Michigan Technological University

Michigan Technological University
Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2017

DEVELOPING A GIS TOOL FOR INFINITE SLOPE STABILITY ANALYSIS (GIS-TISSA)

Jonathon Sanders
Michigan Technological University, jsander1@mtu.edu

Copyright 2017 Jonathon Sanders

Recommended Citation

Sanders, Jonathon, "DEVELOPING A GIS TOOL FOR INFINITE SLOPE STABILITY ANALYSIS (GIS-TISSA)", Open Access Master's Thesis, Michigan Technological University, 2017.
https://doi.org/10.37099/mtu.dc.etdr/461

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

 Part of the Geology Commons

DEVELOPING A GIS TOOL FOR INFINITE SLOPE STABILITY ANALYSIS (GIS-TISSA)

By

Jonathon D. Sanders

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Geology

MICHIGAN TECHNOLOGICAL UNIVERSITY

2017

© 2017 Jonathon D. Sanders

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Geology.

Department of Geological and Mining Engineering and Sciences

Thesis Advisor: *Dr. Thomas Oommen*

Committee Member: *Dr. John Gierke*

Committee Member: *Dr. Ann Maclean*

Department Chair: *Dr. John Gierke*

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my committee members, Dr. Ann Maclean and Dr. John Gierke for their support of my research. Special thanks to my advisor, Dr. Thomas Oommen, for his patience and support through the entirety of the research.

To Rudiger Escobar-Wolf and Sajin Kumar Kochappi Sathyan for providing me with information and answers invaluable to completing this research.

To Brittany Buschell and Kelly McLean for taking care of logistics and answering trivial graduate student questions.

To my family, relatives and in-laws, for supporting me to no end.

And finally, to my wife, who put up with never-ending discussions where I seemed to be talking to myself on a daily basis.

Abstract:

The Probabilistic Infinite Slope Analysis model (PISA-m) is a widely used computer program that uses infinite slope equations to calculate the spatially varying Factor of Safety of slopes. ESRI's ArcGIS software and accompanying geoprocessing tools have become a mainstay in spatial data processing, and received full support for Python with the release of version 10. With many of the geoprocessing tools now available as a Python function, the software can be used for physics-based spatial landslide hazard analysis. A model that mimics PISA-m and its processing of normally distributed soil properties was created using the Python utility as a tool for ArcGIS. The newly created ArcGIS tool is referred as the GIS Tool for Infinite Slope Stability Analysis (GIS-TISSA). The tool was tested using the example data from PISA-m and case-study data from the district of Kannur, Kerala, India. The results from both areas highlight how different slope calculations can affect the overall calculation of the Factor of Safety, as well as the new model's ability to accurately predict Factor of Safety of slopes in an area.

1. Introduction

1.1 Landslide Overview

Landslides are downslope movement of soil, rock, and/or organic matter along a rupture or shear strained surface (USGS, 2008). Landslides occur in response to a triggering mechanism (Varnes, 1978) such as earthquakes, intense rainfall (Smith et al. 2015), volcanic eruptions (Schaefer et al. 2015; Schaefer et al. 2016), weathering, freeze-thaw (Zwissler et al. 2014), and flooding (USGS, 2008). A slope prone to instability most likely has many causes, for its failure (Varnes, 1958). These causes can include geological or morphological causes, both of which prime the slope for failure. While the triggering mechanism was the final event before a landslide, there are many underlying conditions which play a role in the failure. Human activities also cause landslides. Landslides are a major cause of deaths and infrastructure damage. Between 2004 and 2010, the Durham Fatal Landslide Database recorded 2620 landslides that killed a total of 32,322 people globally (Petley, 2012). EM-DAT, another global landslide database, lists 186 landslide events occurring between 1994 and 2003, killing 8,679 overall and causing 427 million dollars of damage (Kuriakose, 2006). However, since many small events go unnoticed it is difficult to fully estimate losses, both databases are considered to underestimate the effects of landslides, more so with EM-DAT (Petley, 2012, Kuriakose, 2006). While deaths may be the first loss immediately thought of after any natural disaster, the economic damage can be devastating in the long term. On

December 13th, 1982, a 220 ha (543.6 acre) landslide at Ancona, Italy killed one person but caused an estimated 700 million dollars in damages (Guzzetti, 2000).

A major distinction in the class of landslide is the type of slip surface: translational or rotational. Rotational landslides occur on an upward curved surface, with movement rotating around an axis parallel to the contour of the terrain, and most frequently occur in homogenous materials. A translational landslide moves along a planar surface with little rotational movement, commonly along a boundary between two soil types, or along the soil-bedrock boundary (USGS, 2008). This difference in slip surface commonly means that translational landslides are usually shallower than rotational landslides. For the purposes of this paper, the translational landslide description will be inferred in any mention of landslides unless otherwise noted from this point on. Translational landslides occur much more frequently than rotational landslides. For example, 267 of 360 (74%) landslides in a 61.6 sq. km. (23.8 sq. mi.) region north of Lisbon were translational (Zêzere, 2005), as were 1,377 of 1,460 (94%) landslides near Kurseong Town in West Bengal, India (Ghosh, 2011). In these cases, the remaining landslides were either rotational or complex movement, which is a combination of translational and rotational process. Therefore, predicting the susceptibility of slopes to translational landslide is critical. Moreover, global warming is expected to increase the frequency and intensity of severe rainfall events, a primary factor that triggers landslides increasing the exposure of communities and infrastructure to landslide risk (Gariana and Guzzetti, 2016).

1.2 Susceptibility of Slopes to Landslides: Factor of Safety Modelling

The stability of slopes is reflected by a property called the Factor of Safety (FS), which is defined as the ratio of resisting and driving forces within the slope. The complexity of the FS equation changes depending on assumed conditions of the slope. For example, Ahmed et al. (2012) contrasted with Duncan (2000), both calculate FS, however Ahmed's equations are much more involved. Many different FS equations are often incorporated into modelling tools that calculate the FS, such as Map-Based Probabilistic Infinite Slope Analysis (PISA-m) (Haneburg, 2007), the Transient Rainfall Infiltration and Grid-Based Regional Slope-Stability Model (TRIGRS), and Scoops3D. Furthermore, models also have different means of analysis to calculate the FS, depending on the intention of the model, for example, infinite slope analysis or limit-equilibrium analysis. PISA-m is a map based probabilistic infinite slope analysis model that performs probabilistic static or seismic slope stability calculations. Although the model can be run to find the mean FS, it can also be used to find the probability of shallow, translational landslides. The model uses a First-Order, Second Moment (FOSM) version of the infinite slope equation. This allows PISA-m to calculate the FS without having to average multiple iterations to obtain a value, saving time and computing resources. PISA-m requires inputs of a Digital Elevation Model (DEM), a soil unit raster, and a tree unit raster, converted to either Surfer or Arc ASCII grid format. Additionally, a parameter file is created containing keywords and values used by the model, including the distributions of the soil and tree properties used in the FS equation. This model calculates the FS of a pixel, two dimensions, on any slope in any one moment in time (Haneberg, 2007).

The Transient Rainfall Infiltration and Grid-Based Regional Slope-Stability Model (TRIGRS) adds the complexity of time, while still using infinite slope analysis, to calculate the FS value. As with PISA-m, TRIGRS can help identify slopes where translational landslides may occur. TRIGRS computes pore pressure in multiple moments in time due to rainfall infiltration. Since the FS equation is partly dependent upon the pore pressure, the FS can also change. Inputs for TRIGRS include hydraulic conductivity, rainfall amounts, and time intervals, along with soil properties. The extra input variables over PISA-m are included to solve a complicated calculation for groundwater pressure head, which is then placed in the FS equation for the model. This model also calculates the FS in two dimensions. (Baum et al., 2008).

Scoops3D computes the FS of a slope while also modelling the volume of the potential slide in three dimensions. Using “method of columns” limit equilibrium analysis, Scoops3D calculates multiple (millions) slip surfaces. The surfaces with the lowest stability are combined into a three dimensional, scoop shaped surface of a rotational landslide. Following the theme, the model accepts three dimensional properties, as well as earthquake loading. Naturally, the FS equation for this model is more complicated to solve for the extra dimension of stability. Scoops3D also allows the user to choose between different FS equations, all of which tweak the results to better incorporate certain slope properties (Reid et al, 2015).

1.3 Geographic Information Systems and Spatial Data Analysis

The Geographic Information System (GIS) industry is a multi-billion-dollar industry where softwares are continually being developed and maintained. ESRI, the creator of ArcView and ArcGIS, recently introduced their new program ArcPro.

Spatial data availability is also set to increase in the future. Landsat 9 satellite is planned for launching in December 2020. This satellite will allow the continued access to free global spectral data. NOAA is also constantly adding precipitation data to more states, as well as updating existing data. While these are just a couple of examples, there are many countries and agencies across the globe beginning to collect and make available spatial data for a wide range of topics, including political boundaries, soils, and even storm pipe networks.

1.4 PISA-m in ArcGIS: GIS-TISSA

PISA-m requires ASCII formatted files for analysis, a file type that is included in the definition of a raster, generally described as “a regular grid to cover the space and the value in each grid cell to represent the characteristic of a spatial phenomenon at the cell location” (Chang, 2016). Rater data is additionally described as either continuous or discrete in nature. The use of discrete data for some of the soil and tree properties required by PISA-m can cause information to be lost. For example, Dietrich et al. (1995) discusses the importance of using continuous data for soil depth in slope stability modelling. Variation of soil depth is not necessarily constrained to soil classes, and local changes in soil depths can be overlooked when confined to a broader class. Dietrich’s paper explains the creation of a model to estimate soil depth continuously across a

region, and later calls for similar models to be created for root cohesion. Many GIS programs can handle both continuous and discrete raster data, and the approach PISA-m uses can be improved upon by using continuous data, where acceptable, to predict local slope stability more accurately.

Other times spatial data is only available in vector format. Since PISA-m is a stand-alone slope stability prediction model that uses only raster data, another program must be used to convert data from vector to raster format. Once again, many GIS programs can do this conversion, however ESRI ArcGIS and ERDAS IMAGINE have user-friendly tools that help automate the process.

ESRI has also opted to not only include the base version of Python 2.7 in ArcGIS installations, but created a Python package named ArcPy. Most of the software's geoprocessing tools can be called directly from this package and used in scripts to automate the processing of spatial data. Users well versed in ArcGIS processing can easily combine base Python packages with the geoprocessing tools in the ArcPy package to create robust models.

1.5 Objectives

This paper will attempt to use methods used in PISA-m to create a new model using ArcGIS and the accompanying tools in the ArcPy package. This newly created model, referred as GIS-TISSA, will be verified against PISA-m using the example data provided with PISA-m. GIS-TISSA will then be used to predict slope stability in the Kerala district of Kannur in India. This result will be compared to a PISA-m result of the same area. The

creation of data inputs for GIS-TISSA representing the case study area will be discussed as well.

2. Study Area

The PISA-m example data consists of three rasters, DEM, soil classes, and tree classes, and a parameter file containing the soil and tree property values and their distributions, all used in FS calculations. Each raster is formatted as an ESRI ASCII file with 501 rows and 501 columns. DEM values range from nearly 90 meters to 217 meters. Both soil and tree class rasters contain two classes, although the classes for either raster do not cover the same area. This data will be modelling by PISA-m and GIS-TISSA, using the results from both models to verify the FS prediction accuracy of GIS-TISSA.

Kannur is the second northern-most district in the state of Kerala, India (Figure 1). The months of June through December are considered monsoon season, bringing 2240.4 mm/yr to the river basin in Kannur. Additionally, this basin also receives the largest amount of annual rainfall in Kerala, at 3,107 mm/yr (Jain, 2012).

The Kannur population reached 2,523,003 in 2011. The urban population in 2001 increased to 1,640,986 in 2011, while rural population decreased to 882,017 in the same period (CensusInfo, 2013). This indicates that people are moving to relatively more densely populated areas, increasing the number of people potentially effected by a single landslide or other natural hazard. The district has slopes ranging from zero

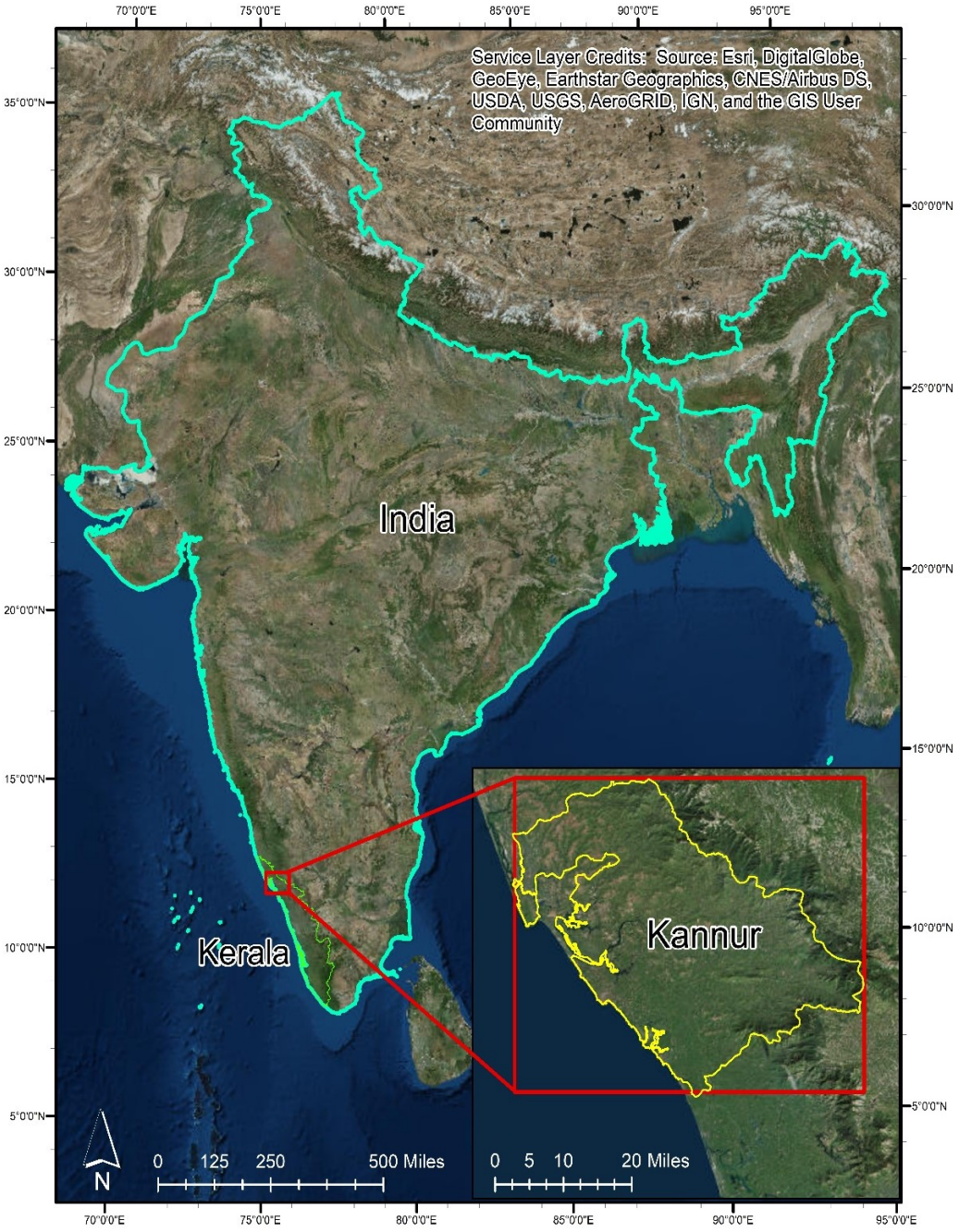

Figure 1: Map of the study area, marked by the red box. Although the focus of the study was Kannur, the bounding box created and used in GIS-TISSA includes areas outside the political boundaries of Kannur. Created by author using data from ESRI and public source data.

degrees near the coast to greater than 50 degrees in the foothills of the Western Ghats. The large amount of rainfall coupled with the variation in slope creates an environment that is favorable for landslides. Soils near the coast are generally sandy and well drained, while the soils in the Western Ghats foothills are either gravelly clay or gravelly loam. The area between the coast and mountains is mostly gravelly clay of varying depth with intermittent clayey soils. Most soil units are well drained, although there are some clayey soils that are poorly drained.

In June of 2016, monsoon rains caused landslides throughout the district, which damaged cropland, houses, and roads (Times of India, 2016). In August of 2012, heavy rains, floods, and landslides were blamed for the death of nine in Kannur and neighboring Kozhikode district. (Madhyamam, 2012).

Once verified, GIS-TISSA will be used to predict the stability of slopes in the Indian district of Kannur, and verified against the results of PISA-m in the same area. The DEM was 5-degree tile of SRTM data (1 arc-second resolution), downloaded from USGS EarthExplorer, as well as Landsat 8 Red and Near-Infrared (NIR) band rasters, used in the creation of Normalized Difference Vegetation Index (NDVI). The Landsat imagery comes from May 19, 2017, and has 30 meter resolution. Soil maps of northern and southern Kerala were used for soil boundaries and properties in the Kannur region (NBSS, 1996).

3. GIS-TISSA Code

GIS-TISSA combines the equations used in PISA-m, a root cohesion estimation equation, and the geoprocessing tools of ArcGIS to calculate the FS value in a rectangular region. After some manual preprocessing is completed, GIS-TISSA performs more preprocessing

immediately after initiation before creating intermediate spatial data that is used in the final calculation of FS. This section will go into detail the preprocessing, processing, and calculations performed in GIS-TISSA.

3.1 Manual Preprocessing

PISA-m requires each of its three ASCII input files to be preprocessed so each has the same extent and resolution before running the model. A goal for GIS-TISSA is to include this preprocessing, allowing for a more raw form of the inputs to be used. However, there are three other processing steps that must be performed on data before it can be input into the model.

1) All spatial data must be assigned the same spatial reference, allowing the script to clip to the correct coordinates. In addition, all the data layers should have a common spatially overlapping area.
2) The soil data must be in feature class data type, and must have an integer attribute field named “SoilInteger”. Furthermore, this attribute column must have an integer representing each soil unit in the feature class. (Figure 2)

	OBJECTID *	SoilInteger
	1	2
	2	1

Figure 2: Example of "SoilInteger" attribute field in attribute table of a feature class. Each integer represents a soil class in the feature class.

3) A comma separated value (CSV) file must be created containing the values for each soil property. At the moment, GIS-TISSA only calculates FS from normal

distributions. Due to this, only two columns, the first headed with “mean”, and second headed with “sd” (standard deviation) are required. These headers allow the script to create either mean or standard deviation rasters for each property. The values in the CSV should be grouped by property, and there should be as many values for each property as there are units in the soil feature. Additionally, the values representing each unit should appear in the same order as the units in the “SoilInteger” attribute field. Finally, the properties must be in the following order in the CSV for the script to work correctly: Friction Angle, Soil Cohesion, Moist Unit Weight, and Saturated Unit Weight. In the code, water unit weight has set units of N/m^3, and so every soil property should have the same format of units in the CSV. (Table 1, Appendix I)

Table 1: Example of CSV used by GIS-TISSA using the same soil units from Figure 2. For this particular number of soils, every two rows are a new soil property. The order of properties shown is essential for proper use of GIS-TISSA.

mean	sd	Property	Soil Unit
32	0.81	Friction Angle	2
33	0.81		1
10000	39	Soil Cohesion	2
5500	42		1
0.1	4	Soil Depth	2
0.1	4		1
0.5	0.1	Phreatic Ratio	2
0.5	0.1		1
20000	26	Saturated Unit Weight	2
21500	22		1
16500	32	Moist Unit Weight	2
18000	25		1

3.2 Programed Preprocessing

The beginning of the code performs simple variable definitions that are used throughout the code. A snap raster is defined in the GUI, adjusting the pixels alignment of any input raster to match that of the defined snap raster. This assures that the pixels in every raster align exactly with one another, regardless of starting position. Using this setting, the lower left corner of an extent of a raster is snapped to the lower left corner of the nearest pixel in the snap raster. When the resolutions of the input raster and snap raster are different, the bottom and left boundaries of the extent will be the same, while the top and right boundaries can be different from the snap raster. When the resolutions are the same, processed extents will cover the same area.

The next portion of the script prepares the input layers for further analysis. An area of interest (AOI) feature class is created based on the user input for the AOI entry field in the GUI. The minimum and maximum X and Y bounding values, i.e. longitude and latitude, of the input area are used to create a bounding box around the area. At this point, the spatial reference of the soil input file is saved, and is used throughout the code to specify a reference for any newly created layers. The AOI layer is no exception; assigning a spatial reference to it gives a position in space to the coordinates representing the corners of the box. These corners are looped through to create a list of coordinates, which is then used in the data management tool "*Clip*". Any raster input into the GUI will be cropped using this tool. On the other hand, any input feature classes will be clipped using the analysis tool "*Clip*" and the newly created AOI feature class. After all inputs are clipped, the rasters are resampled to the resolution of the specified

processing resolution, which should also be the smallest resolution of any of the rasters. For consistency, all input rasters are resampled using the bilinear resampling technique and are resampled regardless of resolution, allowing the program to be as generic as possible. Values in a raster with resolution equal to that of the processing resolution are not altered, effectively changing the name of the rasters those that are used later in the code. A slope raster is then created from the resampled DEM using the spatial analyst tool Slope. The ArcGIS "*Slope*" tool uses the values of the 8 nearest neighboring pixels (Figure 3) around a center pixel to calculate slope (Equation 1), output to units of degrees. The window used to gather DEM values used in the slope calculation starts at the very edge of the input raster. As it encounters "NoData" values, it assigns to those pixels the value of the center pixel in the window (Figure 4).

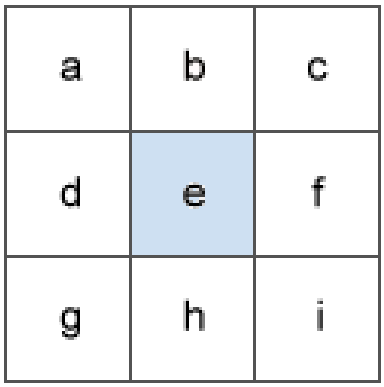

Figure 3: Pixel position reference for the slope equation used in the Slope tool in ArcGIS. From ArcMap online resources.

$$slope = \text{atan}(\sqrt{x^2 + y^2}) \quad (1)$$

where:

$$x = \frac{(c+2f+i)-(a+2d+g)}{8dx} \quad (1.1)$$

$$y = \frac{(g+2h+i)-(a+2b+c)}{8dx} \quad (1.2)$$

20	40	60		20	40	60
NoData	Center Pixel 55	70	→	55	Center Pixel 55	70
NoData	75	80		55	75	80

Figure 4: Example of how the Slope tool in ArcGIS handles "NoData" values. NoData pixels on the right are assigned the value of the center pixel before the slope is calculated.

Then, the "*Conditional tool"*, from the spatial analyst toolbox, is performed on the slope raster to assign any values with less than the minimum slope value a "NoData" value instead. This step mimics the "minslope" input in PISA-m. Finally, the slope raster units are converted from degrees to radians by dividing the slope by 57.29578, or 180/π. The trigonometric functions in the FS equation are their own tools in ArcGIS, and expect the angles input into those tools to be in radians, and the resulting raster is used in the FS equation.

Next, the code converts the soil feature class into a raster one soil unit at a time, using the "*Feature to Raster"* conversion tool. First, a temporary layer of the feature class is created, and each row of the attribute table, or each soil unit, is selected one at a time. Since the "*Feature to Raster"* tool performs its analysis only on the selected features, the result is a single unit being converted to a raster at a time. The "SoilInteger" attribute field is used to assign raster values to each soil unit. Once every individual unit raster is created, the "*Cell Statistics"* tool is used to combine the individuals into a single

soil unit raster. The individual unit rasters are then divided by the value of the unit raster, resulting in a raster with the value of one. These rasters are used later in the code to assign the correct soil properties to each soil unit. A list of the unit rasters is used to keep track of how many and what order the units appear, adding each unit raster to the list as it's made. The length of this list, or number of units converted, is also defined after all units have been converted, and is also used in assigning property values.

3.3 NDVI to Estimate Root Cohesion and Surcharge

The next portion of the code allows the user to either input their own root cohesion and surcharge data, create that information using the NDVI, or ignore those two variables all together. GIS-TISSA is written to accept any of these three combinations of inputs. Since NDVI is considered the overall productivity and biomass (Pettorilli, 2005), it can be used to estimate root cohesion and tree surcharge. First, the NDVI is created from the Red and NIR rasters input in the GUI. It is worth noting at this point that ArcGIS tools must make one raster at a time for each operation in an equation. For example, the numerator, denominator, and division of the NDVI equation (Equation 2) are each their own raster in GIS-TISSA. Creating a raster for each operation in an equation is a common occurrence the program.

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{2}$$

After the NDVI raster is created, the "*Reclassify*" tool is used twice: 1) Classify NDVI values above 0 to 0.1 and 0.1 to 1, respectively to 0 and 1; 2) Classify NDVI values of -1

to 0.3, 0.3 to 0.6, and 0.6 to 1, respectively to 0, 500, and 2000 N/m^2. These two reclassifying steps use the classes as explained by Jaafari et al. (2014) for barren area, a class between shrub and forests, and forest.

The result from the first reclassify step is used to identify which areas are vegetated and in turn should have root cohesion applied to them. Using Equation 3 to calculate root cohesion from NDVI (Huang et al., 2006). Uniformly distributed rasters of C_{min} and C_{int} are first created using the "*Create Random Raster*" tool from the data management toolbox. These rasters are then given a spatial reference and multiplied by the vegetated area raster created previously. Equation 3 is then performed one operation at a time until a root cohesion raster is created, covering only the regions in the study area that are represented as vegetated by the NDVI.

$$C = C_{min} + C_{int} * \frac{NDVI+1}{2} \quad (3)$$

Reclassify step 2 characterize any area that can has vegetation considered heavy enough to warrant being assigned a surcharge value. Any NDVI values less than 0.3, which represent as large as shrubs or grass, were not considered because they do not have enough surcharge to influence slope stability (Norris et al., 2008, Jaafari et al., 2014). The values of 500 and 2000 are the minimum and maximum surcharge from forests with 30 to 50 meter tree height (Greenwood et al., 2004).

3.5 Water Unit Weight

Since water unit weight is considered constant in the PISA-m model, GIS-TISSA uses the "*Create Constant Raster*" tool to create a constant raster with the value of 9810 N/m^3.

This raster is then assigned a spatial reference, and is eventually used in lieu of the water unit weight variable in the FS equation. It is important to note that since the units of water unit weight are N/m^3, those units are expected in the rest of the soil and tree properties.

3.6 Soil Parameter Loops

The FOSM nature of PISA-m becomes even more relevant in GIS-TISSA, which creates the soil property rasters from scratch. Instead of creating a few hundred rasters for each property to correctly assess a Monte-Carlo approach, FOSM allows for only one raster to be created for each property, greatly improving run times.

The last process of GIS-TISSA creates rasters for the soil parameters friction angle, cohesion, moist unit weight, and saturated unit weight. Using for-loops, lists, and functions from the "*Itertools"* and "*CSV"* packages of ArcPy, the CSV file created in the preprocessing is read row by row. The mean value is read first, and a constant raster is created with the mean value over the entire extent covered by the AOI feature class. The same is done for the standard deviation value. These rasters are then multiplied by their respective individual soil unit raster, resulting in soil unit rasters with their soil parameter values assigned. The next row in the CSV is read, and the process is repeated until a counter reaches the value equal to the length of the soil unit list created when creating the soil units. As either a mean of standard deviation raster is created, it is added to a list of all rasters for that statistic. After all soil units have been used, these lists are used in combination with the Cell Statistics tool to combine all mean or standard deviation rasters of each soil property. The entire process above is repeated

for each soil property, in the specific order of Friction Angle, Soil Cohesion, Saturated Unit Weight, and Moist Unit Weight. The properties must be entered in this order in the CSV file because of the order that the units appear in the code.

3.7 Factor of Safety Calculations

Finally, the desired statistics for FS can be calculated, starting with the average, which is calculated using the mean value of each property. The tools in the ArcGIS math toolset can only perform one operation at a time, saving the result as a new raster. For example, calculating the mean FS (Equation 4) in GIS-TISSA uses 21 individual intermediate equations, and therefore rasters. The variance of the FS is found by calculating the partial derivative of Equation 4 with respect to each variable in the equation. This result in 10 separate equations, one for each property (Appendix II), determined using the software Wolfram Mathematica. The variance of each property is found by squaring their standard deviations, and the partial derivative equations are squared as well. Each property's variance is then multiplied by its respective partial derivative square, and the values for each property are added, resulting in FS variance (Equation 5). The square root of the variance is taken, resulting in the standard deviation of FS. Equation 5 creates 49 more temporary rasters before the standard deviation raster is calculated. Water unit weight considered constant over the entire area, and so is also dropped from the variance equation.

$$FS = \frac{c_r + c_s + (q_t + \gamma_m D + (\gamma_{sat} - \gamma_w - \gamma_m) H_w D)(\cos\beta)^2 \tan\varphi}{(q_t + \gamma_m D + (\gamma_{sat} - \gamma_m) H_w D) \sin\beta \cos\beta} \quad (4)$$

where:

c_r = root cohesive strength

c_s = soil cohesive strength

q_t = uniform surcharge due to weight of vegetation

γ_m = unit weight of moist soil above phreatic surface

γ_{sat} = unit weight of saturated soil below phreatic surface

γ_m = unit weight of water

D = thickness of soil above slip surface

H_w = height of phreatic surface above slip surface, normalized to soil thickness

β = slope angle

φ = angle of internal friction

$$s_{FS}^2 = \sum_i \left(\frac{\partial FS}{\partial x_i}\right)_{x_{avg}}^2 * s_{x_i}^2 \quad (5)$$

where:

$s_{x_i}^2$ = variance of the i[th] independent variable

s_{FS}^2 = variance of the FS

The probability of sliding and reliability index can only be calculated after the mean and variance have been calculated. Probability is equivalent to the cumulative distribution function of the mean FS equation evaluated at FS = 1 (Equation 6).

$$Prob\{FS \leq 1\} = CDF(FS(1)) \quad (6)$$

PISA-m assumes the probability is lognormally distributed, which is reflected in the use of a lognormal probability equation (Equation 7), also used in GIS-TISSA. As mentioned when discussing Equation 3, the probability is evaluated at FS = 1 (x_{crit}). Equations 7.1 through 7.7 are intermediate equations that are eventually substituted into Equation 7.

$$logprob = 0.5 * \left(1 + erf\left(\frac{\log(x_{crit}) - \mu}{1.414214\sigma}\right)\right) \quad (7)$$

$$\mu = \log(mean) - \frac{\sigma^2}{2} \quad (7.1)$$

$$\sigma = \sqrt{\log\left(\frac{\frac{var}{mean}}{mean} + 1\right)} \tag{7.2}$$

The calculated mean and variance of FS are used to find μand σ (Equations 7.1, 7.2), while x in Equation 7.3 represents the fraction in the error function (erf) parenthesis. The variables t and x are used to calculate the complement of the error function (Equation 7.4), $erfc$.

$$t = \frac{1}{1+0.5|x|} \tag{7.3}$$

$$erfc = t * e^{z} \tag{7.4}$$

where:

$$\begin{aligned} z = -|x|^2 &- 1.26551223 + t\,(1.00002368 + t\,(0.37409196 + t\,(0.09678418 + \\ &t\,(-0.1828806 + t\,(0.27886807 + t\,(-1.13520398 + t\,(1.4885187 + \\ &t\,(-0.82215223 + t\,(0.17087277)))))))) \end{aligned} \tag{7.5}$$

Equations 7.4 and 7.5 are an estimation of the complement of the error function using the Chebyshev approximation (Press et al., 1992). A conditional statement is performed on the complementary error function (Equation 7.6) before finally calculating the value of the error function at x (Equation 7.7).

$$erfc = \begin{cases} 2 - erfc, & if\ x < 0 \\ erfc, & otherwise \end{cases} \tag{7.6}$$

$$erf = 1 - erfc \tag{7.7}$$

The *erf* value calculated in Equation 7.7 can be directly substituted for the error function and its parenthesis in Equation 7.

The reliability index is simpler to calculate:

$$RI = \frac{FS_{avg}-1}{s_{FS}} \quad (8)$$

where:

s_{FS} = standard deviation of FS

Equation 7 and its intermediate equations result in 42 intermediate rasters. There are also five rasters that are created as a product of the conditional Equation 7.6, for a total of 47 rasters created in calculating the probability of sliding in its entirety. Reliability is calculated using three rasters, including the mean and standard deviation outputs.

3.8 Graphical User Interface

A Graphical User Interface (GUI) has been created to allow for easier use of the GIS-TISSA. In the GUI, there are the following inputs: 1) DEM (raster), 2) Soils (feature class) 3) Red Band (raster), 4) NIR Band (raster), 5) Root Cohesion (raster), 6) Surcharge (raster), 7) Soil Depth (raster), 8) Phreatic ratio (raster), 9) AOI (feature class), 10) CSV containing soil property values, 11) Minimum slope value, and 12) Processing Resolution, i.e., resampling resolution (raster). Next, the name of the output geodatabase is specified, the location where all calculated rasters will be saved to. Two more optional check boxes named “Probability” and “Reliability” are present. If either or both is checked, the specified statistic will be created along with the mean and standard deviation of FS that is automatically created. Finally, the snap raster is defined (Figure 5). As mentioned previously, the Red and NIR, and root cohesion and surcharge rasters are optional and either pair can be interchanged with the other. If none of the four rasters are specified, root cohesion and surcharge are both given the value of zero.

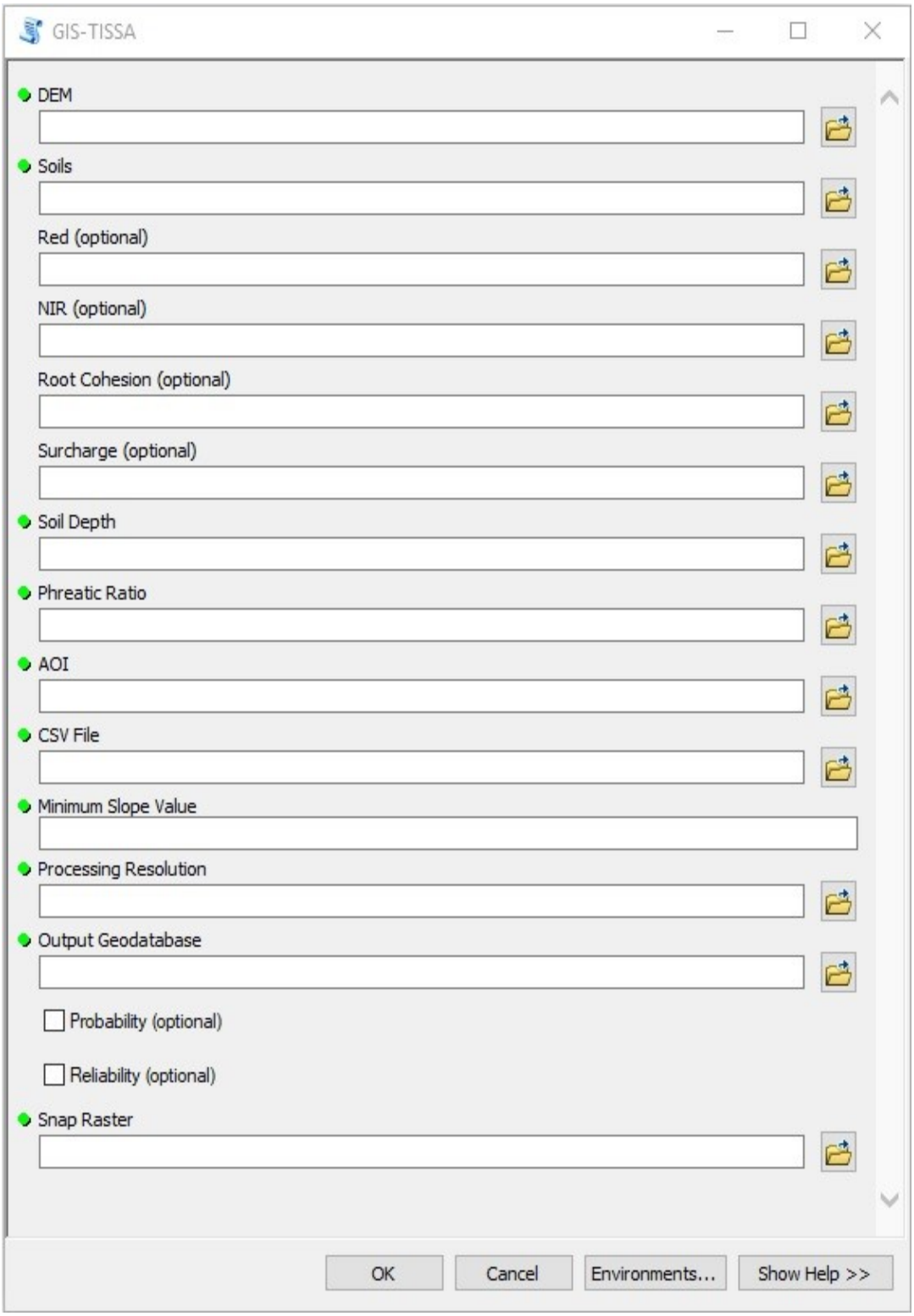

Figure 5: Screenshot of GUI used to provide GIS-TISSA with the necessary inputs.

4. Verification

PISA-m comes with example input rasters that can be input into GIS-TISSA. The code behind GIS-TISSA was changed to assign every FS variable as PISA-m does; four soil parameter loops were added for soil depth, phreatic ratio, root cohesion, and surcharge. An extra CSV file was also used to assign the tree property values. Slight differences between the results of both models can be attributed to the way slope is calculated in both models.

PISA-m calculates the mean slope from a DEM using a finite difference approximation, and uses the four nearest neighbors to a center pixel (Equation 9). The slope is only calculated for any pixel where the four neighboring pixels have a value other than "NoData" (Figure 6). If the window used in gathering DEM values encounters "NoData" or similar value within the window,

the slope for that center pixel will not be calculated.

$$\beta_{r,c} = \arctan\left(\frac{\sqrt{\left(z_{r,c+1} - z_{r,c-1}\right)^2 + \left(z_{r+1,c} - z_{r-1,c}\right)^2}}{2dx}\right) \quad (9)$$

where:

$\beta_{r,c} =$ slope of center pixel

$r =$ row number of center pixel

$c =$ column number of center pixel

$z =$ elevation value of pixel represented by row and column numbers

$dx =$ resolution of DEM raster

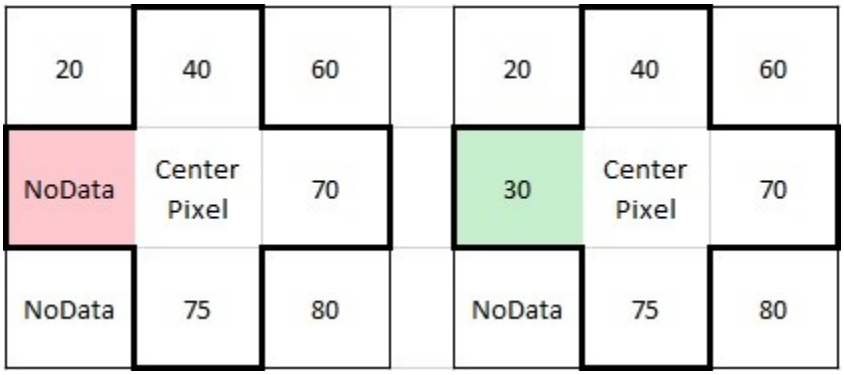

Figure 6: Example of how "NoData" values around a center pixel value effect the calculation of slope. Thick borders denote the shape of the value gathering window. PISA-m will not calculate the slope of the center pixel on the left, but will do so on the right.

To verify that the slope calculation was indeed the only difference between PISA-m and GIS-TISSA, 20 points were randomly chosen throughout the area covered by the data, recording the slope and mean FS values for each model. Further, a manual hand calculation of the FS value was performed using the PISA-m equations and the GI-TISSA equations. A scatter plot of the slope calculation from ArcGIS and PISA-m is shown in Figure 7. The R-squared and RMSE values for this scatter plot are 0.9782 and 1.514, respectively. A scatter plot of the hand calculations with the corresponding model output is presented in Figure 8. It is observed from this plot that the hand calculations matched their respective model outputs, showing that the models were both performing as designed, with only the slope values in the hand calculations differing between the two calculations. This difference in slope is propagated through to the calculation of FS, resulting in slightly different values of FS (Figure 9). An interesting occurrence to note is when slope decreases, the potential difference between the FS in

both models increased (Figure 10). Especially, when the slopes are less than about 10 degrees the difference in FS values between PISA-m and GIS-TISSA are high. The comparable results of FS from PISA-m and GIS-TISSA verify the validity and applicability of GIS-TISSA.

Figure 7: Graph comparing the slopes calculated by PISA-m and GIS-TISSA.

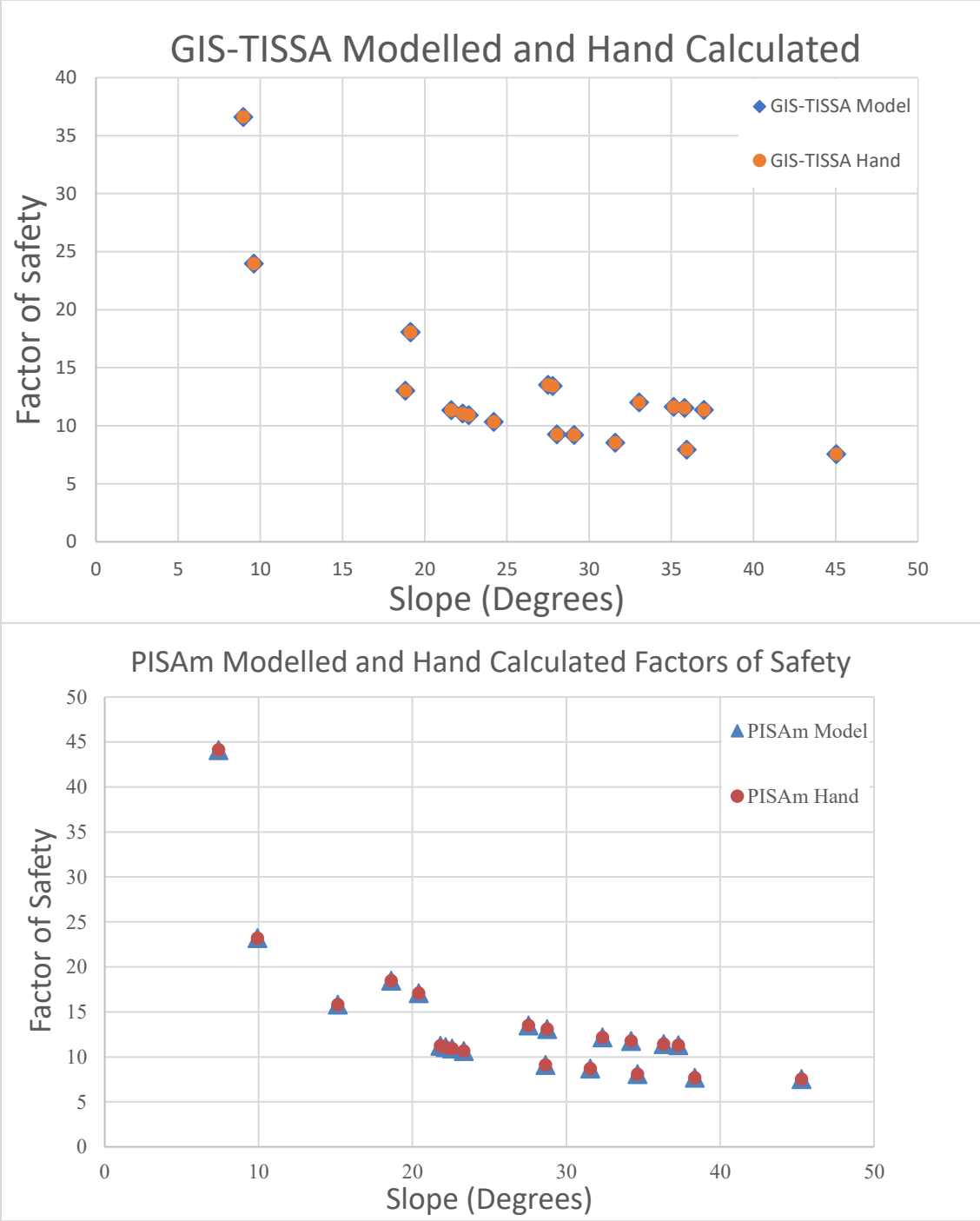

Figure 8: The two graphs above depict hand calculated and model calculated Factors of Safety for GIS-TISSA (Top) and PISA-m (Bottom). In both cases, values for the same model were calculated to be within four decimal places of their predicted value.

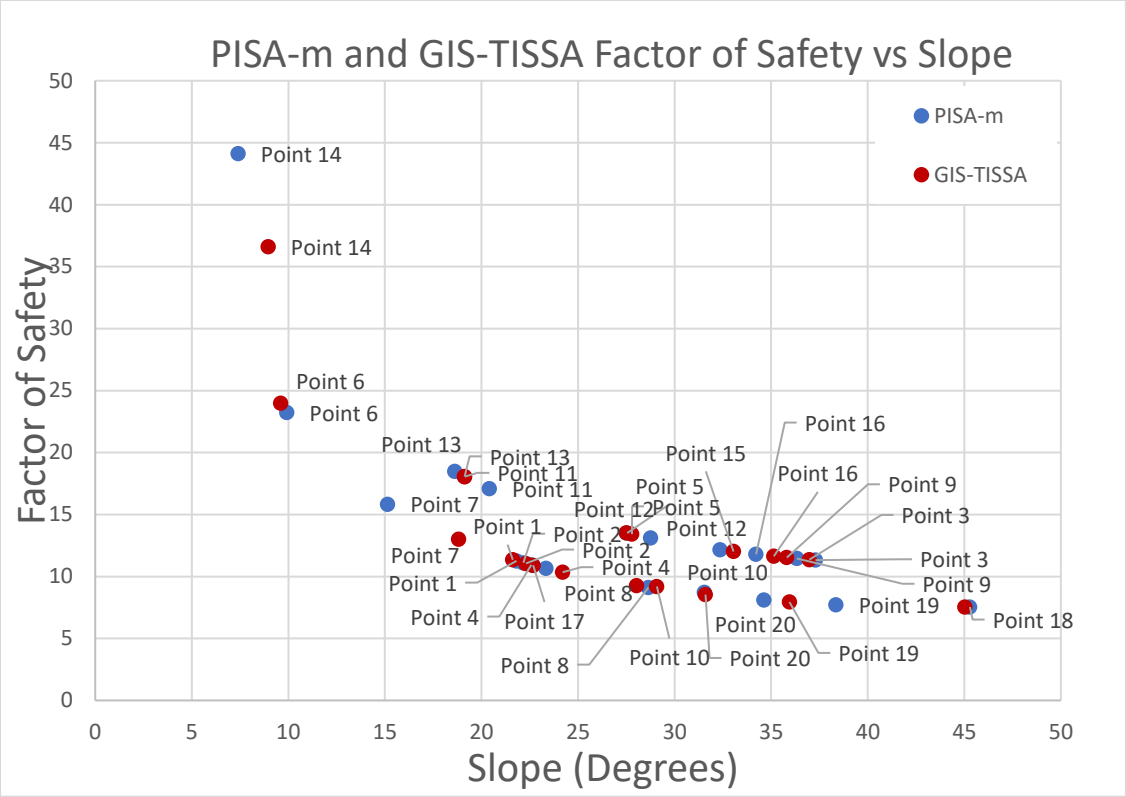

Figure 9: Chart showing the difference in Factor of Safety between PISA-m and GIS-TISSA caused by different slope values for the same point. A few of the same random point had near exact Factors of Safety between both models which caused overlapping charted points, and so one label was used. Different random points that overlapped are labelled for both points. For example, Point 11 of GIS-TISSA and Point 13 of PISA-m fall one over the other, so both are labelled for clarity.

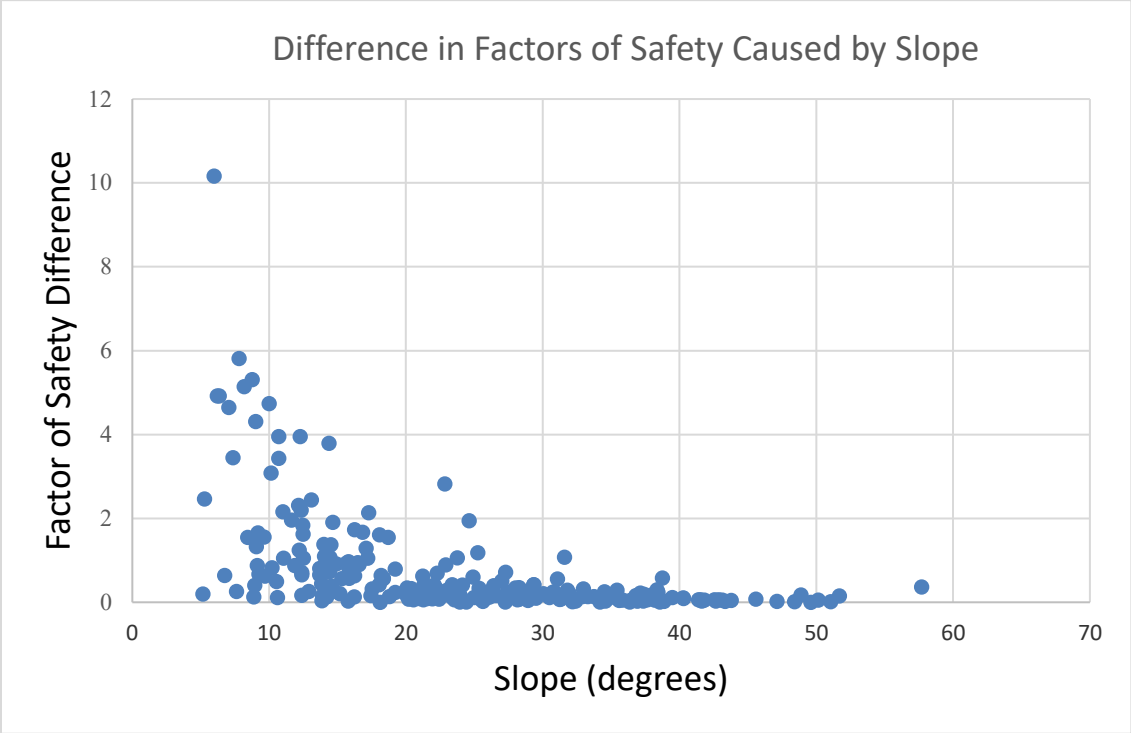

Figure 10: Chart showing the effect that decreasing slope values have on the potential difference in FS between PISA-m and GIS-TISSA. As slope decreases, the variance in difference between the two models increases. Slope values are calculated by GIS-TISSA.

5. Kannur Case Study

PISA-m and GIS-TISSA were both used to model the average FS of slope stability for rectangular bounding box around the district of Kannur, India. The results were compared with each other to test GIS-TISSA's ability in a real world application.

5.1 Data and Preprocessing

Additional data processing was needed to obtain the necessary inputs to calculate the FS in the Kannur region so to match the data used for the same area in PISA-m. The soil unit areas were digitized from the soil maps by first georeferencing them using the tools available in ArcGIS. The polygons of each soil were then traced and added to a new

polygon feature class. These soil classes were combined as per USCS classification, reducing the number of soils in the area to five from 16. A new attribute field was added to the feature class, called “SoilInteger”, containing the soil mapping unit for each soil. Finally, the Dissolve tool was used on the feature class, combining any rows in the attribute table with the same soil mapping unit into one feature.

Soil property values from field measurements conducted in the region (Sajin Kumar, personal communication, July 19, 2017). The mean values for properties are shown in Table 2, while standard deviation values are shown in Table 3.

Table 2: Mean soil property values for the region of Kannur.

	Soil Unit				
Property	1	2	3	4	5
Friction Angle (degrees)	34.5	24	32.22	27	32.8
Soil Cohesion (N/m^2)	0	18000	32950	32361	26478
Depth (m)	12	5.5	11.2	8.3	6.5
Moist Unit Weight (N/m^2)	20787	18165	21000	20459	22752
Saturated Unit Weight (N/m^2)	19500	17850	20915	18296	15058

Table 3: Standard deviation soil property values for the region of Kannur.

	Soil Unit				
Property	1	2	3	4	5
Friction Angle (degrees)	2.6	4.04	1.82	2.31	3.93
Soil Cohesion (N/m^2)	0	4041.45	7647.29	0	8492.82
Depth (m)	5.2	3.18	8.08	7.22	2.89
Moist Unit Weight (N/m^2)	852.63	844.83	1621.78	567.25	567.54
Saturated Unit Weight (N/m^2)	1154.7	1760.92	8580.58	1645.45	1331.08

A soil depth by soil class raster was created using the soil depth properties in Tables 2 and 3, while the phreatic ratio raster, 0.5, was kept constant for the entire area (Sajin Kumar, personal communication, July 19, 2017).

Red and NIR bands of Landsat imagery was used to create an NDVI for the region. The NDVI was classified using the ranges discussed in Holben (1986). Root cohesion and surcharge values were assigned to the NDVI classes following the studies of Kuriakose and van Beek (2011), and are found in Table 4 and 5 (Sajin Kumar, personal communication, July 19, 2017).

Table 4: Mean tree property values for the region of Kannur, assigned by NDVI class.

	Soil Unit				
Property	1	2	3	4	5
Root Cohesion (N/m^2)	0	4762	4762	4762	4762
Surcharge (N/m^2)	0	1190	1190	1190	1190

Table 5: Standard deviation tree property values for the region of Kannur, assigned by NDVI class.

	Soil Unit				
Property	1	2	3	4	5
Root Cohesion (N/m^2)	0	5842.21	5842.21	5842.21	5842.21
Surcharge (N/m^2)	0	481.22	481.22	481.22	481.22

5.2 Kannur Results

PISA-m and GIS-TISSA were used to model the average FS the Kannur region. The results are similar to that of the comparison of the example data. FS values between the two models are different, and are attributed to the difference in slope calculation between the two models (Figure 11). In this figure, any point on the chart has a companion point

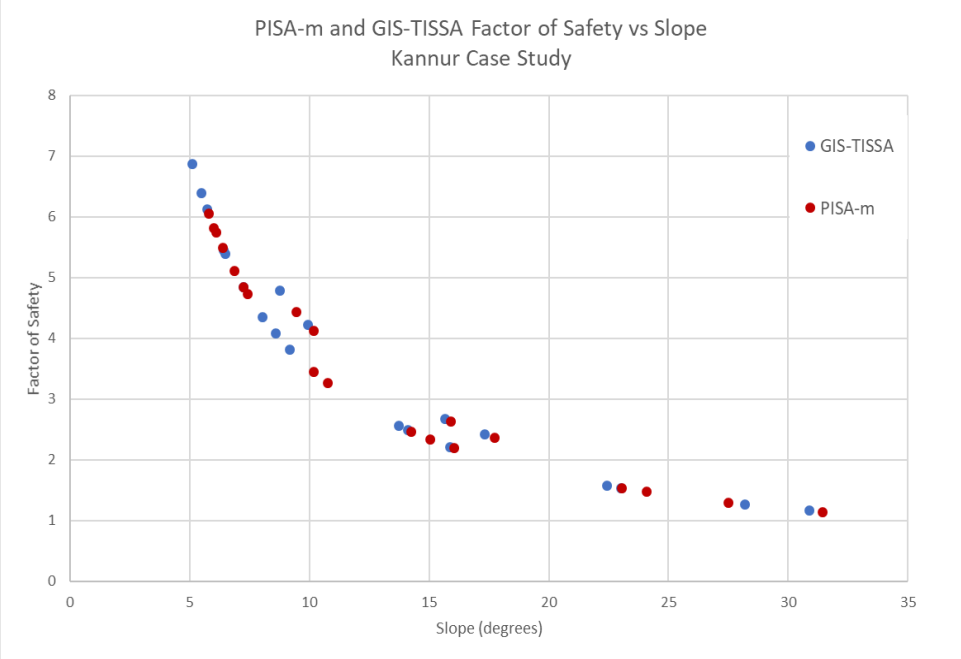

Figure 11: Chart showing the difference in Factor of Safety values predicted by both models. Each point has a companion point from the other model nearby.

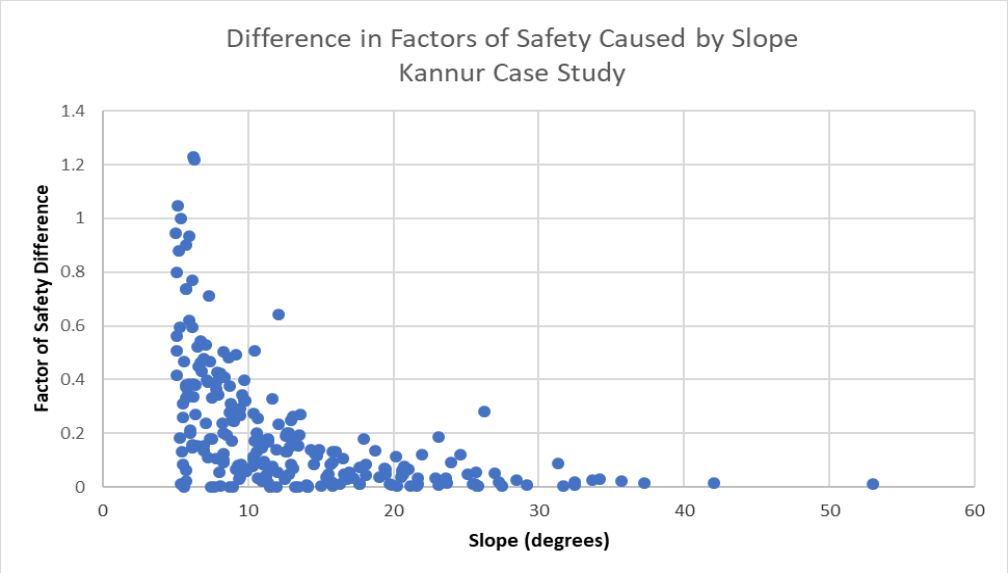

Figure 12: Chart showing the variation in the difference of Factor of Safety between both models as slope increases. The slope values are as calculated by GIS-TISSA.

nearby from the other model. For example, between 30 and 35 degrees slope there are only two points, one for each model. The variation in the difference in slope shows a similar trend as before as well (Figure 12). The similarity of results between the case study and the example further prove the FS prediction accuracy of GIS-TISSA in a real-world application of the model.

A comparison of the FS predicted by both models over the entire region shows slight differences in maximum and minimum values, but otherwise shows no noticeable differences in the spatial prediction of FS (Figure 13).

6. Conclusion

This paper has outlined the creation and running of a GIS based algorithm for FS. The new model was run on example data from the model it was based on, as well as data created for a real world application in Kannur. During testing, the way slope is calculated and its effect was brought into focus during analysis. Although the difference in slope calculations is minor, the effect is transferred through to FS calculation, at times causing large differences in FS. Zhou and Liu (2003) discuss in detail six slope equations, including that used in PISA-m, and lists advantages and disadvantages of using each in certain situations. However, none of these equations are mentioned to have usefulness, or lack thereof, when calculating FS. The slope algorithm used in ArcGIS is yet another way to calculate slope. Taking slope calculations into account, GIS-TISSA has shown that it has the potential to estimate slope stability at the same accuracy as PISA-m.

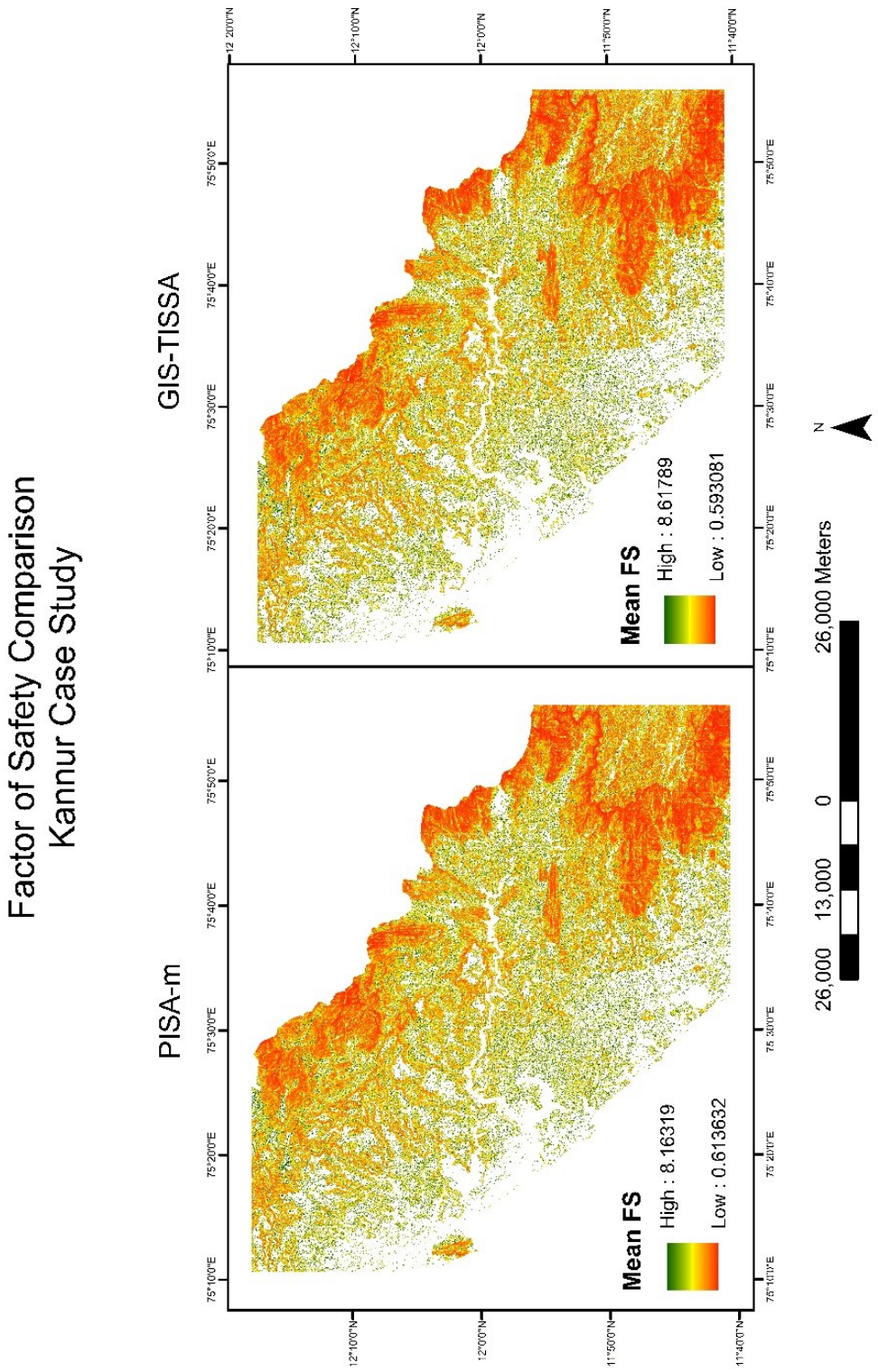

Figure 13: Comparison of PISA-m and GIS-TISSA FS predictions in the region in and directly surrounding Kannur.

Furthermore, GIS-TISSA is written in an environment that the geospatial community commonly uses. While there are limitations, such as run time, and lack of property distributions, with GIS-TISSA now, changes made in the future can alleviate these. As research in the community continues, GIS-TISSA can be remolded to accommodate more variables, improving it accuracy.

7. Limitations

There are other factors that should be considered between PISA-m and GIS-TISSA. PISA-m can calculate FS for the entire Kannur region in less than a minute, while GIS-TISSA takes around 10 minutes. This is because of the way calculations are performed within code of the models. PISA-m calculates values one pixel at a time, regardless of the complexity of the equation, allowing the model to calculate only the rasters needed. GIS-TISSA calculates for all pixels in an entire raster at once, forcing the model to calculate one raster per math operation in the equation. GIS-TISSA also expects more inputs than PISA-m, requiring the user to obtain more before the model can be run. However, GIS-TISSA is also able to ingest continuous raster data, as opposed to PISA-m that can only accept discrete data. Continuous data may be used to better calculate small scale slope stability, and more work could be done to compare the use of both types of data.

GIS-TISSA can only calculate the statistics for slope stability using normally distributed soil and tree property values, while PISA-m can use six other distributions, reducing the ability to tune the model to the correct property values.

GIS-TISSA is currently initiated from a ArcToolbox, which is created using the setup wizard in ArcGIS, and comes with its own limitations. The model script is not embedded in the tool, and alternatively must be included with the toolbox as a separate file as it is shared. Furthermore,

the tool will not run on any drive that does not have the same drive letter as the original drive it was created on. This cripples the ability to share the tool among the community.

8. Future Work

GIS-TISSA is planned on being improved upon in the future. First, support will be added for the other six distributions in PISA-m, as well as the ability to choose the distribution for each property in each of the tree and soil types. These distributions include none, empirical, uniform, triangular, extreme, and beta-pert.

ArcGIS allows tools to be created while embedding the code directly into the toolbox. Called a Python toolbox, this allows the model to be shared without the toolbox limitations discussed previously. Using a Python toolbox also allows greater customization of the GUI, allowing for more complex inputs to be entered while keeping the GUI easy to use and understand.

Other changes may be made to GIS-TISSA as research, old or new, is found relevant to the model and its ability to predict slope stability.

References

Ahmed, A., Ugai, K., and Yang, Q.Q. 2012. Assessment of 3D slope stability analysis methods based on 3D simplified Janbu and Hovland methods. International Journal of Geomechanics 12(2):81-89.

Barlow, J., Y, Martin., and S.E. Franklin 2003. Detecting translational landslide scars using segmentation of Landsat ETM+ and DEM data in the northern Cascade Mountains, British Columbia. Canada Journal of Remote Sensing 29:510-517.

Baum, R.L., Savage, W.Z., and Godt, J.W. 2008. TRIGRS-A fortran program for transient rainfall infiltration and grid-based regional slope-stability analysis, version 2.0. U.S. Geological Survey Open-File Report, 2008-1159, 75p

Chang, K. 2016. Introduction to geographic information systems. University of Idaho, Eighth edition.

Dai, F.C., and Lee, C.F. 2002. Landslide characteristics and slope instability modeling using GIS, Lantau Island, Hong Kong. Geomorphology 42:213-228.

Dai, Z., Wang, F., Huang, Y., Song, K., and Iio, A. 2016. SPH-based numerical modeling for the post-failure behavior of the landslides triggered by the 2016 Kumamoto earthquake. Geoenvironmental Disasters 3:24.

Dietrich, W., Reiss, R., Hsu, M., and Montgomery, D. 1995. A process-based model for colluvial soil depth and shallow landsliding using digital elevation data. Hydrological Processes 9:383-400

Gariano, S. L., & Guzzetti, F. (2016). Landslides in a changing climate. Earth-Science Reviews, 162, 227-252.

Ghosh, S., Carranza, E.J.M., Westen, C.J., and Jetten, V.G., Bhattacharya, D.N. 2011. Geomorphology 131:35-56.

Gomez, H., and Kavzoglu, T. 2005. Assessment of shallow landslide susceptibility using artificial neural networks in Jabonosa River Basin, Venezuela. Engineering Geology 78:11-27

Greenwood, J., Norris, J., and Wint, J. 2004. Assessing the contribution of vegetation to slope stability. Geotechnical Engineering 157(4).

Guzzetti, F. 2000. Landslide fatalities and the evaluation of landslide risk in Italy. Engineering Geology 58:89-107.

Haneburg, W. 2007. PISA-m: Map-Based Probabilistic Infinite Slope Analysis User Manual. Haneburg Geoscience.

Highland, L.M., and Bobrowsky, P. 2008. The landslide handbook: A guide to understanding landslides. U.S. Geological Survey Circular 1325.

Holben, B.N., 1986. Characteristics of maximum-value composite image from temporal AVHRR data. International Journal of Remote Sensing. 7(11): 1417-1434

Huang, J., Kao, S., Hsu, M., and Lin, J. 2006. Stochastic procedure to extract and to integrate landslide susceptibility maps: an example of mountainous watershed in Taiwan. Natural Hazards and Earth System Sciences 6: 803-805

Jaafari, A., Najafi, A., Pourghasemi, H.R., Rezaeian, J., and Sattarian, A. 2014. GIS-based frequency ratio and index of entropy models for landslide susceptibility assessment in the Caspian forest, northern Iran. International Journal of Environmental Science and Technology.

Jaafari, A., Najafi, A., Pourghasemi, H.R., Rezaeian, J., Sattarian, A., and Ghajar, I. 2015. Planning road networks in landslide-prone areas: A case study from the northern forests of Iran. Land Use Policy 47: 198-208

Jain, S.K., and Kumar, V. 2012. Trend analysis of rainfall and temperature data for India. Current Science 102(1): 37-94

Kuriakose, S. 2006. Effect of vegetation on debris flow initiation: Conceptualization and parameterization of a dynamic model for debris flow initiation in Tikovil River Basin, Kerala, using PCRaster. International Institute for Geo-information Science and Earth Observation, Indian Institute of Remote Sensing.

Kuriakose, S.L., van Beek, L.P.H., 2011. Plant root strength and slope stability. In: Encyclopedia of Agrophysics. 622-627.

Madhyamam.com. 2012. Landslides caused by cloudburst: GSI. Kerala. http://www.madhyamam.com/en/node/3963

National Bureau of Soil Science and Land Use Planning (NBSS), 1996. Soil Map of Kerala.

Norris, J.E., Stokes, A., Mickovski, S.B., Cammeraat, E., Beek, R., Nicoll, B.C., and Achim, A. 2008. Slope stability and erosion control: ecotechnological solutions. Springer.

Office of the Registrar General and Census Commissioner. 2011. CensusInfo India 2011 population totals.

Pelletier, J.D., Broxton, P.D., Hazerberg, P., Zeng, X., Troch, P.A., Niu, G., Williams, Z., Brunke, M.A., and Gochis, D. 2016. A gridded global data set of soil, intact regolith, and sedimentary thicknesses for regional and global land surface modeling. Journal of Advances in Modeling Earth Systems 8: 41-65.

Petley, D. 2012. Global patterns of loss of life from landslides. Geology 40:927-930.

Pettorilli, N., Vik, J.O., Mysterud, A., Gaillard, J., Tucker, C.J., and Stenseth, N. 2005. Using the satellite-derived NDVI to assess ecological responses to environmental change. Ecology and Evolution 20: 503-510.

Press, W.H., Teukolsky, S.A., and Vetterling, W.T., Flannery, B.P. 2002. Numerical recipes in C: The art of scientific computing. Cambridge University Press 2.1.

Ramm, F. 2017. OpenStreetMap data in layered GIS format. OpenStreamMap and Geofabrik.

Reid, M.E., Christian, S.B., Brien, D.L., and Henderson, S.T. 2015. Scoops3D-Software to analyze 3D slope stability throughout a digital landscape. U.S. Geological Survey Techniques and Methods, book 14, chapter A1, 218p.

Schaefer, L. N., Lu, Z., & Oommen, T. (2015). Dramatic volcanic instability revealed by InSAR. Geology, 43(8): 743-746.

Schaefer, L. N., Lu, Z., & Oommen, T. (2016). Post-eruption deformation processes measured using ALOS-1 and UAVSAR InSAR at Pacaya Volcano, Guatemala. Remote Sensing, 8(1): 73.

Shi, W., Wang, M., Li, X., Pichel, W.G. 2011. Ocean sand ridge signatures in the Bohai Sea observed by satellite ocean color and synthetic aperture radar measurements. Remote Sensing of Environment 115: 1926-1934

Smith, D. M., Oommen, T., Bowman, L. J., Gierke, J. S., & Vitton, S. J. 2015. Hazard assessment of rainfall-induced landslides: a case study of San Vicente volcano in central El Salvador. Natural Hazards, 75(3): 2291-2310.

The Times of India. 2016. Landslide hits Alakkode in Kannur district. http://timesofindia.indiatimes.com/city/kozhikode/Landslide-hits-Alakkode-in-Kannur-district/articleshow/52983084.cms

Varnes, D.J. 1958. Landslide types and processes. U.S. Geological Survey.

Varnes, D.J. 1978. Slope movement types and processes.

Zezere, J.L., Trigo, R.M., and Trigo, I.F. 2005. Shallow and deep landslides induced by rainfall in the Lisbon region (Portugal): assessment of relationships with the North Atlantic Oscillation. Natural Hazards and Earth System Science, Copernicus Publications on behalf of the European Geosciences Union 5: 331-344.

Zhou, Q., and Liu, X. 2004. Analysis of errors of derived slope and aspect related to DEM data properties. Computers and Geosciences 30:369-378.

Zwissler, B., Oommen, T., & Vitton, S. (2014). A study of the impacts of freeze–thaw on cliff recession at the Calvert Cliffs in Calvert County, Maryland. Geotechnical and Geological Engineering, 32(4), 1133-1148.

APPENDIX I: Example CSV Input

Table 6: Example of a correctly formatted CSV file used in GIS-TISSA to assign soil properties to the correct soil units. The property values in this file are those used in the example parameter file on the page before, only with the soil units reversed. Due to how ArcGIS was adding the "SoilInteger" attribute field to the soil feature class, the reversal of soil units was necessary. The first two columns are the only needed columns, and should start in the top-left most cell if created in Excel.

mean	sd	Property	Soil Unit
32	0.81	Friction Angle	2
33	0.81		1
10000	39	Soil Cohesion	2
5500	42		1
0.1	4	Soil Depth	2
0.1	4		1
0.5	0.1	Phreatic Ratio	2
0.5	0.1		1
20000	26	Saturated Unit Weight	2
21500	22		1
16500	32	Moist Unit Weight	2
18000	25		1

APPENDIX II: Partial Derivative Property Equations

Root and Soil Cohesion

$$\frac{\partial FS}{\partial c_r} = \frac{\partial FS}{\partial c_s} = \frac{\csc\beta \sec\beta}{q_t + D\gamma_m + DH_w(\gamma_{sat} - \gamma_m)}$$

Tree Surcharge

$$\frac{\partial FS}{\partial q_t} = \frac{cot\beta tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{csc\beta sec\beta(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_w-\gamma_m))tan\varphi}{(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m))^2}$$

Moist Unit Weight

$$\frac{\partial FS}{\partial \gamma_m} = \frac{cot\beta(D-DH_w)tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{csc\beta sec\beta(D-DH_w)(c_r+c_s+ cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_w-\gamma_m))tan\varphi}{(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m))^2}$$

Saturated Unit Weight

$$\frac{\partial FS}{\partial \gamma_{sat}} = \frac{Dcot\beta H_w tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{Dcsc\beta sec\beta H_w(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_w-\gamma_m))tan\varphi}{(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m))^2}$$

Water Unit Weight – not used in Factor of Safety variance equation

$$\frac{\partial FS}{\partial \gamma_w} = \frac{Dcot\beta H_w tan\varphi}{q_t + D\gamma_m + DH_w(\gamma_{sat} - \gamma_m)}$$

Soil Depth

$$\frac{\partial FS}{\partial D} = \frac{cot\beta(\gamma_m+H_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{csc\beta sec\beta(\gamma_m+H_w(\gamma_{sat}-\gamma_m))(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi}{((q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m))^2}$$

Phreatic Ratio

$$\frac{\partial FS}{\partial H_w} = \frac{Dcot\beta(\gamma_{sat}-\gamma_m-\gamma_w)tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{Dcsc\beta sec\beta(\gamma_{sat}-\gamma_m)(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi}{(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m))^2}$$

Slope

$$\frac{\partial FS}{\partial \beta} = -\frac{2(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} - \frac{csc^2\beta(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi)}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)} + \frac{sec^2\beta(c_r+c_s+cos^2\beta(q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m-\gamma_w))tan\varphi)}{q_t+D\gamma_m+DH_w(\gamma_{sat}-\gamma_m)}$$

Friction Angle

$$\frac{\partial FS}{\partial \varphi} = \frac{cot\beta csc^2\varphi(q_t + D\gamma_m + DH_w(\gamma_{sat} - \gamma_m - \gamma_w))}{q_t + D\gamma_m + DH_w(\gamma_{sat} - \gamma_m)}$$

APPENDIX III: GIS-TISSA Script

```
# ---------------------------------------------------------------
# Program Title: GIS-TISSA
# Program Author: Jonathon Sanders
# Date Started: 11/21/2016
# Date Updated or Revised: 7/30/2017
# Program Description: GIS-TISSA program
# ---------------------------------------------------------------
import arcpy
from arcpy import env
from arcpy.sa import *
import sys
import traceback
import os
import time
import csv
import itertools
import numpy

start = time.time()

arcpy.CheckOutExtension("Spatial")
arcpy.CheckOutExtension("3D")

# allow script to overwrite existing feature classes
arcpy.env.overwriteOutput = True
arcpy.env.workspace = "G:\\Grad\\Thesis\\India.gdb"
arcpy.env.parallelProcessingFactor = "50%"

DEM_Input = arcpy.GetParameterAsText(0)
Soil_Input = arcpy.GetParameterAsText(1)
Red_Input = arcpy.GetParameterAsText(2)
NIR_Input = arcpy.GetParameterAsText(3)
rootCohesion_Input = arcpy.GetParameterAsText(4)
surcharge_Input = arcpy.GetParameterAsText(5)
soilDepth_Input = arcpy.GetParameterAsText(6)
phreaticZone_Input = arcpy.GetParameterAsText(7)
AOI = arcpy.GetParameterAsText(8)
CSV = arcpy.GetParameterAsText(9)
minSlope = arcpy.GetParameterAsText(10)
Minimum_Resolution = arcpy.GetParameterAsText(11)
Outpath = arcpy.GetParameterAsText(12)
Probability = arcpy.GetParameterAsText(13)
Reliability = arcpy.GetParameterAsText(14)
SnapRaster = arcpy.GetParameterAsText(15)



minRes = arcpy.GetRasterProperties_management(Minimum_Resolution, "CELLSIZEX")
minResResult = minRes.getOutput(0)
# Grab spatial reference for created study area
spatRef = arcpy.Describe(AOI).spatialReference
# Create and add coordinates for study area
extentPoly = str(arcpy.env.workspace) + os.sep + "extent.shp"
extent = arcpy.Describe(AOI).extent
array = arcpy.Array()
array.add(extent.lowerLeft)
array.add(extent.lowerRight)
array.add(extent.upperRight)
```

```
array.add(extent.upperLeft)
polygon = arcpy.Polygon(array)
arcpy.CopyFeatures_management(polygon, "StudyArea")
del array
# Specify spatial reference for study area
arcpy.DefineProjection_management("StudyArea", spatRef)
AOI = "StudyArea"
deg2rad = 57.29577951
rows = arcpy.SearchCursor(AOI)
shapeName = arcpy.Describe(AOI).shapeFieldName
# For loop contains extent for creating new rasters
for corner in rows:
    feat = corner.getValue(shapeName)
    extent = feat.extent
    clip = "%.10f %.10f %.10f %.10f" % (extent.XMin, extent.YMin, extent.XMax,
extent.YMax)
    # Collect all soils from possibly larger area
    arcpy.MakeFeatureLayer_management(Soil_Input, "AllSoilsLYR")
    allSoilList = []
    with arcpy.da.SearchCursor(Soil_Input, "SoilInteger") as cursor:
        for row in cursor:
            soilClass = int('{0}'.format(row[0]))
            allSoilList.append(soilClass)
        arcpy.AddMessage("Full Area of Soils Collected")
    soilLen = len(allSoilList)
    arcpy.Delete_management("AllSoilsLYR")
    # Clip all inputs by study area
    arcpy.Clip_management(DEM_Input, clip, "DEMClip", AOI, "", "NONE", "")
    arcpy.Clip_management(soilDepth_Input, clip, "soilDepthClip", AOI, "",
"NONE", "")
    arcpy.Clip_management(phreaticZone_Input, clip, "phreaticZoneClip", AOI,
"", "NONE", "")
    arcpy.Clip_analysis(Soil_Input, AOI, "SoilClip")
    if Red_Input and NIR_Input:
        arcpy.Clip_management(Red_Input, clip, "RedClip", AOI, "", "NONE", "")
        arcpy.Clip_management(NIR_Input, clip, "NIRClip", AOI, "", "NONE", "")
        arcpy.Resample_management("RedClip", "RedResamp", minResResult,
"BILINEAR")
        arcpy.Resample_management("NIRClip", "NIRResamp", minResResult,
"BILINEAR")
    if rootCohesion_Input and surcharge_Input:
        arcpy.Clip_management(rootCohesion_Input, clip, "rootClip", AOI, "",
"NONE", "")
        arcpy.Clip_management(surcharge_Input, clip, "surchargeClip", AOI, "",
"NONE", "")
        arcpy.Resample_management("rootClip", "rootResamp", minResResult,
"BILINEAR")
        arcpy.Resample_management("surchargeClip", "surchargeResamp",
minResResult, "BILINEAR")
    # Resample all rasters, give common name for later
    arcpy.Resample_management("DEMClip", "DEMResamp", minResResult, "BILINEAR")
    arcpy.Resample_management("soilDepthClip", "soilDepthResamp", minResResult,
"BILINEAR")
    arcpy.Resample_management("phreaticZoneClip", "phreaticZoneResamp",
minResResult, "BILINEAR")
    arcpy.AddMessage("Rasters Resampled")
    global soilDepth
    soilDepth = "soilDepthClip"
    global phreaticRatio
    phreaticRatio = "phreaticZoneClip"
    # Create Slope
```

```
    slope = Slope("DEMResamp", "DEGREE")
    slope.save("Slope")
    minSlope = Con("Slope", "Slope", "", "Value > %s" % minSlope)
    minSlope.save("minSlope")
    radSlope = Divide("minSlope", deg2rad)
    radSlope.save("radSlope")
    arcpy.AddMessage("Slope Created")
    # Create feature layer of soil for select by attribute
    arcpy.MakeFeatureLayer_management("SoilClip", "SoilsLYR")
    rows = arcpy.da.SearchCursor("SoilsLYR", "SoilInteger")
    clipSoilList = []
    clipSoilRasters = []
    # Create soil class for each soil integer in soil file, named after integer
    arcpy.FeatureToRaster_conversion("SoilClip", "SoilInteger", "SoilClasses",
minResResult)
    for row in rows:
        variable = int("{0}".format(row[0]))
        whereClause = '"SoilInteger" = %d' % variable
        arcpy.SelectLayerByAttribute_management("SoilsLYR", "NEW_SELECTION",
whereClause)
        arcpy.FeatureToRaster_conversion("SoilsLYR", "SoilInteger",
"SoilClass_%d" % variable, minResResult)
        # soilClassList.append("SoilClass_%d" % variable)
        soilIntTo_1 = Divide("SoilClass_%d" % variable, variable)
        soilIntTo_1.save("SoilClassFix_%d" % variable)
        arcpy.AddMessage("Soil Type %d Created" % variable)
        clipSoilRasters.append("SoilClassFix_%d" % variable)
        clipSoilList.append(variable)
    CSVrows = []
    for i, j in enumerate(allSoilList):
        for k in clipSoilList:
            if j == k:
                CSVrows.append(i)
    arcpy.Delete_management("SoilLYR")
    arcpy.AddMessage("Soil Class Rasters Created")
    # NDVI
    if Red_Input and NIR_Input:
        numerator = Minus("NIRResamp", "RedResamp")
        numerator.save("numerator")
        denominator = Plus("NIRResamp", "RedResamp")
        denominator.save("denominator")
        numeratorFloat = Float("numerator")
        numeratorFloat.save("numeratorFloat")
        NDVI = Divide("numeratorFloat", "denominator")
        NDVI.save("NDVI")
        arcpy.AddMessage("NDVI Created")
        # NDVI reclassify
        # Vegetated for root cohesion
        # High and low density for surcharge
        outVegetated = Reclassify("NDVI", "Value",
                                  RemapRange([[-1, 0.1, 0], [0.1, 1, 1]]),
"NODATA")
        outVegetated.save("vegetated")
        shrubForest = Reclassify("NDVI", "Value",
                                 RemapRange([[-1, 0.3, 0], [0.3, 0.6, 1], [0.6,
1, 0]]), "NODATA")
        shrubForest.save("shrubForest")
        forest = Reclassify("NDVI", "Value", RemapRange([[-1, 0.6, 0], [0.6, 1,
1]]), "NODATA")
        forest.save("forest")
        surchargeTest = Reclassify("NDVI", "Value", RemapRange([[-1, 0.3, 0],
```

```
[0.3, 0.6, 500], [0.6, 1, 2000]]),
                                        "NODATA")
        surchargeTest.save("surcharge")
        # Cmin and Cint creation
        arcpy.CreateRandomRaster_management(Outpath, "Cmin", "INTEGER 0 20000",
clip, minResResult)
        arcpy.CreateRandomRaster_management(Outpath, "Cint", "INTEGER 0 30000",
clip, minResResult)
        arcpy.DefineProjection_management(Outpath + "\\" + "Cmin", spatRef)
        arcpy.DefineProjection_management(Outpath + "\\" + "Cint", spatRef)
        minCohesion = Times(Outpath + "\\" + "Cmin", "vegetated")
        minCohesion.save("MinCohesion")
        intCohesion = Times(Outpath + "\\" + "Cint", "vegetated")
        intCohesion.save("IntCohesion")
        arcpy.Delete_management(Outpath + "\\" + "Cmin")
        arcpy.Delete_management(Outpath + "\\" + "Cint")
        # Root Cohesion calculations
        numeratorCoh = Plus("NDVI", 1)
        numeratorCoh.save("numeratorCoh")
        parenthesis = Divide("numeratorCoh", 2)
        parenthesis.save("parenthesis")
        multiply = Times("parenthesis", "IntCohesion")
        multiply.save("second")
        rootCohesionRaster = Plus("second", "MinCohesion")
        rootCohesionRaster.save("rootCohesion")
        global rootCohesion
        rootCohesion = "rootCohesion"
        global surcharge
        surcharge = "surcharge"
    if rootCohesion_Input and surcharge_Input:
        global rootCohesion
        rootCohesion = rootCohesion_Input
        global surcharge
        surcharge = surcharge_Input
    if not Red_Input and not NIR_Input and not rootCohesion_Input and not
surcharge_Input:
        rootCo = CreateConstantRaster(0, "FLOAT", minResResult, clip)
        rootCo.save("rootCohesion")
        arcpy.DefineProjection_management("rootCohesion", spatRef)
        surcharge = CreateConstantRaster(0, "FLOAT", minResResult, clip)
        surcharge.save("surcharge")
        arcpy.DefineProjection_management("surchage", spatRef)
        global rootCohesion
        rootCohesion = "rootCohesion"
        global surcharge
        surcharge = "surcharge"
    arcpy.AddMessage("Surcharge Created")
    arcpy.AddMessage("Root Cohesion Created")
    # Constant raster for water unit weight
    H2OUW = CreateConstantRaster(9810, "FLOAT", minResResult, clip)
    H2OUW.save("waterUW")
    arcpy.DefineProjection_management("waterUW", spatRef)
    arcpy.AddMessage("Water Unit Weight Created")
    # Friction Angle
    FAmeanList = []
    FAsdList = []
    FAList = []
    statList = []
    with open(CSV) as csvfile:
        for row in itertools.islice(csv.DictReader(csvfile), 0, soilLen):
            indStat = []
```

```
            mean = float(row["mean"])
            sd = float(row["sd"])
            indStat.append(mean)
            indStat.append(sd)
            statList.append(indStat)
    i = 0
    for props in enumerate(statList):
        for val in CSVrows:
            if props[0] == val:
                mean = props[1][0]
                sd = props[1][1]
                radMean = mean/deg2rad
                radSD = sd/deg2rad
                frictionAngleM = CreateConstantRaster(radMean, "FLOAT",
minResResult, clip)
                frictionAngleM.save("frictionAngleMean_%s" % i)
                frictionAngleS = CreateConstantRaster(radSD, "FLOAT",
minResResult, clip)
                frictionAngleS.save("frictionAngleSD_%s" % i)
                frictionSoilM = Times(clipSoilRasters[i],
"frictionAngleMean_%s" % i)
                frictionSoilM.save("frictionSoilMean_%s" % i)
                frictionSoilS = Times(clipSoilRasters[i], "frictionAngleSD_%s"
% i)
                frictionSoilS.save("frictionSoilSD_%s" % i)
                FAmeanList.append("frictionSoilMean_%s" % i)
                FAsdList.append("frictionSoilSD_%s" % i)
                FAList.extend(["frictionSoilMean_%s" % i, "frictionSoilSD_%s" %
i])
                i += 1
    frictionAngleMean = CellStatistics(FAmeanList, "SUM", "DATA")
    frictionAngleMean.save("FrictionAngleMean")
    frictionAngleSD = CellStatistics(FAsdList, "SUM", "DATA")
    frictionAngleSD.save("FrictionAngleSD")
    arcpy.AddMessage("Friction Angle Created")
    # Cohesion
    CohesionMeanList = []
    CohesionSDList = []
    CohesionList = []
    statList = []
    with open(CSV) as csvfile:
        for row in itertools.islice(csv.DictReader(csvfile), soilLen, (soilLen
* 2)):
            indStat = []
            mean = float(row["mean"])
            sd = float(row["sd"])
            indStat.append(mean)
            indStat.append(sd)
            statList.append(indStat)
    i = 0
    for props in enumerate(statList):
        for val in CSVrows:
            if props[0] == val:
                mean = props[1][0]
                sd = props[1][1]
                cohesionM = CreateConstantRaster(mean, "FLOAT", minResResult,
clip)
                cohesionM.save("cohesionMean_%s" % i)
                cohesionS = CreateConstantRaster(sd, "FLOAT", minResResult,
clip)
                cohesionS.save("cohesionSD_%s" % i)
```

```
                cohesionSoilM = Times(clipSoilRasters[i], "cohesionMean_%s" %
i)
                cohesionSoilM.save("cohesionSoilMean_%s" % i)
                cohesionSoilS = Times(clipSoilRasters[i], "cohesionSD_%s" % i)
                cohesionSoilS.save("cohesionSoilSD_%s" % i)
                arcpy.Delete_management("cohesionMean_%s" % i)
                arcpy.Delete_management("cohesionSD_%s" % i)
                CohesionMeanList.append("cohesionSoilMean_%s" % i)
                CohesionSDList.append("cohesionSoilSD_%s" % i)
                CohesionList.extend(["cohesionSoilMean_%s" % i,
"cohesionSoilSD_%s" % i])
                i += 1
    cohesionMean = CellStatistics(CohesionMeanList, "SUM", "DATA")
    cohesionMean.save("CohesionMean")
    cohesionSD = CellStatistics(CohesionSDList, "SUM", "DATA")
    cohesionSD.save("CohesionSD")
    arcpy.AddMessage("Cohesion Created")
    # Saturated Unit Weight
    SUWMeanList = []
    SUWSDList = []
    SUWList = []
    statList = []
    with open(CSV) as csvfile:
        for row in itertools.islice(csv.DictReader(csvfile), (soilLen * 3),
(soilLen * 4)):
            indStat = []
            mean = float(row["mean"])
            sd = float(row["sd"])
            indStat.append(mean)
            indStat.append(sd)
            statList.append(indStat)
    i = 0
    for props in enumerate(statList):
        for val in CSVrows:
            if props[0] == val:
                mean = props[1][0]
                sd = props[1][1]
                SUWM = CreateConstantRaster(mean, "FLOAT", minResResult, clip)
                SUWM.save("SUWMean_%s" % i)
                SUWS = CreateConstantRaster(mean, "FLOAT", minResResult, clip)
                SUWS.save("SUWSD_%s" % i)
                SUWSoilM = Times(clipSoilRasters[i], "SUWMean_%s" % i)
                SUWSoilM.save("SUWSoilMean_%s" % i)
                SUWSoilS = Times(clipSoilRasters[i], "SUWSD_%s" % i)
                SUWSoilS.save("SUWSoilSD_%s" % i)
                arcpy.Delete_management("SUWMean_%s" % i)
                arcpy.Delete_management("SUWSD_%s" % i)
                SUWMeanList.append("SUWSoilMean_%s" % i)
                SUWSDList.append("SUWSoilSD_%s" % i)
                SUWList.extend(["SUWSoilMean_%s" % i, "SUWSoilSD_%s" % i])
                i += 1
    SUWMean = CellStatistics(SUWMeanList, "SUM", "DATA")
    SUWMean.save("SUWMean")
    SUWSD = CellStatistics(SUWSDList, "SUM", "DATA")
    SUWSD.save("SUWSD")
    arcpy.AddMessage("Saturated Unit Weight Created")
    # Moist Unit Weight
    MUWMeanList = []
    MUWSDList = []
    MUWList = []
    statList = []
```

```
    with open(CSV) as csvfile:
        for row in itertools.islice(csv.DictReader(csvfile), (soilLen * 2),
(soilLen * 3)):
            indStat = []
            mean = float(row["mean"])
            sd = float(row["sd"])
            indStat.append(mean)
            indStat.append(sd)
            statList.append(indStat)
    i = 0
    for props in enumerate(statList):
        for val in CSVrows:
            if props[0] == val:
                mean = props[1][0]
                sd = props[1][1]
                MUWM = CreateConstantRaster(mean, "FLOAT", minResResult, clip)
                MUWM.save("MUWMean_%s" % i)
                MUWS = CreateConstantRaster(mean, "FLOAT", minResResult, clip)
                MUWS.save("MUWSD_%s" % i)
                MUWSoilM = Times(clipSoilRasters[i], "MUWMean_%s" % i)
                MUWSoilM.save("MUWSoilMean_%s" % i)
                MUWSoilS = Times(clipSoilRasters[i], "MUWSD_%s" % i)
                MUWSoilS.save("MUWSoilSD_%s" % i)
                arcpy.Delete_management("MUWMean_%s" % i)
                arcpy.Delete_management("MUWSD_%s" % i)
                MUWMeanList.append("MUWSoilMean_%s" % i)
                MUWSDList.append("MUWSoilSD_%s" % i)
                MUWList.extend(["MUWSoilMean_%s" % i, "MUWSoilSD_%s" % i])
                i += 1
    MUWMean = CellStatistics(MUWMeanList, "SUM", "DATA")
    MUWMean.save("MUWMean")
    MUWSD = CellStatistics(MUWSDList, "SUM", "DATA")
    MUWSD.save("MUWSD")
    arcpy.AddMessage("Moist Unit Weight Created")
# FoS variables
soilCohesion = "CohesionMean"
MUW = "MUWMean"
SUW = "SUWMean"
WUW = "waterUW"
slope = "radSlope"
frictionAngle = "FrictionAngleMean"
# Mean Factor of Safety Equation
num1 = Minus(SUW, MUW)
num1.save("num1")

num2 = Minus("num1", WUW)
num2.save("num2")

num3 = Times(phreaticRatio, soilDepth)
num3.save("num3")

num4 = Times("num2", "num3")
num4.save("num4")

num5 = Times(MUW, soilDepth)
num5.save("num5")

num6 = Plus("num4", "num5")
num6.save("num6")

num7 = Plus(surcharge, "num6")
```

```
num7.save("num7")

num8 = Cos(slope)
num8.save("num8")

num9 = Square("num8")
num9.save("num9")

num10 = Tan(frictionAngle)
num10.save("num10")

num11 = Times("num9", "num10")
num11.save("num11")

num12 = Times("num7", "num11")
num12.save("num12")

num13 = Plus("num12", soilCohesion)
num13.save("num13")

num14 = Plus("num13", rootCohesion)
num14.save("num14")

num15 = Times("num1", "num3")
num15.save("num15")

num16 = Plus("num5", "num15")
num16.save("num16")

num17 = Plus(surcharge, "num16")
num17.save("num17")

num18 = Sin(slope)
num18.save("num18")

num19 = Times("num17", "num18")
num19.save("num19")

num20 = Times("num8", "num19")
num20.save("num20")
num20.save("cohesions")

num21 = Divide("num14", "num20")
num21.save(Outpath + "\\" + "FoS_Mean")
num21.save("FoS_Mean")
arcpy.AddMessage("Mean Factor of Safety Calculated")

num22 = Divide(1, "num10")
num22.save("num22")

num23 = Times("num22", "num10")
num23.save("num23")

num24 = Divide("num23", "num17")
num24.save("num24")

num25 = Divide(1, "num18")
num25.save("num25")

num26 = Divide(1, "num8")
num26.save("num26")
```

```
num27 = Times("num14", "num26")
num27.save("num27")

num28 = Times("num25", "num27")
num28.save("num28")

num29 = Square("num17")
num29.save("num29")

num30 = Divide("num28", "num29")
num30.save("num30")

num31 = Minus("num24", "num30")
num31.save("treeSurcharge")

num32 = Minus(soilDepth, "num3")
num32.save("num32")

num33 = Times("num32", "num23")
num33.save("num33")

num34 = Divide("num33", "num17")
num34.save("num34")

num35 = Times("num32", "num28")
num35.save("num35")

num36 = Divide("num35", "num29")
num36.save("num36")

num37 = Minus("num34", "num36")
num37.save("moist")

num38 = Times("num3", "num22")
num38.save("num38")

num39 = Divide("num38", "num17")
num39.save("num39")

num40 = Times("num3", "num28")
num40.save("num40")

num41 = Divide("num40", "num29")
num41.save("num41")

num42 = Minus("num39", "num41")
num42.save("saturated")

num43 = Divide("num38", "num17")
num43.save("water")

num44 = Times(phreaticRatio, "num2")
num44.save("num44")

num45 = Plus(MUW, "num44")
num45.save("num45")

num46 = Times("num44", "num23")
num46.save("num46")
```

```
num47 = Divide("num46", "num17")
num47.save("num47")

num48 = Times("num1", phreaticRatio)
num48.save("num48")

num49 = Plus("num48", MUW)
num49.save("num49")

num50 = Times("num28", "num49")
num50.save("num50")

num51 = Divide("num50", "num29")
num51.save("num51")

num52 = Minus("num47", "num51")
num52.save("depth")

num53 = Times("num23", "num2")
num53.save("num53")

num54 = Times("num53", soilDepth)
num54.save("num54")

num55 = Divide("num54", "num17")
num55.save("num55")

num56 = Times("num27", "num1")
num56.save("num56")

num57 = Times("num56", soilDepth)
num57.save("num57")

num58 = Divide("num57", "num29")
num58.save("num58")

num59 = Minus("num55", "num58")
num59.save("phreatic")

num60 = Times("num7", "num10")
num60.save("num60")

num61 = Times(-2, "num60")
num61.save("num61")

num62 = Divide("num61", "num17")
num62.save("num62")

num63 = Square("num25")
num63.save("num63")

num64 = Times("num63", "num14")
num64.save("num64")

num65 = Divide("num64", "num17")
num65.save("num65")

num66 = Square("num26")
num66.save("num66")

num67 = Times("num66", "num14")
```

```
num67.save("num67")

num68 = Divide("num67", "num17")
num68.save("num68")

num69 = Minus("num62", "num64")
num69.save("num69")

num70 = Plus("num69", "num68")
num70.save("slope")

num71 = Cos(frictionAngle)
num71.save("num71")

num72 = Divide(1, "num71")
num72.save("num72")

num73 = Square("num72")
num73.save("num73")

num74 = Times("num22", "num73")
num74.save("num74")

num75 = Times("num74", "num7")
num75.save("num75")

num76 = Divide("num75", "num17")
num76.save("frictionAngle")

# Partial derivatives with respect to each variable (Brackets)
# Cohesion partial derivative equation the same for soil and root
# Surcharge has constant values, doesn't have variance, no need to calculate
brackets
cohesionBracket = Square("cohesions")
cohesionBracket.save("cohesionBrack")
moistBracket = Square("moist")
moistBracket.save("moistBrack")
satBracket = Square("saturated")
satBracket.save("satBrack")
depthBracket = Square("depth")
depthBracket.save("depthBrack")
phreaticBracket = Square("phreatic")
phreaticBracket.save("phreaticBrack")
frictionBracket = Square("frictionAngle")
frictionBracket.save("frictionBrack")

soilCohesionVariance = Square("CohesionSD")
soilCohesionVariance.save("soilCohesionV")
moistVariance = Square("MUWSD")
moistVariance.save("moistV")
saturatedVariance = Square("SUWSD")
saturatedVariance.save("saturatedV")
frictionVariance = Square("FrictionAngleSD")
frictionVariance.save("frictionV")

soilCohesionGroup = Times("cohesionBrack", "soilCohesionV")
soilCohesionGroup.save("soilCohesionGroup")
moistGroup = Times("moistBrack", "moistV")
moistGroup.save("moistGroup")
satGroup = Times("satBrack", "saturatedV")
satGroup.save("satGroup")
```

```
frictionGroup = Times("frictionBrack", "frictionV")
frictionGroup.save("frictionGroup")

FoSVariance = CellStatistics(["soilCohesionGroup", "moistGroup", "satGroup",
"frictionGroup"], "SUM", "DATA")
FoSVariance.save("FoS_Var")

FoS_SD = SquareRoot("Fos_Var")
FoS_SD.save("FoS_SD")
FoS_SD.save(Outpath + "\\" + "FoS_SD")
arcpy.AddMessage("Standard Deviation of Factor of Safety Created")

# Error Function
if str(Probability) == "true":

    err1 = Divide("FoS_Var", "FoS_Mean")
    err1.save("err1")

    err2 = Divide("err1", "FoS_Mean")
    err2.save("err2")

    err3 = Plus("err2", 1)
    err3.save("err3")

    err4 = Log10("err3")
    err4.save("err4")

    err5 = SquareRoot("err4")
    err5.save("sigma")

    err6 = Times("sigma", "sigma")
    err6.save("err6")

    err7 = Divide("err6", 2)
    err7.save("err7")

    err8 = Log10("FoS_Mean")
    err8.save("err8")

    err9 = Minus("err8", "err7")
    err9.save("mu")

    err10 = Log10("err10")
    err10.save("err10")

    err11 = Minus("err10", "mu")
    err11.save("err11")

    err12 = Times(1.414214, "err5")
    err12.save("err12")

    err13 = Divide("err11", "err12")
    err13.save("x")

    err14 = Abs("x")
    err14.save("err14")

    err15 = Times(0.5, "err14")
    err15.save("err15")

    err16 = Plus(1, "err15")
```

```
err16.save("err16")

err17 = Divide(1, "err16")
err17.save("t")

err18 = Times("err17", 0.17087277)
err18.save("err18")

err19 = Minus("err18", 0.8215223)
err19.save("err19")

err20 = Times("err19", "err17")
err20.save("err20")

err21 = Plus("err20", 1.4885187)
err21.save("err21")

err22 = Times("err21", "err17")
err22.save("err22")

err23 = Minus("err22", 1.13520398)
err23.save("err23")

err24 = Times("err23", "err17")
err24.save("err24")

err25 = Plus("err24", 0.27886807)
err25.save("err25")

err26 = Times("err25", "err17")
err26.save("err26")

err27 = Minus("err26", 0.18628806)
err27.save("err27")

err28 = Times("err27", "err17")
err28.save("err28")

err29 = Plus("err28", 0.09678418)
err29.save("err29")

err30 = Times("err29", "err17")
err30.save("err30")

err31 = Plus("err30", 0.37409196)
err31.save("err31")

err32 = Times("err31", "err17")
err32.save("err32")

err33 = Plus("err32", 1.0002368)
err33.save("err33")

err34 = Times("err33", "err17")
err34.save("err34")

err35 = Minus("err34", 1.265512231)
err35.save("err35")

err36 = Times("err14", "err14")
err36.save("err36")
```

```
    err37 = Negate("err36")
    err37.save("err37")

    err38 = Minus("err37", "err35")
    err38.save("z")

    err39 = Exp("z")
    err39.save("err39")

    err40 = Times("err17", "err39")
    err40.save("erfcc")

    erfccMinus = Minus(2, "erfcc")
    erfccMinus.save("erfccMinus")

    erfccFix = Con("x", "erfccMinus", "erfcc", "Value < 0")
    erfccFix.save("erfccFix")

    erf = Minus(1, "erfccFix")
    erf.save("erf")

    err41 = Plus(1, "erf")
    err41.save("err41")

    logprob = Times(0.5, "err41")
    logprob.save(Outpath + "\\" + "Probability")
    arcpy.AddMessage("Probability Created")

# Reliability
if str(Reliability) == "true":

    rel1 = Minus("FoS_Mean", 1)
    rel1.save("rel1")

    rel2 = Divide("rel1", "FoS_SD")
    rel2.save(Outpath + "\\" + "Reliability")
    arcpy.AddMessage("Reliability Created")
arcpy.AddMessage("Program Complete")
arcpy.AddMessage(time.time() - start)
```