Title

CryoSheds: a GIS Modeling Framework for Generating Hydrologic Watersheds for Cryo-Hydrologic Systems using Digital Elevation Models and Remote Sensing Observations

Author

Pitcher, Lincoln H

Publication Date

2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

CryoSheds: a GIS Modeling Framework for Generating Hydrologic Watersheds for Cryo-Hydrologic Systems using Digital Elevation Models and Remote Sensing Observations

A thesis submitted in partial satisfaction of the requirements

for the degree Master of Arts in Geography

by

Lincoln H Pitcher

2015

ABSTRACT OF THE DISSERTATION

CryoSheds: a GIS Modeling Framework for Generating
Hydrologic Watersheds for cryo-hydrologic systems using Digital Elevation Models
and Remote Sensing Observations

By

Lincoln H Pitcher

Master of Arts in Geography

University of California, Los Angeles, 2015

Professor Laurence C. Smith, Chair

A semi-automated modeling framework for generating hydrographic watersheds for cryo-hydrologic systems using Geographic Information Systems (GIS) tools is presented. The framework derives two alternate types of watersheds i) hydraulic pressure potential (Shreve 1972; Cuffey & Paterson 2010; Banwell et al. 2013), which determines surface/subsurface flow paths from the hydrostatic equation, using surface and basal topography DEMs; and ii) surface (i.e. surface flow paths) as inferred from a surface topography DEM alone. The framework utilizes standard hydrologic modeling tools available in the ArcGIS 10.2 and the ArcPy library. Specifically, DEM depression filling, flow direction, flow accumulation, basin and watershed

tools are used in conjunction with custom ArcPy routines to aggregate sub basins, identify hydrologic flow divides and delineate ice sheet hydraulic pressure potential and surface ice watersheds. Both watershed types are delineated for seven nested watersheds in southwest Greenland, derived from remotely sensed pour points along the Aussivigssuit River and its tributaries. The two alternate methods produce watersheds with dissimilar outcomes, particularly at higher elevations (670 m and above) on the ice sheet. For the Aussivigssuit River hydrologic network, surface DEM watersheds tend to be both larger in size and extend to higher elevations when compared to the hydraulic potential watersheds.

The thesis of Lincoln H Pitcher is approved:

Gregory S. Okin

Yongwei Sheng

Laurence C. Smith, Committee Chair

University of California, Los Angeles

2015

Table of Contents:

1. Introduction:..... 1
2. Data Requirements and Study Site: 3
3. Watershed Modeling Framework: 5
3.1 Outlet Snapping to DEM grid: 5
3.2 Land Watershed Delineation: 6
3.3 Ice sub-basin Delineation: 7
3.4 Defining ice sub basin modeled meltwater pour points: 8
3.5 Watershed delineation: 9
4. Results and Conclusions: 9
4.1 Areal differences between hydraulic potential and surface watersheds: 10
4.2 Elevation Extent differences between hydraulic potential and surface watersheds: 10
4.3 Discussions and Conclusions: 11
Tables: 13
Figures: 16
Appendix: 23
Delineate Ice Sub Basins: Surface DEM 23
Delineate Ice Sub Basins: Hydraulic Potential 25
Define Modeled Pour Points for Ice Sub Basins 28
Generate Land Watersheds 31
Merge Land Watersheds into one file and populate Pour Point Identifier Attribute Field 33
One Pixel Ice Edge Buffer 35
Define Modeled Pour Points for Ice Sub Basins 37
For Each Pour Point Merge Edge Ice Sub Basins 39
Iterative Check for Contributing Internal Ice Sub Basin and CryoSheds Growing Routine 43
References: 53

Acknowledgements:

This thesis will be submitted as a journal article for publication in a peer reviewed journal co-authored by Laurence C. Smith. I thank my committee members Laurence C. Smith, Gregory Okin, and Yongwei Sheng for their assistance and support in the development of this thesis.

Funding for this research was provided by the NASA Cryospheric Science Program Grants: NNX11AQ38G and NNX14AH93G, managed by Dr. Thomas P Wagner; the NASA Earth and Space Science Fellowship Grant: NNX14AP57H, and the UCLA Graduate Division Research Mentorship Fellowship.

I also thank:

Vena W. Chu

Colin J. Gleason

Kang Yang

Jida Wang

Asa K. Rennermalm

Kelly J. Easterday

1. Introduction:

Geographic Information Systems (GIS) based tools provide a practical platform for extracting hydrologic parameters from topographic and/or remotely sensed information. The advent of gridded digital elevation models (DEMs), in particular, enhances the utility of GIS based extraction of hydrologic parameters at local, regional and global scales (Morse 1968; Peucker & Douglas 1975; Mark 1984; Band 1986; Tarboton et al. 1991; Wilson et al. 2007). In terrestrial surface water systems, basic hydraulics dictate that water flows through preferential flow routes from areas of high to low elevation. GIS tools and DEMs are particularly useful for simulating these topographically defined flow-routes and separating contributing hydrologic areas according to bounding watershed ridgelines i.e. divides (Collins 1975; Marks et al. 1983; Moore et al. 1983; O'Callaghan & Mark 1984; Band 1986; Band et al. 2000).

More recently, terrestrial GIS and DEM based watershed approaches have been applied to cryo-hydrologic systems to produce hydrologic flow paths and watershed divides for glaciated systems (Rippin et al. 2003; Lewis & Smith 2009; Mernild & Liston 2012; Rennermalm et al. 2013; Bamber et al. 2013; Banwell et al. 2013; Smith et al. 2015). In such systems, principles of physics and hydraulics dictate that meltwater drains along a pressure gradient from areas of high to low pressure, the pattern of which is influenced by both bedrock and surface topography (Shreve 1972; Paterson 1993; Cuffey & Paterson 2010). However, some cryo-hydrologic studies generate watersheds using gridded surface topography alone, a direct analog for traditional terrestrial watershed delineation (Mernild & Liston 2012; Hasholt et al. 2013).

Both the hydraulic potential and the surface DEM method seek to represent different physical processes for cryo-hydrologic systems. Hydraulic potential watersheds control for surface, en-, and subglacial runoff and routing while surface DEM generated watersheds account

for surface runoff and routing. While the hydraulic potential method is generally favored from a theoretical standpoint, it is currently unknown which of these two methods is more accurate in practice for geographically delimiting the extent and shape of cryo-hydrologic watershed boundaries. Watershed delineation in cryo-hydrologic systems is further complicated by DEM resolution and varied topography along the perimeter of the ice sheets and glaciers which results in standard GIS toolboxes producing a number of small (with areas less than 10 pixels) sub basins (primarily along the ice sheet/glacier perimeters) that do not appear to be hydrologically accurate or relevant. In recognition of these uncertainties, Smith et al. 2015 propose adopting an end-member approach using both hydraulic potential and surface DEM watershed delineation methods with a range of DEM resolutions, to produce a spectrum of possible watersheds for a single pour point.

To encourage generation this multi-watershed approach, a standardized, reproducible procedure for estimating a variety of potential watersheds in cryo-hydrologic systems is needed. To that end, this study incorporates hydraulic potential and surface DEM methods into a single GIS modeling framework, here termed CryoSheds. This procedure combines assumed or observed meltwater pour points along the perimeter of an ice-sheet or glacier with gridded elevation data into a GIS-based model to produce watersheds constrained to a defined pour-point for both hydraulic pressure potential and surface DEM approaches. A feasibility demonstration is applied to seven remotely sensed pour points along the perimeter of the Greenland Ice Sheet (GrIS) in western Greenland, yielding seven candidate cryo-hydrologic watersheds that route water from the GrIS interior over and/or through the ice sheet to user-specified pour points in proglacial (terrestrial) lakes/rivers/streams. The approach thus differs from previous methods by

effectively coupling the ice sheet and proglacial zones, and allowing flexible, user-defined pour points to be located anywhere between the ice edge and coastline.

2. Data Requirements and Study Site:

The CyroSheds watershed modeling framework developed here requires inputs from three different sources: (1) Surface and basal topography DEMs; (2) a land-ice-ocean mask; and (3) user-selected pour points. For this first demonstration, these datasets consisted of: IceBridge BedMachine Greenland surface DEM (Morlighem et al. 2015), which is derived from the Greenland Mapping Project (GIMP) Greenland surface DEM (surface DEM) (Howat et al. 2014); BedMachine Greenland mass conservation bedrock topography (basal DEM) (Morlighem et al. 2011; Morlighem et al. 2014; Morlighem et al. 2015); the IceBridge BedMachine land cover mask, which is derived from the GIMP land cover product (land ice mask) (Morlighem et al. 2015; Howat et al. 2014); and seven remotely sensed meltwater pour points observed in 30m resolution Landsat satellite imagery.

The gridded topographic datasets are all parameters of the IceBridge BedMachine Greenland data product, hosted by the National Snow and Ice Data Center (Morlighem et al. 2014; Morlighem et al. 2015), available in netcdf format at 150m x150m posting in the Polar Stereographic North ESPG 3413 projection. The GrIS surface DEM (Figure 1A) is a downscaled version of the products released and hosted by Greenland Mapping Project (GIMP) (Howat et al. 2014). The BedMachine GrIS Bedrock topography DEM (Figure 1B) uses a mass conservation approach that derives ice thickness from both ice velocity estimated using Synthetic Aperture Radar interferometry and Airborne Radar estimates of ice thickness for outlet glaciers (Morlighem et al. 2011; Morlighem et al. 2014), and spatial interpolation techniques (Figure 1C).

The land-ice-ocean mask (Figure 1D) is bundled with the BedMachine data (Morlighem et al. 2014) but is a downscaled version of the GIMP ice cover mask which classifies all of Greenland as grounded ice, floating ice, or ice free land surface (Howat et al. 2014).

Seven pour points were manually identified using a Landsat-8 OLI image (Scene ID LC80070132014218LGN00) collected on August 6, 2014 (Figure 2). The pour points were precisely mapped to encompass the Aussivigssuit River (AR) watershed and the nested watersheds corresponding with its northern tributary Pinguarssup Alanguata Kugssua (PK) and its southern tributary Manitsut Alanguisa Kugssuat (MK). Note that pour points PKN and PKS comprise the northern and southern branches of the PK tributary, while pour points MKN and MKS comprise the northern and southern branches of the KM tributary. Pour point PK maps the northern branch of the AR while pour point MK maps the southern branch of the AR. Pour point AR is located along the main channel of the AR river downstream of any additional GrIS sourced meltwater inputs and is directly connected to the ocean. Viewed collectively, these seven pour points provide an integrated view of seven nested land ice watersheds in southwest Greenland.

The majority of the GrIS directly connects to the global ocean via marine terminating glaciers and large calving fronts. However, the southwest coast of Greenland comprises an ice-free swath of land separating the GrIS from the ocean. In this area, land-terminating outlet glaciers terminate in proglacial rivers, streams and lakes that route GrIS meltwater to the global ocean. Within this area, the AR was identified as an ideal demonstration site for CryoSheds because it has multiple tributaries, and it exhibits varied surface and bedrock topography (especially along the GrIS perimeter).

3. Watershed Modeling Framework:

The utility of CryoSheds is to generate varying GrIS watersheds upstream of user-defined meltwater pour points, by integrating internally drained and/or edge basins based on modeled outflow locations and directions (Figure 3). As such, it builds upon previous studies (e.g.: Lewis and Smith 2009; Mernild and Liston 2012; Smith et al. 2015 and others) by routing watershed extents from the GrIS interior through proglacial hydrologic outlets to the ocean.

The CryoSheds modeling framework operates in ArcGIS 10.2 and the ArcPY geoprocessing environment. Many of the GIS routines are standard ArcGIS spatial analyst hydrology tools (most require the spatial analyst extension), while additional routines and loops were scripted in python using the ArcPy library (python scripts are included in appendix). Its major geoprocessing steps are outlined in Figure 5 and are discussed in detail in the sections to follow.

3.1 Outlet Snapping to DEM grid:

CryoSheds defines a pour point as any user-specified proglacial location that is snapped to a DEM-derived hydrologic stream network intersecting the ice edge. In DEM based feature extraction, hydrologic features are reduced to a set of pixels and a vector line connecting the centroid of each pixel is computed. To define contributing watersheds, each outlet location must snap to the DEM hydrologic grid. To achieve this, streams are extracted from the input surface DEM following Tarboton et al. (1991) and others and pour points are manually snapped to the DEM extracted stream network (Figure 2).

To extract streams, first a depression-free DEM is created by filling all sinks. This step ensures that every cell in the DEM is a part of at least one path of cells that routes to the edge of the DEM (Jenson & Domingue 1988). Next, the 8-neighbor flow direction (i.e. the direction that

water is routed from each cell to its downstream neighbor) is calculated for each cell in the DEM (Jenson & Domingue 1988). The flow direction grid is then used as the input to calculate a flow accumulation grid which, for every cell, calculates the cumulative number of upstream cells that flow to that cell. For example, a cell with a flow accumulation value of zero corresponds with a ridgeline, while cells with high flow accumulation values correspond with hydrologic features and/or flow paths (Jenson & Domingue 1988).

For the purposes of this CryoSheds demonstration, streams were defined as cells with flow accumulation values of 100 or greater. This is a relatively low threshold but the DEM extracted streams were solely used to validate the AR river network and to snap pour points to the BedMachine grid. It is important to note that for many GIS applications an automated snap outlet to grid procedure is sufficient. However, an automated approach is not recommended for use in Greenland because anomalies in land surface DEM extracted streams may route meltwater into ancillary watersheds where no ice-connected hydrologic features are observed.

3.2 Land Watershed Delineation:

For each assumed or observed pour point, the upstream watershed is delineated for the contributing non-ice land surface area using the input depression free surface DEM, flow direction grid and snapped meltwater pour points (Figure 2). Next, a one pixel mask is generated for the interior of the GrIS boundary file, the mask is intersected with each land watershed and a pour point numeric identifier is appended to the mask. This one-pixel mask with identifier is used to join ice sub basins into watersheds (Figure 3).

3.3 Ice sub-basin Delineation:

Ice sub basins define regions of an ice sheet or glacier that drain to a common pour point. Ice sub basins are generated entirely from an input DEM (ice surface DEM or hydraulic pressure grid) and thus no a priori pour point is defined. It is important to emphasize that these are potential ice sub basins and do not necessarily reflect active surface or subsurface hydrologic flow (which require above-freezing meteorological conditions to also be present). Following Smith et al. 2015 ice sub basins are generated using both hydraulic potential and surface DEM approaches, the first uses a hydraulic pressure grid and the second uses a surface DEM (Figure 4 and Figure 6).

To generate ice sub-basins from the surface DEM:

1. Clip the surface DEM to the ice mask
2. Iteratively fill the DEM by removing sinks
3. Calculate the eight-pixel flow direction for each grid cell
4. Use the flow direction grid to generate ice sub-basins.

To generate ice sub basins from a hydraulic potential grid:

1. Clip surface DEM and basal DEM to the ice mask
2. Iterative fill the surface DEM and the basal DEM by removing sinks
3. Calculate the hydraulic potential grid as:

$$\varphi \cong \rho_i g(h_s + 0.1y)$$

where φ is the calculated height of the pressure grid, ρ_i is the density of ice (assumed to be a constant 917 kg m^{-3}), g is acceleration due to gravity (9.81 m s^{-2}), h_s is elevation of the ice sheet surface (m), and y is the elevation of the underlying basal topography (m).

The hydraulic potential method assumes: (1) that water flows along the steepest subglacial hydraulic potential gradient (Banwell et al. 2013; Shreve 1972), (2) that meltwater generated at the surface of the GrIS reaches the bed and drains along an impermeable ice-bed interface (Bjornsson 1986; Banwell et al. 2013), and (3) that water pressure equals ice overburden pressure (Shreve 1972; Cuffey & Paterson 2010; Bamber et al. 2013)

4. Calculate the eight-pixel flow direction for each grid cell
5. Use the flow direction grid to calculate ice sub-basins. It is important to note that flow direction and drainage divides are dominated by surface topography except for in cases where the bedrock slope is significantly steeper compared to the surface slope (Lewis & Smith 2009; Mernild & Liston 2012)

3.4 Defining ice sub basin modeled meltwater pour points:

CryoSheds defines ice sub basin pour points as the maximum flow accumulation grid cell contained within each ice sub basin. To extract ice sub basin outlets first the maximum flow accumulation for any cell contained within each ice sub basin is universally defined as the value for that entire sub basin. Next, the raster calculator tool is used to generate a difference raster, which is the difference of the flow accumulation raster and the constant value maximum flow accumulation ice sub basin mask. The result is a mask of each ice sub basin in which any grid cell with a value of zero is the meltwater pour point for the given ice sub basin in which it is contained. These modeled outlets are used, in combination with the flow direction at the outlet, to merge ice sub-basins into watersheds.

3.5 Watershed delineation:

To generate watersheds, the one-pixel land watershed ice edge buffer is intersected with ice sub basin modeled meltwater outlets. Then, the pour point identifier is joined to the modeled outlet and the corresponding ice sub basin for each outlet is selected. These sub basins that flow from the GrIS and into one of the delineated land watersheds are merged together for each land watershed and meltwater outlet, here termed “ice-land watershed”. Next, unmerged, internally drained ice sub basins are fused with ice watershed. Finally, a custom ice watershed iterative growing algorithm is applied in order to handle non-contained internally drained ice sub basins. The iterative growing routine starts by searching for non-contained internally drained ice sub basin pour points that are within a pixel buffer of the ice watershed edge. If pour point(s) are found, the flow direction at that pixel is determined and the pour point is projected in the direction of its DEM based flow direction until its intersection with the ice watershed of interest, another ice watershed, an ice free land surface or the ocean. If it is determined that the non-contained ice sub basin drains into the ice watershed then the sub basin is merged with the ice watershed. This ice watershed growing routine process is repeated until all non-contained ice sub basins that share a boundary with the ice watershed are checked. All ArcPy scripts of this process are included in the appendix.

4. Results and Conclusions:

CryoSheds represents a self-contained, automated modeling framework that enables flexible production of varying cryo-hydrologic watersheds that route water over ice and land towards user-defined pour points. It yields two theoretically different types of cryo-hydrologic watershed delineations, one using hydraulic pressure potential to determine hydrostatic surface/subsurface

flow paths, and the other using ice surface topography alone to infer supraglacial flow paths and watershed divides. CryoSheds employs standard ArcGis tools supplemented by additional ArcPy scripts (see appendix). A first demonstration of CryoSheds for seven meltwater pour points in southwest Greenland is shown (Figure 7). Note that the two delineation approaches produce watersheds with significantly different areal sizes and elevation extents, as will be described next.

4.1 Areal differences between hydraulic potential and surface watersheds:

As compared to hydraulic potential watersheds, surface watersheds had areal differences ranging from 4.5% to 169.2%, with a general tendency for surface watersheds to be larger than hydraulic potential watersheds (Table 1). For example, for the MKS pour point the surface DEM watershed is 395.9 km^2 and the hydraulic potential watershed is 378.5 km^2, a 4.5% difference. In contrast, for pour point AR the surface DEM ice watershed is 22628.8km^2 and the hydraulic potential ice watershed is 3340.9 km^2, a 148.5% difference (complete size differences summarized in Table 1). Areal differences in ice watersheds generated with the hydraulic potential and surface DEM techniques given the same pour point underscores the potential for using both methods to produce end-member estimates of watersheds for cryo-hydrologic systems.

4.2 Elevation Extent differences between hydraulic potential and surface watersheds:

In addition to areal differences, surface watersheds typically have different shapes than corresponding hydraulic potential watersheds (Figure 7), with surface watershed tending to extend to higher elevations (Table 3). For example, the maximum ice surface elevation in the surface DEM ice watershed for pour point AR is 2560 m (WGS 1984 datum) whereas the

maximum elevation in the hydraulic potential watershed for pour point AR is 1884 m. Similarly, for pour point PK, the maximum elevation in the surface DEM ice watershed is 1357 while the maximum elevation in the hydraulic potential ice watershed is 999 (complete elevation differences are summarized in Table 3).

The maximum agreement between watershed delineations is found for pour point MKS, where 90.5% of the surface DEM ice watershed intersects the hydraulic potential ice watershed and 94.6% of the hydraulic potential ice watershed intersects the surface DEM watershed. The most dissimilar watersheds are those corresponding with outlet PKM. For the PKM ice watershed 8.1% of the surface DEM ice watershed intersects the hydraulic potential watershed and 97.8% of the hydraulic potential ice watershed intersects the surface DEM watershed (complete percent intersect results are summarized in Table 2). Differences in ice watershed elevation extent and ice watershed intersection percent are likely to due to variations in basal topography that influence hydrologic flow divides.

4.3 Discussions and Conclusions:

The CryoSheds modeling framework offers new utility to the Greenland hydrology science community that currently lacks a standardized procedure for effectively integrating ice sub basins into watersheds draining to defined pour point at or downstream of the ice edge. Furthermore, it offers seamless generation of two end-member watershed delineation practices currently in use by the community, each yielding different outcomes as underscored by the contrasting areal and shape differences presented here. There is no obvious explanation for these observed differences; but they are unsurprising as the two approaches simulate different physical processes. Two other possible explanations are (1) differences in root data sources between

surface and basal DEMs, which may cause grid disagreements when the two datasets are merged into a hydraulic pressure grid and results are compared with just the surface DEM; and (2) the bedrock topography in the AK watershed may be sufficiently steep compared to the surface topography such that the hydraulic potential watershed divides are dominated by bedrock rather than surface topography.

In conclusion, the CryoSheds modeling framework offers a flexible, standardized toolkit for generating a variety of ice watersheds for cryo-hydrologic systems. The non-trivial differences between the surface DEM and hydraulic potential approaches observed in this demonstration study suggest that future research should incorporate additional empirical datasets, particular hydrologic outflow (Rennermalm et al. 2012a; Rennermalm et al. 2012b; Rennermalm et al. 2014; Smith et al. 2015; Hasholt et al. 2013; Mernild & Liston 2012; Mernild & Hasholt 2009) to determine if either approach excels for the purpose of delineating cryo-hydrologic watersheds for the Greenland ice sheet. Until then, we suggest considering both watershed types simultaneously as end member possibilities (as presented here using CryoSheds modeling), rather than any single delineation alone.

Tables:

Watershed Size Differences				
Outlet Identifier	Stream Order	Surface Area km2	Potentiometric Area km2	% area difference
MKS	Low	395.9	378.5	4.5
MKN	Low	21702.5	2733.9	155.2
PKS	Low	148.5	196.7	27.9
PKN	Low	381.9	31.8	169.2
PK	Medium	530.4	228.5	79.6
MK	Medium	22098.4	3112.4	150.6
AR	High	22628.8	3340.9	148.5
Summary Statistics				
Minimum		148.5	31.8	4.5
Average		9698.1	1431.8	105.1
Median		530.4	378.5	148.5
Maximum		22628.8	3340.9	169.2

Table 1: Summary of areal differences for surface DEM and hydraulic potential ice watersheds. Percent area difference for each outlet are calculated as the absolute value of the area difference between surface and potentiometric watersheds divided by the average area of the surface and hydraulic potential ice watersheds.

pour point identifier	intersecting area km^2	surface DEM % intersect	Hydraulic potential % intersect
MKS	358.22	90.48	94.65
MKN	2715.59	12.51	99.33
PKS	140.40	94.57	71.40
PKN	31.12	8.15	97.81
PK	172.13	32.45	75.34
MK	3092.24	13.99	99.35
AR	3320.69	14.67	99.40

Table 2: Summary of areal intersection for ice watersheds categorized by pour point. The intersecting area km^2 category is the areal extent of intersection between the two ice watersheds for a given pour point. The surface % intersect category defines the percent of a surface DEM ice watershed that intersects a hydraulic potential ice watershed for a given pour point. The hydraulic potential % intersect category defines the percent of a hydraulic potential ice watershed that intersects a surface DEM ice watershed for a given pour point.

Pour Point identifier	Surface DEM elevation (m)		Hydraulic Potential elevation (m)	
	Min	Max	Min	Max
MKS	486	1362	486	1331
MKN	233	2560	233	1884
PKS	215	877	215	999
PKN	199	1357	199	670
PK	199	1357	199	999
MK	233	2560	233	1884
AR	199	2560	199	1884

Table 3: Summary table of minimum and maximum elevations (in meters above the World Geographic System 1984 datum) contained within both surface DEM and hydraulic potential ice watersheds.

Figures:

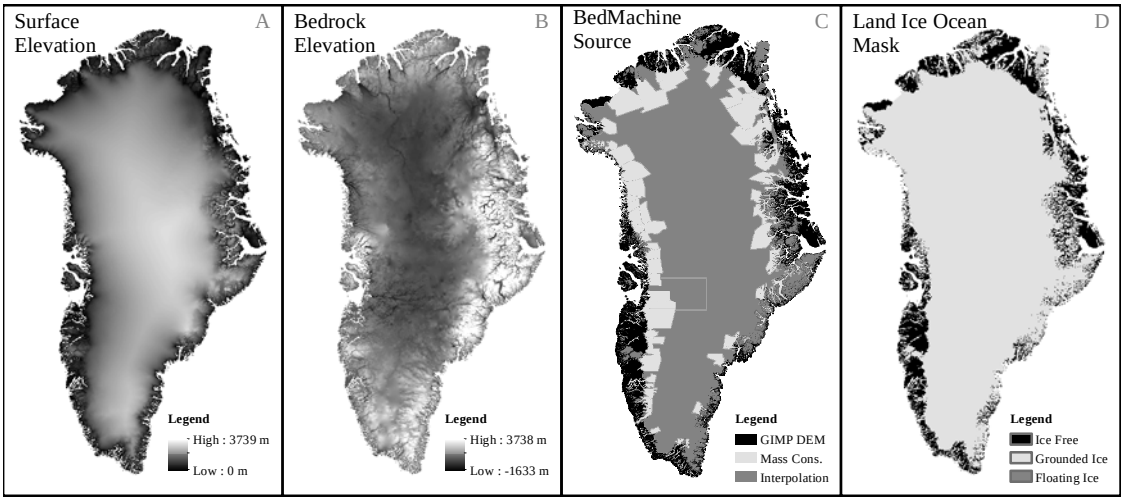

Figure 1: IceBridge BedMachine Greenland Layers. (A) The surface elevation DEM for both the ice free land surface area and GrIS ice extent is derived from the Greenland Mapping Project DEM (Howat et al. 2014). The surface elevation DEM product has a resolution of 30m but in order to be included with the IceBridge BedMachine Greenland layers, it was resampled to 150m resolution. (B) The IceBridge BedMachine Greenland bedrock elevation data (Morlighem et al. 2015; Morlighem et al. 2014). (C) The bedrock elevation data was derived using a mass conservation approach for outlet glaciers (Morlighem et al. 2011; Morlighem et al. 2014). (D) The land ice ocean mask is derived from the Greenland Mapping Project data product (Howat et al. 2014).

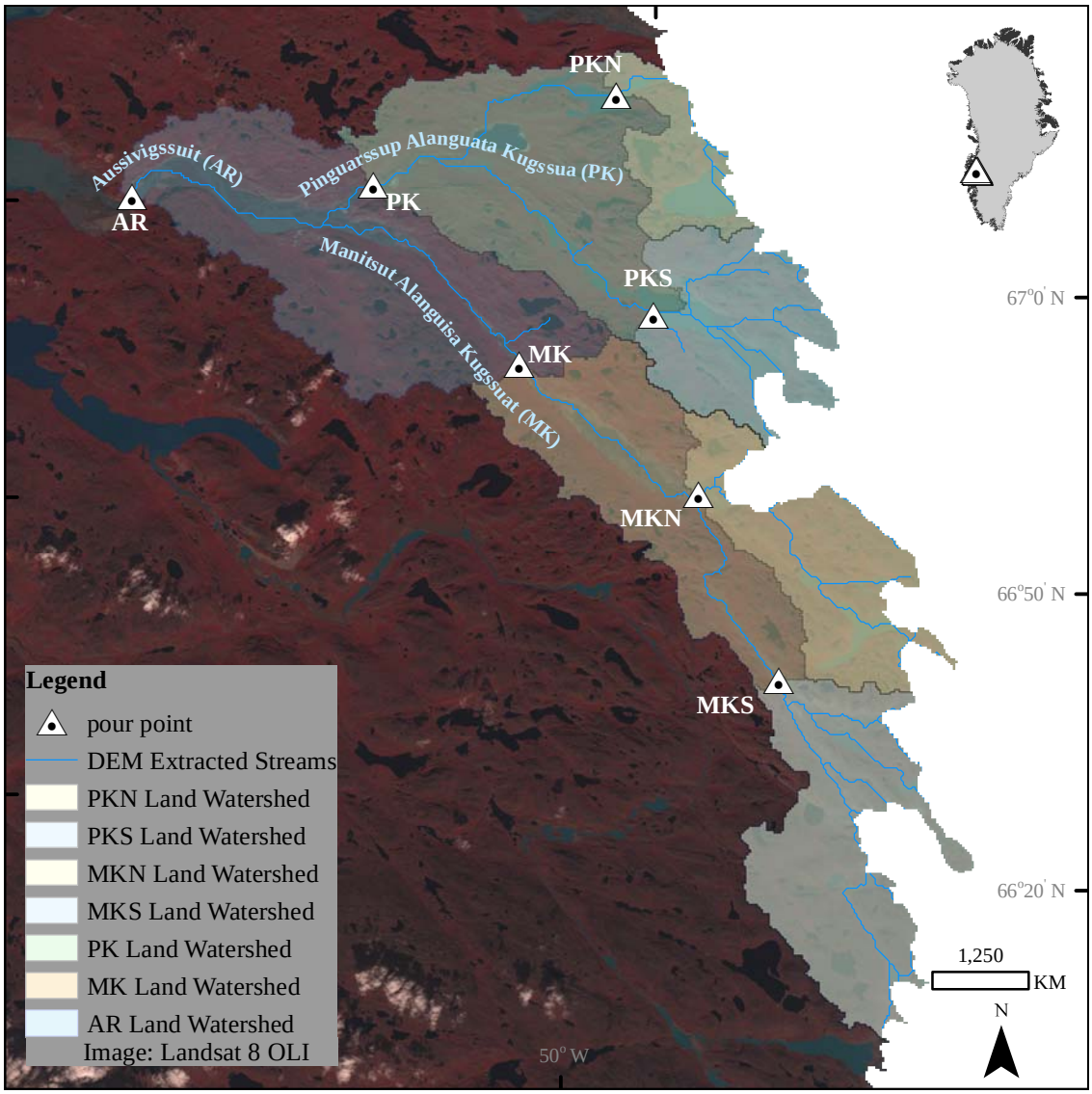

Figure 2: Seven meltwater pour points (triangles) all within the Aussivigssuit (AR) river watershed, southwest Greenland, were manually identified using a Landsat OLI image collected on August 6, 2014. The AR river is fed from the north by the Pinguarssup Alanguata Kugssua (PK) river and from the south by the Manitsut Alanguisa Kugssuat river (MK). Pour points - any point location along a terrestrial hydrologic feature that is sourced from cryo-hydrologic meltwater runoff – are drawn along the main stem of the AR river downstream of any additional inputs, along both the MK and PK tributaries and along the northern (PKN and MKN) and southern (PKS and MKS) branches of the MK and PK tributaries. The DEM extracted rivers (blue lines) and the nested land watersheds for each of the seven pour points are mapped to their intersections with the GrIS perimeter.

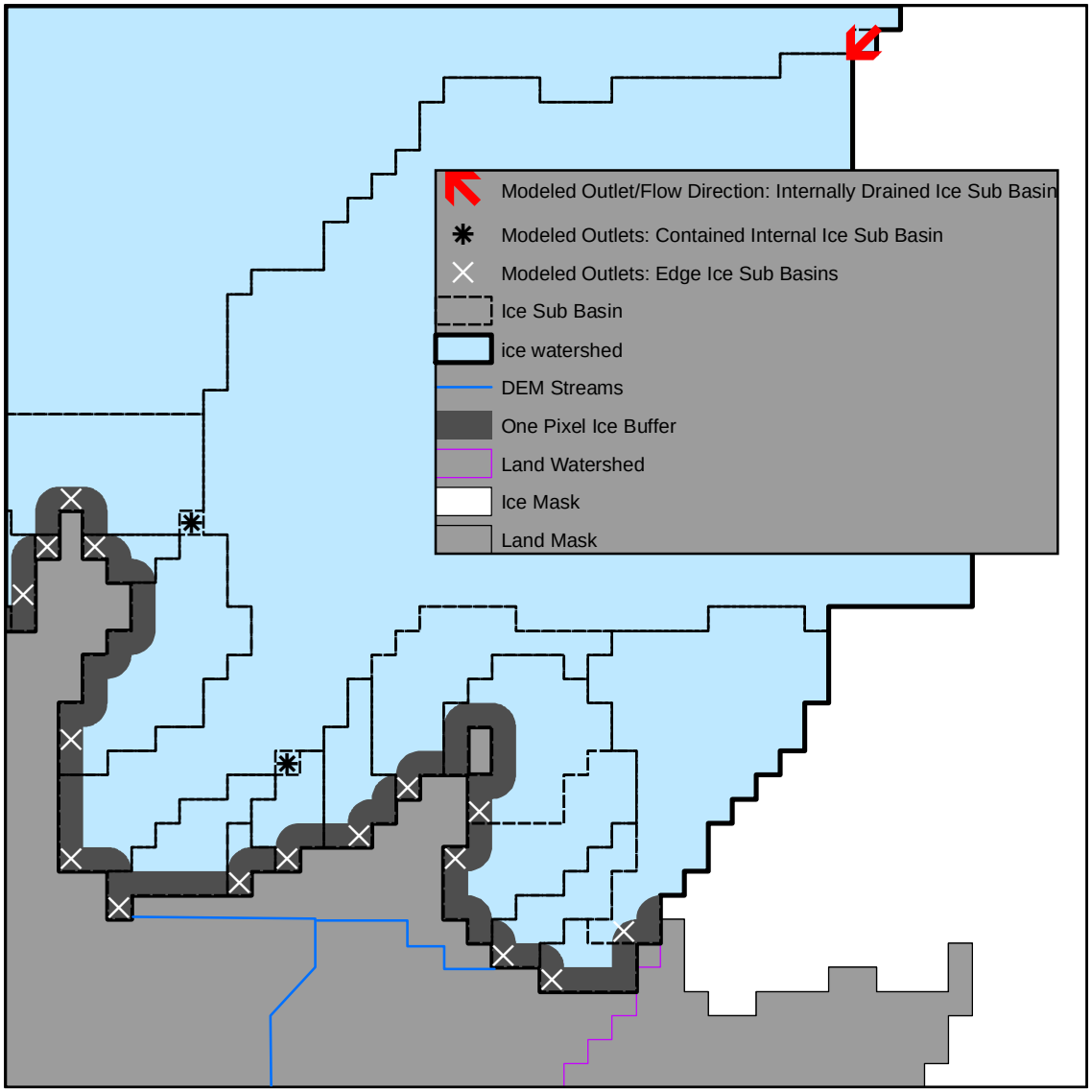

Figure 3: Illustrative figure displaying required inputs and intermediate products produced within CryoSheds modeling framework. The light grey shaded region maps non ice covered land surface area. The blue line represents the DEM extracted stream network to its intersection with the ice edge. The purple line represents the land watershed boundary to its intersection with the ice edge. The white region maps part of the ice sheet not included in the example ice watershed. The white 'x' symbols map the location of modeled pour points for ice sub basins that touch the edge of the ice sheet. CryoSheds merges ice sub basins with modeled pour points that fall within one pixel of the ice perimeter. The contained internal ice sub basins and corresponding outlets are marked by the * symbol. CryoSheds merges contained ice sub basins into the ice watershed. The red arrow plots the modeled pour point locations and flow directions of non-contained internally drained ice sub basins. CryoSheds merges such non-contained ice sub basins based on the flow direction at the modeled pour point.

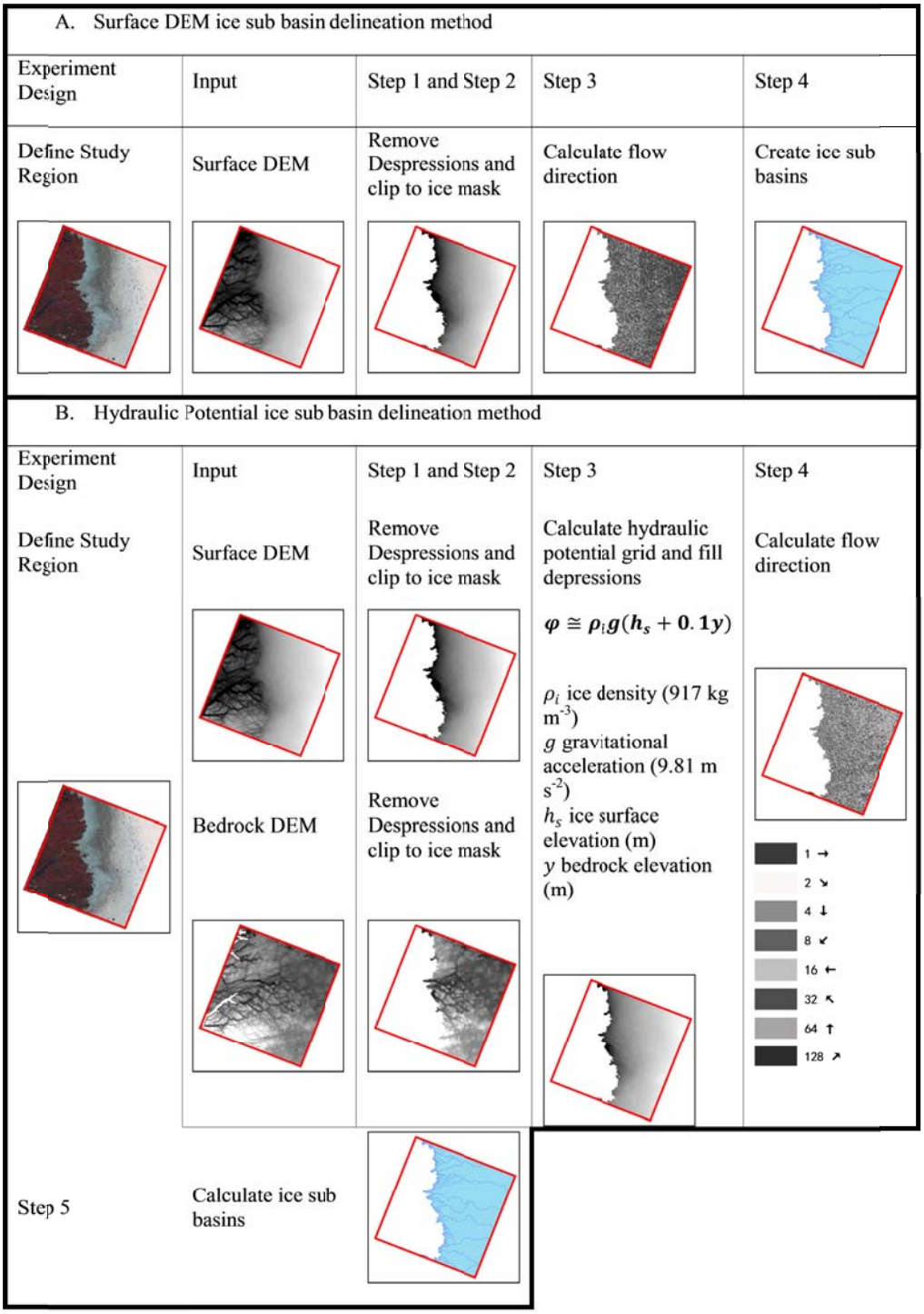

Figure 4: Conceptual diagram of ice sub basin generation steps used in the CryoSheds modeling framework. Subsection A (top) outlines the considerations, inputs and steps required to generate ice sub basins using the Surface DEM method. Subsection B (bottom) outlines the considerations, inputs and steps required to generate ice sub basins using the hydraulic potential method. Step numbers refer to those outlined in section 3.4.

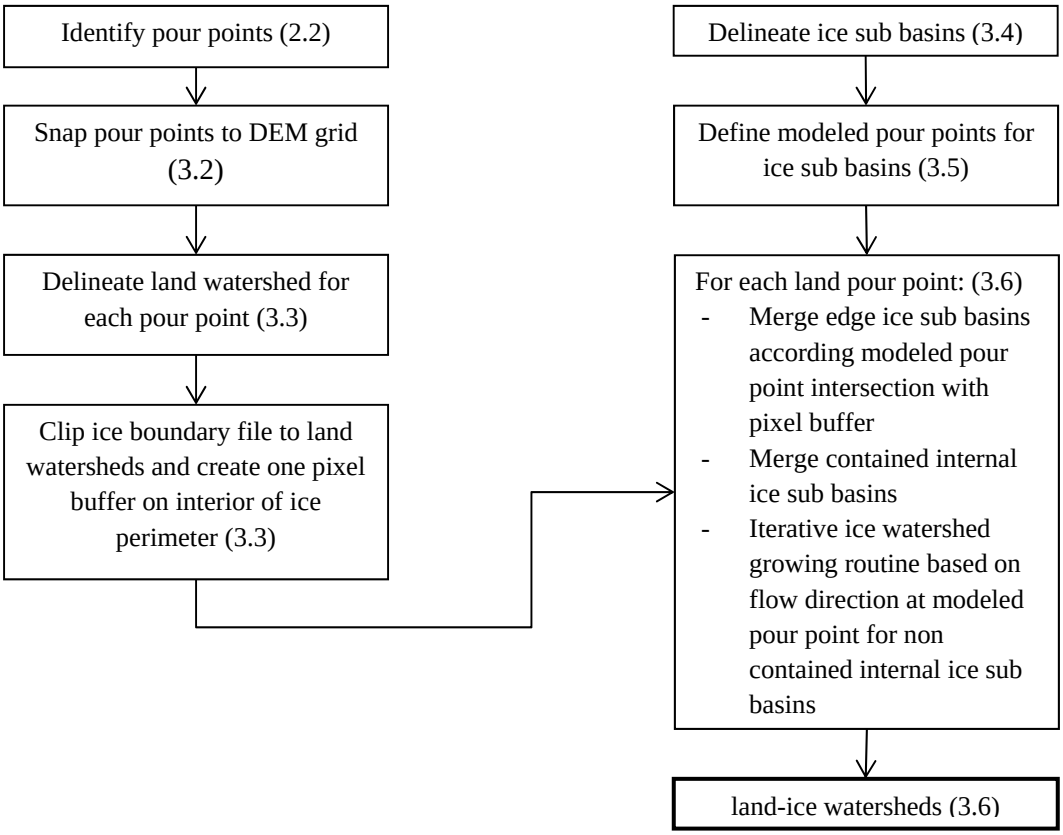

Figure 5 Conceptual diagram of the CryoSheds hydrographic watershed modeling framework. Each box corresponds to step in CryoSheds and parentheses refer to specific subsection in which the step is explained.

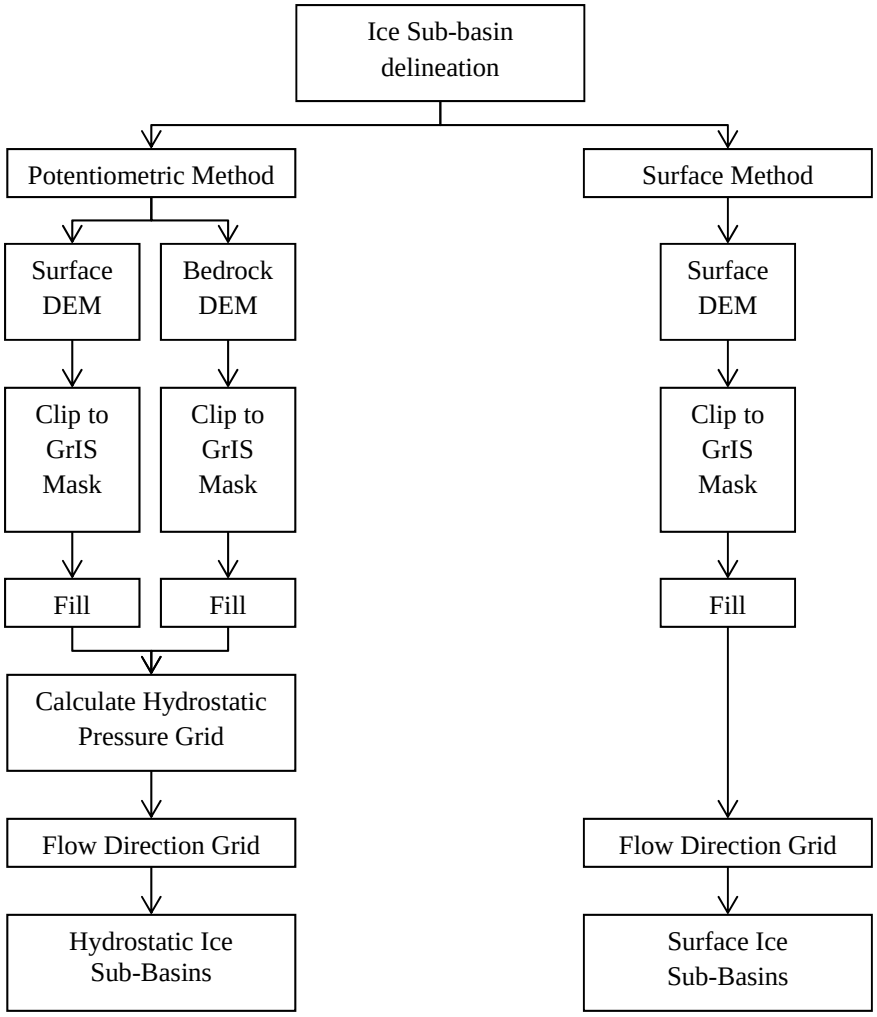

Figure 6: Ice sub-basin generation steps for both the surface DEM and hydraulic potential approaches. The hydraulic potential approach combines surface and bedrock topography into a pressure grid. This method assumes that that water flows along a pressure gradient from areas of high to low pressures (Cuffey & Paterson 2010; Shreve 1972). The surface method assumes that cryo-hydrologic drainage is driven by surface topography alone. In both methods all DEM depressions are assumed to be errors and are iteratively filled such that water is routed to the ice edge.

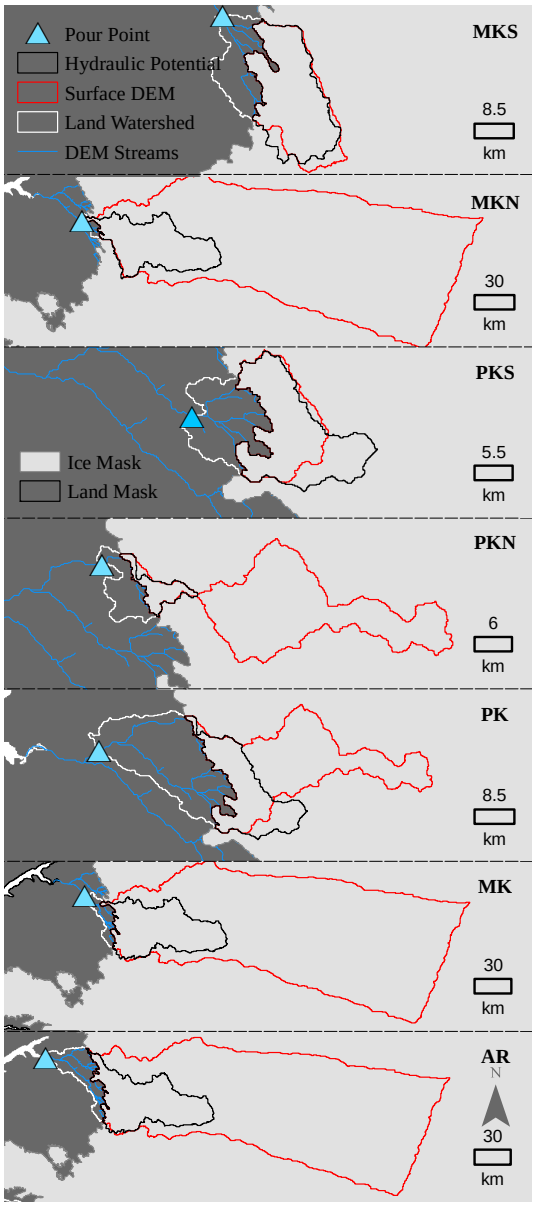

Figure 7: Map of ice watersheds for the seven pour points in the AR watershed. Triangles mark pour point locations. White lines note land watershed divides and blue lines note DEM extracted stream network. The hydraulic potential ice watersheds are mapped in black and the surface DEM ice watersheds are mapped in red. The light grey region demarcates the GrIS perimeter and extent while the dark grey region demarcates the non-ice land surface ocean shoreline and extent.

Appendix:

Delineate Ice Sub Basins: Surface DEM

```
#-------------------------------------------------------------------------------
# Name:        GenerateIceSubBasins_Surface.py
# Purpose:     Creates drainage basins for Greenland Ice Sheet based solely
on
#              surface topography DEM
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Declare input variables
 #
 #Ice Surface DEM Raster File
surfaceDEM = Raster('###INPUT NAME###')
                    #^^^insert file name here
surfaceDEMString = '###INPUT NAME###'
                   #^^^insert file name here
 #
 #Ice Mask polygon .shp file
iceMask = '###INPUT NAME###'
          #^^^insert file name here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
```

```
#Create basins based on Surface Topography only
 #
print "Generating basins"

#clip surface DEM to ice mask extent
 #
outExtractByMask = ExtractByMask(surfaceDEM, iceMask)
outExtractByMaskName = (surfaceDEMString.split("."))[0] + "_clipIce.tif"
outExtractByMask.save(outExtractByMaskName)
print "extract by mask successful. . ."

#fill the DEM
 #
outFill = Fill(outExtractByMask)
outFillName = (outExtractByMaskName.split("."))[0] + "_fill.tif"
outFill.save(outFillName)
print "DEM fill successful. . ."

#calculate each pixels flow direction
 #
outFlowDirection = FlowDirection(outFill, "NORMAL")
outFlowDirectionName = (outFillName.split("."))[0] + "_fdr_NoForce.tif"
outFlowDirection.save(outFlowDirectionName)
print "DEM flow direction successful. . ."

#calculate each pixels flow accumulation
 #
outFlowAccumulation = FlowAccumulation(outFlowDirection)
outFlowAccumulationName = (outFlowDirectionName.split("."))[0] + "_fac.tif"
outFlowAccumulation.save(outFlowAccumulationName)
print "DEM flow accumulation successful. . ."

#generate basin raster
 #
outBasin = Basin(outFlowDirection)
outBasinRasterName = (outFlowDirectionName.split("."))[0] + "_basin.tif"
outBasin.save(outBasinRasterName)
print "basins (raster) successful. . ."

#generate basin vector
 #
outBasinVectorName = (outFlowDirectionName.split("."))[0] + "_basin.shp"
arcpy.RasterToPolygon_conversion(outBasin, outBasinVectorName, "NO_SIMPLIFY")
print "basins (vector) successful. . ."

#delete variables, clear locks
 #
del outExtractByMask, outExtractByMaskName
del outFill, outFillName
del outFlowDirection, outFlowDirectionName
del outBasin, outBasinRasterName, outBasinVectorName

print "Basins surface Complete"
#-------------------------------------------------------------------------------
```

Delineate Ice Sub Basins: Hydraulic Potential

```
#-------------------------------------------------------------------------------
# Name:        GenerateIceSubBasins_HydraulicPotential.py
# Purpose:     Creates drainage basins for Greenland Ice Sheet based on
#              Hydraulic Potential method (Lewis & Smith 2009 and others)
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = "###INPUT FILE PATH###"
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Declare Input Variables
 #
 #Surface DEM
surfaceDEM = "###INPUT NAME###"
             #^^^insert file name here
 #
 #Bed DEM
bedDEM = "###INPUT NAME###"
         #^^^insert file name here
 #
 #Ice Mask
iceMask = "###INPUT NAME###"
          #^^^insert file name here
'''
    NOTE: Hydraulic Potential Method
    Pa = Pi * g * (h + 0.1 * y)
        Pa - subglacial hydraulic potential
```

```
        Pi - density of ice (assumed to be constant 917 kg m^-3)
        g - acceleration due to gravity (9.81 m s^-2)
        h - elevation ice surface, m
        y - elevation of subglacial topography, m
'''
 #Pi - density of ice (assumed to be constant 917 kg m^-3)
Pi = 917
 #
 #g - acceleration due to gravity (9.81 m s^-2)
g = 9.81
 #
 #h - elevation ice surface (surfaceDEM) in m
h = Raster(surfaceDEM)
 #
#y - elevation of subglacial topography (bedDEM) in m
y = Raster(bedDEM)
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
#Create hydraulic potential basins
 #
print "Generating basins. . ."

#clip surface DEM to ice mask extent
 #
outExtractByMaskSurface = ExtractByMask(h, iceMask)
outExtractByMaskSurfaceName = (surfaceDEM.split("."))[0] + "_clipIce.tif"
outExtractByMaskSurface.save(outExtractByMaskSurfaceName)
print "surface DEM clip to ice mask done"

#fill the surface DEM
 #
outFillSurfaceDEM = Fill(outExtractByMaskSurface)
outFillSurfaceDEMName = (outExtractByMaskSurfaceName.split("."))[0] +
"_fill.tif"
outFillSurfaceDEM.save(outFillSurfaceDEMName)
print "surface DEM fill done"

#clip bed DEM to ice mask extent
 #
outExtractByMaskBed = ExtractByMask(y, iceMask)
outExtractByMaskBedName = (bedDEM.split("."))[0] + "_clipIce.tif"
outExtractByMaskBed.save(outExtractByMaskBedName)
print "bedrock DEM clip to ice mask done"

#fill the bed DEM
 #
outFillBedDEM = Fill(outExtractByMaskBed)
outFillBedDEMName = (outExtractByMaskBedName.split("."))[0] + "_fill.tif"
outFillBedDEM.save(outFillBedDEMName)
print "basal DEM fill done"

#generate hydrostatic pressure grid
 #
pgrid = Pi * g * (outFillSurfaceDEM + 0.1 * outFillBedDEM)
pgrid.save("pgrid.tif")
```

```
print "hydraulic potential grid complete"

#fill
 #
outFill = Fill("pgrid.tif")
outFill.save("pgrid_fill.tif")
print "hydraulic potential grid fill complete"

#flow direction (fdr)
 #
outFlowDirection = FlowDirection("pgrid_fill.tif", "NORMAL")
outFlowDirection.save("pgrid_fill_fdr.tif")
print "hydraulic potential flow direction grid done"

#flow accumulation (fac)
 #
outFlowAccumulation = FlowAccumulation("pgrid_fill_fdr.tif", '', 'INTEGER')
outFlowAccumulation.save("pgrid_fill_fdr_fac.tif")
print "hydraulic potential flow accumulation grid done"

#basin raster
 #
outBasin = Basin("pgrid_fill_fdr.tif")
outBasin.save("pgrid_fill_fdr_basin.tif")
print "basins raster done"

#basin vector
 #
arcpy.RasterToPolygon_conversion(outBasin, "pgrid_fill_fdr_basin.shp",
"NO_SIMPLIFY")
print "basins vector done"

#delete variables, clear locks
 #
del outExtractByMaskSurface, outExtractByMaskBed, pgrid, outFill
del outFlowDirection, outFlowAccumulation, outBasin
#---------------------------------------------------------------------------
```

Define Modeled Pour Points for Ice Sub Basins

```
#---------------------------------------------------------------------------
# Name:         DefineModeledPourPointsForIceSubBasins.py
# Purpose:      Defines the modelled pour point, defined as the maximum
#               maximum flow accumulation grid cell contained within every
#               ice sub basin in the input file. This technique uses zonal
#               zonal statistics and a difference raster to define modelled
#               pour points.
#
# Author:       Lincoln H Pitcher
#
# Updated:      26-February-2015
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *
import timeit

#time of script
start = timeit.default_timer()

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
#set file saving parameters
surfaceORpotentiometric = "###INSERT STRING###"
                          #^^^insert: 'surface' or 'hydraulicPotential'
date = "03March2015"
       #^^^insert date: ddMonthYYYY e.g. 01January2015
#---------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------
#Declare Input Variables
 #
 #zone data raster. zones are ice sub basins.
zoneData = "###INPUT NAME###"
           #^^^ice sub basins .tif file, value is each basin id
           #can be surface or hydraulic potential ice sub basins
 #
 # value raster. values are flow accumulation values
valueRaster = "###INPUT NAME###"
              #^^^flow accumulation grid .tif file
              #can be derived from surface or hydraulic potential grid
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#calculate zonal statistics,
 #finds max flow accumulation value in each ice sub basin
 #
zonalStatsOutput = surfaceORpotentiometric + "_zonalStats_" + date + ".tif"
outZonalStats = ZonalStatistics(zoneData,"FID",valueRaster,"MAXIMUM","DATA")
outZonalStats.save(zonalStatsOutput)
print 'zonal stats done'

#calc difference zonal stats output and value raster;
 #0 = MAX FAC point
 #
differenceRasterOutput = surfaceORpotentiometric +
"_zonalStats_difference_FAC_" + date + ".tif"
 #define variable as type rasters
zonalStatsOutputRaster = Raster(zonalStatsOutput)
valueRasterRaster = Raster(valueRaster)
 #Execute Minus
outMinus = Minus(zonalStatsOutputRaster, valueRasterRaster)
 #Save output
outMinus.save(differenceRasterOutput)
print 'difference raster done'

#reclassify difference raster
 #set 0 = MAX FAC point --> 1 = MAX FAC point
 #all else = NoData
reclassifyRasterOutput = surfaceORpotentiometric +
"_zonalStats_difference_FAC_reclass_" + date + ".tif"
 #define variable as type rasters
differenceRasterOutputRaster = Raster(differenceRasterOutput)
arcpy.gp.Reclassify_sa(differenceRasterOutputRaster,"Value","0 1;1 999999999
NODATA",reclassifyRasterOutput,"DATA")
print 'reclassify done'

#conver reclassified MAX FAC to raster to MAX FAC points
outPoints = "iceSubBasin_" + surfaceORpotentiometric + "_maxFACpoints_" +
date + ".shp"
arcpy.RasterToPoint_conversion(in_raster=reclassifyRasterOutput,out_point_fea
tures=outPoints,raster_field="Value")
print 'raster to point conversion done'

#clear variables and data
```

```
del surfaceORpotentiometric, date
del zoneData, valueRaster
del zonalStatsOutput
del differenceRasterOutput
del reclassifyRasterOutput
del outPoints
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#calculate run time of script
 #
stop = timeit.default_timer()
print 'total run time is: ', stop - start
#-------------------------------------------------------------------------------
```

Generate Land Watersheds

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_1_GenerateLandWatersheds.py
# Purpose:     Creates watersheds for pour points in input point file
#              pour points should be pre-snapped to flow accumulation grid.
#              inputs required include: (1) point file with StrmID field
#              (2) flow direction grid from a depression free land DEM.
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Declare Input Variables
 #
points = "###INPUT NAME###"
         #^^^pour points. Must have StrmID field.
 #
fdr = "###INPUT NAME###"
      #^^^flow direction grid from depression free land surface DEM
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# initiate cursor to loop through each pour point in input point file
pointCursor = arcpy.SearchCursor(points)

# loop through each pour point in points file
```

```
for pt in pointCursor:

    # select point by FID and copy to new shapefile
    rowFID = pt.FID
    rowID = pt.StrmID       ###requires StrmID field in pour point file
    where = ' FID = ' + "%s"%rowFID
    ptName="del_" + (points.split("."))[0] + "_FID" + str(rowFID) + "_" +
".shp"
    arcpy.Select_analysis(points, ptName, where)

    # Execute Watershed
    outWatershed = Watershed(fdr, ptName, "Id")

    # Save the output
    watershedTifName = "Watershed_ID" + str(rowID) + ".tif"
    outWatershed.save(watershedTifName)

    # Convert raster to polygon
    watershedPolyName = "Watershed_ID" + str(rowID) + ".shp"
    arcpy.RasterToPolygon_conversion(watershedTifName, watershedPolyName,
"NO_SIMPLIFY", "VALUE")

    print "loop done"
#-------------------------------------------------------------------------------
```

Merge Land Watersheds into one file and populate Pour Point Identifier Attribute Field

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_2_mergeLandWatershedsWithRivID.py
# Purpose:     merges all land watersheds into one .shp file
#              and and appends RivID from the file name to the attribute
table
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Generate list of all polygon .shp files in workSpace that begin with
Watershed
 #
fcList = arcpy.ListFeatureClasses("Watershed*", "Polygon")
print "fc list includes: ", fcList

#loop through each shp file in list and add pour point id
 #
for fc in fcList:
        print "loop for fc : ", fc

        #define Pout Point ID by string name in file
         #
        strID = long(fc[12:16])
```

```
        #add empty Pour Point Field
         #
        arcpy.AddField_management(fc,'pourPoint','long',9)

        #add Pour Point ID to Pour Point field in shp file
         #
        fields = ('pourPoint')
        with arcpy.da.UpdateCursor(fc, fields) as cursor:
                for row in cursor:
                        row[0] = strID
                        cursor.updateRow(row)

        #clear lock file
         #
        del fc

# merge all watersheds into single .shp file
arcpy.Merge_management(fcList, "Watersheds_merge.shp")

#delete unused variables
del fcList
#-------------------------------------------------------------------------------
```

One Pixel Ice Edge Buffer

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_3_onePixelIceEdgePixelBuffer.py
# Purpose:     Clips one-pixel inside ice edge buffer to each watershed and
#              appends the Pour Point Attribute Identifier to the buffer
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#variables
 #watershed file with pour point attribute identifier field
watersheds = "###INPUT NAME###"
             #^^^watersheds .shp file
 #
 #Ice Edge one pixel buffer
mask = "###INPUT NAME###"
       #^^^one pixel buffer .shp file
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#watershed cursor
watershedsCursor = arcpy.SearchCursor(watersheds)

#loop through each basin
print "begin looping through each basin. . ."
```

```
for watershed in watershedsCursor:

    #select basin by FID and copy to new shapefile
    rowFID = watershed.FID
    where = ' FID = ' + "%s"%rowFID
    watershedName="del_" + (watersheds.split("."))[0] + "_FID" + str(rowFID)
+ "_" + ".shp"
    arcpy.Select_analysis(watersheds, watershedName, where)

    #get the pour point id from the attrib table
    pourPoint = watershed.pourPoint

    #clip mask to watershed id
    arcpy.Intersect_analysis ([watershedName, mask],
"iceMask150mBuffer_PourPointID" + str(pourPoint) + ".shp")

    #delete unused fields
    arcpy.DeleteField_management("iceMask150mBuffer_PourPointID" +
str(pourPoint) + ".shp", ["FID_del_Wa", "FID_bedmac", "ID_1", "GRIDCODE_1",
"ORIG_FID"])

    #delete unused watersheds files
    arcpy.Delete_management(watershedName)

    print "loop for watershed with pour point id: ", pourPoint, " complete"

#delete unused variables and remove locks
del watersheds, mask
#-------------------------------------------------------------------------------
```

Define Modeled Pour Points for Ice Sub Basins

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_4_defineModeledPourPointsForIceSubBasins.py
# Purpose:     Select modeled outlets that intersect each watershed buffer
#              append pour point ID to modeled outlets.
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Declare Input variables
 #mask
maskList = arcpy.ListFeatureClasses("iceMask150mBuffer_PourPointID*")

 #surface points and a surface layer for select by location
surfacePoints = "###INPUT NAME###"
                #^^^surface points
arcpy.MakeFeatureLayer_management(surfacePoints, "surfacePoints_lyr")

 #potentiometric points and a potentiometric layer for select by location
potentiometricPoints = "###INPUT NAME###"
                       #^^^hydraulic potential points
arcpy.MakeFeatureLayer_management(potentiometricPoints,
"potentiometricPoints_lyr")
#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------
#loop through mask and select points
for mask in maskList:

        #define pour point ID
        PourPointID = str(mask[29:33])
        PourPointID_int = int(PourPointID)

        #Make fc a layer
        arcpy.MakeFeatureLayer_management(mask, "mask_lyr")

        #Surface ice basins
         #add a selection to point the layer based on location to point mask
layer
        arcpy.SelectLayerByLocation_management ("surfacePoints_lyr",
"INTERSECT", mask, "", "NEW_SELECTION")
         #write selection to new fc
        arcpy.CopyFeatures_management('surfacePoints_lyr',
'surfacePoints_PourPointID' + PourPointID + '.shp')
         #add pourPoint field to new fc
        arcpy.AddField_management('surfacePoints_PourPointID' + PourPointID +
'.shp', 'pourPoint', 'LONG', 9)
         #update row with Pour Point ID
        fields = ('pourPoint')
        with arcpy.da.UpdateCursor('surfacePoints_PourPointID' + PourPointID
+ '.shp', fields) as cursor:
                for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

        #Potentiometric ice basins
         #add a selection to point the layer based on location to point mask
layer
        arcpy.SelectLayerByLocation_management ("potentiometricPoints_lyr",
"INTERSECT", mask, "", "NEW_SELECTION")
         #write selection to new fc
        arcpy.CopyFeatures_management('potentiometricPoints_lyr',
'potentiometricPoints_PourPointID' + PourPointID + '.shp')
         #add pourPoint field to new fc
        arcpy.AddField_management( 'potentiometricPoints_PourPointID' +
PourPointID + '.shp', 'pourPoint', 'LONG', 9)
         #update row with Pour Point ID
        fields = ('pourPoint')
        with arcpy.da.UpdateCursor('potentiometricPoints_PourPointID' +
PourPointID + '.shp', fields) as cursor:
                for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

        print "loop for pour point id: ", PourPointID, " complete"

        #delete variables and clear locks
        del PourPointID, PourPointID_int, fields

#delete variables and clear locks
del maskList, mask, surfacePoints, potentiometricPoints
```

```
#-------------------------------------------------------------------------------
```

For Each Pour Point Merge Edge Ice Sub Basins

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_5_selectEdgeIceBasinsDissovleEliminate.py
# Purpose:     Selects ice sub basins that intersect modelled outlets
#              dissolves basins and eliminated interior polygons
#              only edge basins that intersect one-pixel ice edge buffer
#              are selected.
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
#import modules
import arcpy,os,sys
from arcpy import env
from arcpy.sa import *

#check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")

#Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)

#Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#variables
 #surface point list
 #
surfacePointsList = arcpy.ListFeatureClasses("surfacePoints_PourPointID*")

 #hydraulic potential point list
 #
potentiometricPointsList =
arcpy.ListFeatureClasses("potentiometricPoints_PourPointID*")

 #surface ice sub basins
 #
surfaceBasins = "###INPUT NAME###"
                #^^^ice sub basins, surface
```

```
arcpy.MakeFeatureLayer_management(surfaceBasins, "surfaceBasins_lyr")

 #hydraulic potential ice sub basins
 #
potentiometricBasins = "###INPUT NAME###"
                #^^^ice sub basins, hydraulic potential
arcpy.MakeFeatureLayer_management(potentiometricBasins,
"potentiometricBasins_lyr")

 #land mask
 #
landMask = "###INPUT NAME###"
           #^^^land mask .shp file

 #ice mask buffer
 #
iceMaskBuffer = "###INPUT NAME###"
                #^^^one pixel ice mask buffer
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
#loop through SURFACE points list, dissolve, eliminate, buffer
for points in surfacePointsList:

        #define pour point ID
        PourPointID = str(points[25:29])
        PourPointID_int = int(PourPointID)

        #Make fc a layer
        arcpy.MakeFeatureLayer_management(points, "points_lyr")

        #select by location
         #add a selection to point the layer based on location to point mask
layer
        arcpy.SelectLayerByLocation_management ("surfaceBasins_lyr",
"INTERSECT", points, "", "NEW_SELECTION")
         #write selection to new fc
        arcpy.CopyFeatures_management('surfaceBasins_lyr',
'surfaceIceSubBasins_PourPointID' + PourPointID + '.shp')
         #add pourPoint field to new fc
        arcpy.AddField_management('surfaceIceSubBasins_PourPointID' +
PourPointID + '.shp', 'pourPoint', 'LONG', 9)
         #update row with Pour Point ID
        fields = ('pourPoint')
        with arcpy.da.UpdateCursor('surfaceIceSubBasins_PourPointID' +
PourPointID + '.shp', fields) as cursor:
                for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

        #dissolve
        arcpy.Dissolve_management('surfaceIceSubBasins_PourPointID' +
PourPointID + '.shp', 'surfaceIceSubBasins_PourPointID' + PourPointID +
'_dissolve.shp',"pourPoint")

        #eliminate
```

```
arcpy.EliminatePolygonPart_management('surfaceIceSubBasins_PourPointID' +
PourPointID + '_dissolve.shp', 'surfaceIceSubBasins_PourPointID' +
PourPointID + '_dissolve_elim.shp', 'AREA', 999999000, '','CONTAINED_ONLY')

        #150m outside buffer
        arcpy.Buffer_analysis('surfaceIceSubBasins_PourPointID' + PourPointID
+ '_dissolve_elim.shp', 'surfaceIceSubBasins_PourPointID' + PourPointID +
'_150mBuf.shp', '150 Meters', 'OUTSIDE_ONLY')

        #erase land
        arcpy.Erase_analysis('surfaceIceSubBasins_PourPointID' + PourPointID
+ '_150mBuf.shp', landMask, 'surfaceIceSubBasins_PourPointID' + PourPointID +
'_150mBuf_eraseLand.shp')
         #delete 150m outside buffer
        arcpy.Delete_management('surfaceIceSubBasins_PourPointID' +
PourPointID + '_150mBuf.shp')

        #erase ice perimeter buffer
        arcpy.Erase_analysis('surfaceIceSubBasins_PourPointID' + PourPointID
+
'_150mBuf_eraseLand.shp',iceMaskBuffer,'surfaceIceSubBasins_150mBuf_PourPoint
ID' + PourPointID + '.shp')
         #delete erase land buffer
        arcpy.Delete_management('surfaceIceSubBasins_PourPointID' +
PourPointID + '_150mBuf_eraseLand.shp')

        print "loop for SURFACE basins with pour point id: ", PourPointID, "
complete"

        #delete variables and clear locks
        del PourPointID, PourPointID_int, fields
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
#loop through POTENTIOMETRIC points list, dissolve, eliminate, buffer
for points in potentiometricPointsList:

        #define pour point ID
        PourPointID = str(points[32:36])
        PourPointID_int = int(PourPointID)

        #Make fc a layer
        arcpy.MakeFeatureLayer_management(points, "points_lyr")

        #select by location
         #add a selection to point the layer based on location to point mask
layer
        arcpy.SelectLayerByLocation_management ("potentiometricBasins_lyr",
"INTERSECT", points, "", "NEW_SELECTION")
         #write selection to new fc
        arcpy.CopyFeatures_management('potentiometricBasins_lyr',
'potentiometricIceSubBasins_PourPointID' + PourPointID + '.shp')
         #add pourPoint field to new fc
        arcpy.AddField_management('potentiometricIceSubBasins_PourPointID' +
PourPointID + '.shp', 'pourPoint', 'LONG', 9)
```

```
         #update row with Pour Point ID
        fields = ('pourPoint')
        with arcpy.da.UpdateCursor('potentiometricIceSubBasins_PourPointID' +
PourPointID + '.shp', fields) as cursor:
                for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

        #dissolve
        arcpy.Dissolve_management('potentiometricIceSubBasins_PourPointID' +
PourPointID + '.shp', 'potentiometricIceSubBasins_PourPointID' + PourPointID
+ '_dissolve.shp',"pourPoint")

        #eliminate

arcpy.EliminatePolygonPart_management('potentiometricIceSubBasins_PourPointID
' + PourPointID + '_dissolve.shp', 'potentiometricIceSubBasins_PourPointID' +
PourPointID + '_dissolve_elim.shp', 'AREA', 999999000, '','CONTAINED_ONLY')

        #150m outside buffer
        arcpy.Buffer_analysis('potentiometricIceSubBasins_PourPointID' +
PourPointID + '_dissolve_elim.shp', 'potentiometricIceSubBasins_PourPointID'
+ PourPointID + '_150mBuf.shp', '150 Meters', 'OUTSIDE_ONLY')

        #erase land
        arcpy.Erase_analysis('potentiometricIceSubBasins_PourPointID' +
PourPointID + '_150mBuf.shp', landMask,
'potentiometricIceSubBasins_PourPointID' + PourPointID +
'_150mBuf_eraseLand.shp')
         #delete 150m outside buffer
        arcpy.Delete_management('potentiometricIceSubBasins_PourPointID' +
PourPointID + '_150mBuf.shp')

        #erase ice perimeter buffer
        arcpy.Erase_analysis('potentiometricIceSubBasins_PourPointID' +
PourPointID +
'_150mBuf_eraseLand.shp',iceMaskBuffer,'potentiometricIceSubBasins_150mBuf_Po
urPointID' + PourPointID + '.shp')
         #delete erase land buffer
        arcpy.Delete_management('potentiometricIceSubBasins_PourPointID' +
PourPointID + '_150mBuf_eraseLand.shp')

        print "loop for POTENTIOMETRIC basins with pour point id: ",
PourPointID, " complete"

        #delete variables and clear locks
        del PourPointID, PourPointID_int, fields
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#delete variables and clear locks
del surfacePointsList, potentiometricPointsList, surfaceBasins,
potentiometricBasins
#-------------------------------------------------------------------------------
```

Iterative Check for Contributing Internal Ice Sub Basin and CryoSheds Growing Routine

```
#-------------------------------------------------------------------------------
# Name:        CryoSheds_6_internalBasinCheckAndGrow.py
# Purpose:     check for internal drained basins, when found, merge if flow
#              into basin
#
# Author:      Lincoln H Pitcher
#
# Updated:     26-February-2015
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#define folder path
workSpace = '###INPUT FILE PATH###'
            #^^^insert file path here
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#Environmental Set Up
 #import modules
import arcpy, os, sys
from arcpy import env
from arcpy.sa import *
 #
 #check out spatial analyst extension
arcpy.CheckOutExtension("Spatial")
 #
 #Set Workspace
arcpy.env.workspace = workSpace
outputPath = workSpace
if os.path.exists(workSpace)==False:
        os.mkdir(workSpace)
 #
 #Allow Overwriting Output Files
arcpy.env.overwriteOutput = True
print "environmental set up complete"
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# INPUTS
 # basin list
basinList = arcpy.ListFeatureClasses("###String w/ Wild-card For File
List###")
                                     #^^^string and wild-card for file list
 #
 # land mask
landMask = "###INPUT NAME###"
           #^^^land mask .shp file
 #
 # ice mask
iceMask = "###INPUT NAME###"
           #^^^ice mask .shp file
 # modelled pour points
max_fac_points = "###INPUT NAME###"
           #^^^modelled pour points
```

```
arcpy.MakeFeatureLayer_management(max_fac_points, "max_fac_points_lyr")
 #
 #ice sub basins
iceSubBasin = "###INPUT NAME###"
              #^^^ice sub basins .shp file
arcpy.MakeFeatureLayer_management(iceSubBasin, "iceSubBasin_lyr")
 #
 #constant value raster
CVRaster = "CVRaster0.tif"
   #a constant value raster
   #with pixel size = grid DEM
   #all pixel values == 0
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
#initiate loop through all basins in the basins list
for basin in basinList:
    print 'loop for basin: ', basin
    whileCondition = 1

    basinCheck = basin

    #define pour point ID
     #If surface = 31:35
     #If Hydraulic Potential = 38:42
    PourPointID = str(basin[31:35])
    PourPointID_int = int(PourPointID)
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
    #set while condition
    while whileCondition == 1:
        #CHECK IF THERE ARE ANY INTERNALLY DRAINED BASINS CONTRIBUTING TO
BASIN
        #150m outside buffer
        buffer_1 = (basin.split('.'))[0] + '_150mBuf.shp'
        arcpy.Buffer_analysis(basinCheck, buffer_1, '150
Meters','OUTSIDE_ONLY')
         #
        #erase land
        buffer_2 = (basin.split('.'))[0] + '_150mBuf_eraseLand.shp'
        arcpy.Erase_analysis(buffer_1, landMask, buffer_2)
        #delete buffer 1 file b/c no longer needed
        arcpy.Delete_management(buffer_1)
         #
        #erase ice perimeter buffer
        buffer_3 = (basin.split('.'))[0] + '_150mBuf_eraseLand_eraseIce.shp'
        arcpy.Erase_analysis(buffer_2,iceMask,buffer_3)
        #delete buffer_2 file b/c no longer needed
        arcpy.Delete_management(buffer_2)
         #
        #erase inside of basin
        buffer_4 = (basin.split('.'))[0] +
'_150mBuf_eraseLand_eraseIce_eraseBasin.shp'
        arcpy.Erase_analysis(buffer_3, basinCheck, buffer_4)
        #delete buffer_3 file b/c no longer needed
```

```
        arcpy.Delete_management(buffer_3)
         #
        #select intersecting points by location
        #add a selection to point the layer based on location to point mask
layer
        arcpy.SelectLayerByLocation_management
("max_fac_points_lyr","INTERSECT",buffer_4,"","NEW_SELECTION")
         #
        #declare BUFFER COUNTER (count) by counting number of selected
features
        result = arcpy.GetCount_management("max_fac_points_lyr")
        count = int(result.getOutput(0))
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
        if count>0:
            #check if selected points flow into the basin
            print 'count = ', count
            #COPY SELECTED POINTS TO NEW FC
            points_selection = (max_fac_points.split('.'))[0] + '_ RivID' +
str(PourPointID) + '.shp'
            arcpy.CopyFeatures_management('max_fac_points_lyr',
points_selection)

            #CREATE LAND COVER MASK
             #convert basin shp file to tif
            arcpy.PolygonToRaster_conversion(basin, "pourPoint",
"temp_basin.tif", "CELL_CENTER", "", 150)
             #convert buffer shp file to tif
            arcpy.PolygonToRaster_conversion(buffer_4, "pourPoint",
"temp_buf.tif", "CELL_CENTER", "", 150)
             #reclassify buffer tif, make all values -9999
            outReclass1 = Reclassify("temp_buf.tif", "Value",
RemapValue([[PourPointID_int,-9999]]))
            outReclass1.save("temp_buf_reclass.tif")
             #delete original buffer tif file b/c no longer needed
            arcpy.Delete_management("temp_buf.tif")
             #mosaic raster

arcpy.MosaicToNewRaster_management("temp_basin.tif;temp_buf_reclass.tif;CVRas
ter0.tif",workSpace,"temp_mosaic.tif","WGS_1984_Stereographic_North_Pole.prj"
,"64_BIT","","1","FIRST")
            lCover = "temp_mosaic.tif"

            #FIND LC VALUE
            #copy points and make cursor
            pointsCopy = (points_selection.split("."))[0] + "_LC.shp"
            arcpy.CopyFeatures_management(points_selection, pointsCopy)

            #add XY to points
            arcpy.AddXY_management(pointsCopy)

            pointCursor = arcpy.UpdateCursor(pointsCopy)
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
```

```
            #loop through each point in pointsCopy
            for point in pointCursor:

                #get point rasterValue, X and Y coordinates
                pointX = point.POINT_X
                pointY = point.POINT_Y
                pointFDR = point.FDR
                pointFID = point.FID
                point_LCValue = point.LCValue

                print '     looping through points with: X = ', pointX, ' Y =
', pointY, ' FDR = ', pointFDR, ' LCValue = ', point_LCValue, ' FID = ',
pointFID
                print '     ', pointFID

                #declare point variables
                     #if LC == 1 & FDR == 1
                pointX_new_FDR1 = pointX + 90
                pointY_new_FDR1 = pointY
                     #if LC == 1 & FDR == 2
                pointX_new_FDR2 = pointX + 90
                pointY_new_FDR2 = pointY - 90
                     #if LC == 1 & FDR == 4
                pointX_new_FDR4 = pointX
                pointY_new_FDR4 = pointY - 90
                     #if LC == 1 & FDR == 8
                pointX_new_FDR8 = pointX - 90
                pointY_new_FDR8 = pointY - 90
                    #if LC == 1 & FDR == 16
                pointX_new_FDR16 = pointX - 90
                pointY_new_FDR16 = pointY
                    #if LC == 1 & FDR == 32
                pointX_new_FDR32 = pointX - 90
                pointY_new_FDR32 = pointY + 90
                    #if LC == 1 & FDR == 64
                pointX_new_FDR64 = pointX
                pointY_new_FDR64 = pointY + 90
                    #if LC == 1 & FDR == 128
                pointX_new_FDR128 = pointX + 90
                pointY_new_FDR128 = pointY + 90
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
                while(point_LCValue < -1):
                    if pointFDR == 1:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR1, pointY_new_FDR1)]
                        insertCursor.insertRow(newPoint)
```

```
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        pointX_new_FDR1 = pointX_new_FDR1 + 90
                        pointY_new_FDR1 = pointY_new_FDR1

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    elif pointFDR == 2:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR2, pointY_new_FDR2)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR2 = pointX_new_FDR2 + 90
                        pointY_new_FDR2 = pointY_new_FDR2 - 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor


                    elif pointFDR == 4:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
```

```
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR4, pointY_new_FDR4)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR4 = pointX_new_FDR4
                        pointY_new_FDR4 = pointY_new_FDR4 - 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    elif pointFDR == 8:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR8, pointY_new_FDR8)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR8 = pointX_new_FDR8 - 90
                        pointY_new_FDR8 = pointY_new_FDR8 - 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    elif pointFDR == 16:
                        print '          point FDR = ', pointFDR
                        #create a temp point
```

```
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR16, pointY_new_FDR16)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR16 = pointX_new_FDR16 - 90
                        pointY_new_FDR16 = pointY_new_FDR16

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    elif pointFDR == 32:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR32, pointY_new_FDR32)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR32 = pointX_new_FDR32 - 90
                        pointY_new_FDR32 = pointY_new_FDR32 + 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor
```

```
                    elif pointFDR == 64:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR64, pointY_new_FDR64)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR64 = pointX_new_FDR64
                        pointY_new_FDR64 = pointY_new_FDR64 + 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    elif pointFDR == 128:
                        print '          point FDR = ', pointFDR
                        #create a temp point
                        tempPoint = ("temp_FDR" + str(pointFDR) +
str(pointFID) + ".shp")
                        arcpy.CreateFeatureclass_management(workSpace,
tempPoint, "Point", pointsCopy, "SAME_AS_TEMPLATE", "SAME_AS_TEMPLATE",
pointsCopy)
                        insertCursor = arcpy.da.InsertCursor(tempPoint,
("SHAPE@XY"))
                        newPoint = [(pointX_new_FDR128, pointY_new_FDR128)]
                        insertCursor.insertRow(newPoint)
                        #get land cover classification for new temp point
                        tempPoint_lCover = (tempPoint.split("."))[0] +
"_lCover.shp"
                        ExtractValuesToPoints(tempPoint, lCover,
tempPoint_lCover, "NONE", "VALUE_ONLY")
                        #access Land Cover Value in new temp point
                        searchCursor = arcpy.SearchCursor(tempPoint_lCover)
                        for tempPoint in searchCursor:
                                lValue = tempPoint.RASTERVALU
                                print '          point lValue = ', lValue
                        #calculate X,Y for new point, projected to next cell
in direction of flow
                        pointX_new_FDR128 = pointX_new_FDR128 + 90
```

```
                        pointY_new_FDR128 = pointY_new_FDR128 + 90

                        del tempPoint, insertCursor, newPoint,
tempPoint_lCover, searchCursor

                    point.setValue("LCValue", lValue)
                    pointCursor.updateRow(point)
                    point_LCValue = lValue
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
            if point_LCValue == 0:
                whileCondition = 0
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
            else:
                #select points by attrib with LCValue = Basin ID Value
                where = ' LCValue = ' + "%s"%PourPointID
                pointsCopySelect=(points_selection.split("."))[0] +
"_LC_Select.shp"
                arcpy.Select_analysis(pointsCopy, pointsCopySelect, where)

                #spatial join with ice sub basins
                arcpy.SelectLayerByLocation_management ("iceSubBasin_lyr",
"INTERSECT", pointsCopySelect, "", "NEW_SELECTION")
                iceSubBasinSelect = (iceSubBasin.split('.'))[0] + '_SelectID'
+ PourPointID + '.shp'
                arcpy.CopyFeatures_management("iceSubBasin_lyr",
iceSubBasinSelect)

                #update row with Pour Point ID
                arcpy.AddField_management(pointsCopySelect, 'pourPoint',
'LONG', 9)
                fields = ('pourPoint')
                with arcpy.da.UpdateCursor(pointsCopySelect, fields) as
cursor:
                    for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

                #merge
                basinsMerge = (basin.split('.'))[0]+ '_SelectID' +
PourPointID + '_merge.shp'
                arcpy.Merge_management([basin, iceSubBasinSelect],
basinsMerge)

                #update row with Pour Point ID
                fields = ('pourPoint')
                with arcpy.da.UpdateCursor(basinsMerge, fields) as cursor:
                    for row in cursor:
                        row[0] = PourPointID_int
                        cursor.updateRow(row)

                #dissolve
```

```
                basinsDissolve = (basin.split('.'))[0]+ '_SelectID' +
PourPointID + '_merge_dissolve.shp'
                arcpy.Dissolve_management(basinsMerge,
basinsDissolve,"pourPoint")

                #elim
                basinsElim = (basin.split('.'))[0]+ '_SelectID' + PourPointID
+ '_merge_dissolve_elim.shp'
                arcpy.EliminatePolygonPart_management(basinsDissolve,
basinsElim, 'AREA', 999999000, '','CONTAINED_ONLY')

                #reset basin check variable
                basinCheck = basinsElim
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
        else:
            whileCondition = 0
#-------------------------------------------------------------------------------
```

References:

Bamber, J.L. et al., 2013. Paleofluvial Mega-Canyon Beneath the Central Greenland Ice Sheet. *Science*, 341(6149), pp.997–999.

Band, L.E. et al., 2000. Modelling Watersheds as Spatial Object Hierarchies: Structure and Dynamics. *Transactions in GIS*, 4(3), pp.181–196. Available at: http://dx.doi.org/10.1111/1467-9671.00048.

Band, L.E., 1986. Topographic Partition of Watersheds with Digital Elevation Models. *Water Resources Research*, 22(1), pp.15–24. Available at: http://dx.doi.org/10.1029/WR022i001p00015.

Banwell, A.F., Willis, I.C. & Arnold, N.S., 2013. Modeling subglacial water routing at Paakitsoq, W Greenland. *Journal of Geophysical Research*, 118(3), pp.1282–1295.

Bjornsson, H., 1986. Delineation Of Glacier Drainage Basins On Western Vatnajokull. *Annals of Glaciology*, 8, pp.19 – 21.

Collins, S.H., 1975. Terrain parameters directly from a digital terrain model. *Canadian Surveyor*, 9, pp.507–518.

Cuffey, K.M. & Paterson, W.S.B., 2010. *The Physics of Glaciers* 4th Editio.,

Hasholt, B. et al., 2013. Observations of Runoff and Sediment and Dissolved Loads from the Greenland Ice Sheet at Kangerlussuaq, West Greenland, 2007 to 2010. *Zeitschrift für Geomorphologie, Supplementary Issues*, 57(2), pp.3–27.

Howat, I.M., Negrete, A. & Smith, B.E., 2014. The Greenland Ice Mapping Project (GIMP) land classification and surface elevation datasets. *The Cryosphere*.

Jenson, S.K. & Domingue, J.O., 1988. Extracting Topographic Structure from Digital Elevation Data for Geographic Information System Analysis. *Photogrammetric Engineering & Remote Sensing*, 54(11), pp.1593–1600.

Lewis, S.M. & Smith, L.C., 2009. Hydrologic drainage of the Greenland Ice Sheet. *Hydrological Processes*, 23(14), pp.2004 – 2011.

Mark, D.M., 1984. Part 4: Mathematical, Algorithmic and Data Structure Issues: Automated Detection Of Drainage Networks From Digital Elevation Models. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 21(2), pp.168–178. Available at: http://dx.doi.org/10.3138/10LM-4435-6310-251R.

Marks, D., Dozier, J. & Frew, J., 1983. Automated Basin Delineation from Digital Terrain Data. *NASA Technical Memorandum*.

Mernild, S.H. & Hasholt, B., 2009. Observed runoff, jökulhlaups and suspended sediment load from the Greenland ice sheet at Kangerlussuaq, West Greenland, 2007 and 2008. *Journal of Glaciology*, 55(193), pp.855–858.

Mernild, S.H. & Liston, G.E., 2012. Greenland freshwater runoff. part ii: distribution and trends, 1960–2010. *Journal of Climate*, 25(17), pp.6015–6035.

Moore, G.K. et al., 1983. *Application of digital mapping technology to the display of hydrologic information; a proof-of-concept test in the Fox-Wolf River Basin, Wisconsin*,

Morlighem, M. et al., 2011. A mass conservation approach for mapping glacier ice thickness. *Geophysical Research Letters*, 38(19). Available at: http://dx.doi.org/10.1029/2011GL048659.

Morlighem, M. et al., 2014. Deeply incised submarine glacial valleys beneath the Greenland ice sheet. *Nature Geoscience*. Available at: http://dx.doi.org/10.1038/ngeo2167.

Morlighem, M. et al., 2015. IceBridge BedMachine Greenland, bed, surface, thickness, mask. *Boulder, Colorado USA: NASA DAAC at the National Snow and Ice Data Center.* Available at: http://nsidc.org/data/docs/daac/icebridge/idbmg4/index.html.

Morse, S.P., 1968. A Mathematical Model for the Analysis of Contour-Line Data. *Journal of the ACM*, 15(2), pp.205–220. Available at: http://doi.acm.org/10.1145/321450.321454.

O'Callaghan, J.F. & Mark, D.M., 1984. The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, 28(3), pp.323–344. Available at: http://www.sciencedirect.com/science/article/pii/S0734189X84800110 [Accessed March 3, 2015].

Paterson, W.S.B., 1993. *The Physics of of Glaciers* 2nd Editio., Elsevier Science Ltd.

Peucker, T.K. & Douglas, D.H., 1975. Detection of Surface-Specific Points by Local Parallel Processing of Discrete Terrain Elevation Data. *Computer Graphics and Image Processing*, 4(4), pp.375–387. Available at: http://www.sciencedirect.com/science/article/pii/0146664X75900052.

Rennermalm, A.K. et al., 2012a. Proglacial river dataset from the Akuliarusiarsuup Kuua River northern tributary, Southwest Greenland, 2008 - 2010, version 1.0. *PANGEA*.

Rennermalm, A.K. et al., 2012b. Proglacial river dataset from the Akuliarusiarsuup Kuua River northern tributary, Southwest Greenland, 2008 - 2010. *Earth System Science Data*, 4(1), pp.1–12.

Rennermalm, A.K. et al., 2014. Proglacial river dataset from the Akuliarusiarsuup Kuua River northern tributary, Southwest Greenland, 2008 - 2013, version 2.0. *PANGEA*.

Rennermalm, A.K. et al., 2013. Understanding Greenland ice sheet hydrology using an integrated multi-scale approach. *Environmental Research Letters*, 8, p.14pp.

Rippin, D. et al., 2003. Changes in geometry and subglacial drainage of Midre Lovénbreen, Svalbard, determined from digital elevation models. *Earth Surface Processes and Landforms*, 28(3), pp.273–298.

Shreve, R.L., 1972. Movement of water in glaciers. *Journal of Glaciology*, 11(62), pp.205–214.

Smith, L.C. et al., 2015. Efficient meltwater drainage through supraglacial streams and rivers on the southwest Greenland Ice Sheet. *Proceedings of the National Academy of Sciences*.

Tarboton, D.G., Bras, R.L. & Rodriguez-Iturbe, I., 1991. On the extraction of channel networks from digital elevation data. *Hydrological Processes*, 5(1), pp.81–100. Available at: http://dx.doi.org/10.1002/hyp.3360050107.

Wilson, J.P., Lam, C.S. & Deng, Y., 2007. Comparison of the performance of flow-routing algorithms used in GIS-based hydrologic analysis. *Hydrological Processes*, 21(8).