy.SearchCursor(featureClass)
currentIN_FID = "GRID_CODE"
currentNear_FID = "GRID_COD_1"
newdata = []
####### Iterate through the rows in the cursor
for row in rows:
  rowN = [row.getValue(currentIN_FID), row.getValue(currentNear_FID)]
  newdata.append (rowN)
  del row  #be sure to delete the cursor objects, otherwise a lock on the table will persist!!!

import numpy as np
c=np.asmatrix(newdata)
p=[63,72,87,200]
for riw in p:
    condition =c[:,1]==riw
    ff= np.extract(condition, c[:,0])
    gg=ff[ff>0]
    gg=scipy.sort((scipy.log(1/gg)))
    #gg=scipy.sort(gg)
    hh1=myfunc(gg)

import numpy
```

```
def myfunc(data):
  DD=[min(data)]
  xx=[0]
  x2=[]
  x3=[max(data)]
  length=(len(data)*0.05)
  if DD<x3:

##  while len(DD)<20:
   for kv in DD:
    count1=[i for i,x in enumerate(data) if kv<=x<kv+0.5]
    dd=len(count1)
    if dd<=length:
      x2=kv+0.5
      dd=dd
      if x2<x3:
       DD.append(x2)
       xx.append(dd)
       #print xx
      else:
        break
else:
     x2=data[count1[int(1+length)]]
     dd=length
     DD.append(x2)
     xx.append(dd)
  print xx, DD
A=numpy.array(xx)
  B=numpy.array(DD)
  DataOut =numpy.column_stack((B,A/(len(data))))
  text_file = open("C:/Users/Nazmus/Desktop/F2001.txt", "a")
 return np.savetxt (text_file, DataOut,fmt=("%4.6f"))
```

Appendix C: Code for distance distribution:

```
import arcpy, os
from arcpy import env
import math
import numpy
import scipy
arcpy.env.overwriteOutput = True
env.workspace =r'E:\Course_work\FALL_2012\CEE6440_GIS\Term Project\tenmilenew'
import arcpy
from arcpy import env
```

```
arcpy.RasterToPoint_conversion("tmdemdist", "tmdemdist.shp", "VALUE")
arcpy.RasterToPoint_conversion("tmdemme", "tmdemme.shp", "VALUE")
inFeatures = ["tmdemdist.shp", "tmdemme.shp"]
intersectOutput = "modelintersectiondist"
clusterTolerance = 1.5
arcpy.Intersect_analysis(inFeatures, intersectOutput, "", clusterTolerance, "point")


featureClass = "modelintersectiondist.shp"
#featureClass = "atanbintersection.shp"
rows = arcpy.SearchCursor(featureClass)
currentIN_FID = "GRID_CODE"
currentNear_FID = "GRID_COD_1"
newdata = []
####### Iterate through the rows in the cursor
for row in rows:
  rowN = [row.getValue(currentIN_FID), row.getValue(currentNear_FID)]
  newdata.append (rowN)
  del row  #be sure to delete the cursor objects, otherwise a lock on the table will persist!!!
import numpy as np
c=np.asmatrix(newdata)
p=[63,72,87,200]
for riw in p:
     condition =c[:,1]==riw
     ff= np.extract(condition, c[:,0])
     gg=ff[ff>0]
     #gg=scipy.sort((scipy.log(1/gg)))
     gg=scipy.sort(gg)
     print gg
     hh1=mydist(gg)

import numpy
def mydist(data):
  DD=[0]
  xx=[0]
  x2=[]
  x3=[max(data)]
  length=(len(data)*0.2)
  if DD<x3:

##  while len(DD)<20:
   for kv in DD:
    count1=[i for i,x in enumerate(data) if kv<=x<kv+500]
    dd=len(count1)
    if dd<=length:
      x2=kv+500
```

```
      dd=dd
      if x2<x3:
       DD.append(x2)
       xx.append(dd)
       #print xx
      else:
        break

##
    else:
     x2=data[count1[int(1+length)]]
     dd=length
     DD.append(x2)
     xx.append(dd)
  print xx, DD

  A=numpy.array(xx)
  B=numpy.array(DD)
  DataOut =numpy.column_stack((B,A/(len(data))))
  text_file = open("C:/Users/Nazmus/Desktop/F2002.txt", "a")

  return np.savetxt (text_file, DataOut,fmt=("%4.6f"))
```